Logics for Actor Networks: A two-stage constrained-hybridisation approach

José Luiz Fiadeiro[a,*], Ionuţ Ţuţu[a,b], Antónia Lopes[c], Dusko Pavlovic[d]

[a]Department of Computer Science, Royal Holloway University of London, UK
[b]Simion Stoilow Institute of Mathematics of the Romanian Academy, Bucharest, Romania
[c]LASIGE and Faculdade Ciências, Universidade de Lisboa, Portugal
[d]Department of Information and Computer Sciences, University of Hawaii, USA

Abstract

Actor Networks are a modelling framework for cyber-physical-system protocols based on Latour's actor-network theory that addresses the way we now create and exploit the power of networks whose components are no longer limited to programs, but can also include humans and physical artefacts as actors. The main contribution of this paper is a logic for modelling and reasoning about such actor networks that results from a two-stage constrained-hybridisation process: the first stage corresponds to a logic that captures the structure of actor networks and the way knowledge or data flows across them; the second addresses their dynamic aspects, i.e., the way actor networks can evolve as a result of the interactions that occur within them. For each of these stages, we develop a sound and complete proof system, and we illustrate how the framework can be used for modelling and analysing properties of cyber-physical-system protocols. This two-stage constrained-hybridisation process advances the theoretical and practical aspects of hybrid logics by providing new insights and results that go beyond the specific domain of actor networks. On the other hand, and in line with Milner's bigraph paradigm, the paper also makes a novel contribution to the development of formal methods for systems where connectivity and locality play a fundamental role.

Keywords: Actor network, Cyber-physical-system protocol, Bigraph, Hybrid logic

1. Introduction

Hybrid Logics. Over the past few years, there has been a renewed interest in modal logics for computer science through the new family of 'hybrid logics' (see [1] for a comprehensive overview), which originated in Arthur Prior's work in the 1960s [2]. In their most basic form, these are logics obtained by enriching ordinary modal logics with nominals – symbols that name individual states (possible worlds) in Kripke models – and a dedicated satisfaction operator @, sometimes denoted by a colon, which enables a change of perspective from the current state to one that is named. A significant body of research exists around the model and proof theories of this class of logics, among which [3, 4, 5, 6, 7] are just a few examples of recent developments that are related to the present study.

One of the applications of hybrid logics that relates to aspects of complex network structures that are of interest to us concerns the specification and verification of reconfigurable systems [8]. In a nutshell, the idea is that system configurations (and the functionalities associated with them) can be regarded as local models of a Kripke structure, and that they can change simply by switching from one mode of operation to another via an accessibility relation. The key advancement here lies in the fact that the characteristic features of the basic hybrid logic can be developed, through a process known as hybridisation [9], on top of an arbitrary logical system used for expressing configuration-specific requirements. This means that, depending on the base logic, configurations can be captured, for example, as algebras, as relational structures or, when the hybridisation process is iterated, even as Kripke models.

[*] Corresponding author

Email addresses: jose.fiadeiro@rhul.ac.uk (José Luiz Fiadeiro), ittutu@gmail.com (Ionuţ Ţuţu), malopes@ciencias.ulisboa.pt (Antónia Lopes), dusko@hawaii.edu (Dusko Pavlovic)

Preprint submitted to Journal of Logical and Algebraic Methods in Programming *February 8, 2019*

Actor Networks. Our interest in this process of hybridisation results from a new modelling framework for cyber-physical-system protocols proposed in [10] around the concept of Actor Network (which we usually abbreviate as ANT). ANTs address networks whose components are no longer limited to programs but can also include humans and physical artefacts as actors. Examples include the authentication protocols used nowadays in online banking, which involve bank customers, smart cards, smart-card readers and mobile phones.

ANTs should be understood in the wider sense of Latour's actor-network theory [11]: actors are cyber-physical entities that have shared agency – from people, to objects, and to locations; they interact through so-called channels, which account, for example, for data or knowledge that an actor may transmit to another, of control that an actor may exert on another, or of movement of an actor inside another (e.g., a person moving into a location).

Connectivity & Locality. ANTs respond to the way we can now create and exploit the power of computational networks. In just under three decades, these have evolved from small ad-hoc networks of computers, primarily developed for distributed calculations, to large-scale cyber-physical networks where interaction, rather than computation, has become the norm, giving rise to new challenges in ensuring the reliability of the systems that are now operating in cyberspace.

One notable example of mathematical formalism that has been proposed for this new class of systems is Milner's bigraphs [12], which have at their core the need to model and analyse systems in terms of two key orthogonal notions: connectivity (actors can be connected to other actors, and these connections can change in time) and locality (actors have a location and can move in space). The underlying semantic model of actor networks (explained in Section 2.4) makes this same distinction: it is a particularisation of bigraphs that is suitable for the kinds of closed protocols that arise in many classes of cyber-physical systems, including those mentioned above. As discussed in the concluding remarks, we are now working on extending our logics to open protocols as enabled by bigraphs, i.e., to systems in which locality and connectivity are not bound at design time.

Scope of research. The ANT-based approach that we consider in this paper enables us to address some of the key aspects of cyber-physical systems, but not all. For example, concerns such as continuity or failure, which are relevant for some cyber-physical-system protocols, are not addressed by this theory. In fact, cyber-physical systems have become so prevalent, and have so many distinct aspects that can be interpreted and studied in fundamentally different ways, that different approaches or interpretations are being put forward, each of which is suitable for a particular set of aspects (see, for example, [13]). For actor-network theory, those defining aspects are the dynamic evolution of the potential interactions between actors and the way in which locality and connectivity interact with each other.

ANTs are discrete-dynamical models in the sense that they account for discrete changes (in connectivity or locality) in cyber-physical networks, and that the physical space is represented at an abstract level. This provides support for dealing with the movement of actors (under a discrete interpretation) by means of network reconfigurations. For other certain kinds of properties of cyber-physical systems, one might need to account instead for their continuous evolution in a continuous space (e.g., in the context of collision avoidance for the ETCS, the European Train Control System); in a similar way, reasoning about properties such as reliability may require support for other features, like uncertainty.

In the literature, one can find a variety of mathematical formalisms and modelling frameworks for cyber-physical systems, many of which consider hybrid-dynamical[1] models (see, e.g., [14, 15, 16]). Among those, [17] and [18] are particularly relevant for the developments that we present herein. The former extends differential dynamic logic with hybrid features that are meant to allow for the explicit representation of the states of cyber-physical systems. Hence, the base logic is different from the one we consider here, and the hybrid features added are one-kinded: they can account for the states of such systems, but not for actors as well. The latter paper deals with a hierarchical hybrid logic (in the context of reconfigurable systems) that does provide support for both actors and system states; however, it does not consider constrained models, and thus it cannot faithfully capture ANTs. Moreover, reasoning is done through a translation into first-order logic, whereas we develop a dedicated hybrid-logic proof system.

Contributions. The ordinary hybridisation process outlined above yields logical systems that are suitable for dealing with the structural aspects of actor networks. For example, they can be used for giving faithful descriptions of the shapes of networks, of the (states of the) actors involved, or of the channels through which interactions can take place. However, in contrast to the general adequacy of hybrid logics to cope with reconfigurations, the challenge raised

[1] The word *hydrid* refers in this case to a combination of discrete & continuous modelling techniques for state change, not to hybrid logic.

by ANts lies precisely in capturing the way networks evolve: the way in which actors move in the space of possible locations and the way their connections change in time. This is because the higher-level reconfigurations of networks and the lower-level interactions between actors are closely intertwined: reconfigurations (in connectivity or locality) are triggered by interactions, and they may result in network states/configurations where there are new opportunities for interaction between actors. Therefore, we propose a new hybridisation step that takes as input a hybrid logic, used for specifying states, and produces another logic with hybrid features for specifying network reconfigurations.

The paper also makes a novel contribution to the development of formal methods for bigraph-like semantic domains. Whereas logics have been developed for bigraphs (e.g., [19], [20]), they are process calculi akin to CSP, CCS, the π-calculus or the ambient calculus, i.e., they are operational characterisations based on terms (representing, say, network configurations) and rules through which such terms can be rewritten (thus accounting for network reconfigurations). Instead, our use of logic is akin to, say, that of temporal logics for modelling and analysing system behaviour in the tradition initiated by Pnueli [21]: the interest is in logics whose sentences express, declaratively, properties of a system, and on inference rules that can be used to infer properties from a system specification. To the best of our knowledge, such a declarative approach has not yet been explored for bigraph-like semantic domains. Our conclusion is that the two-stage constrained-hybridisation process that we propose is particularly well-suited for dealing with both the structural and the dynamic aspects of semantic domains that capture connectivity and locality.

Structure of the paper. The paper consists of three main technical sections. In Section 2, we introduce the underlying model theory of actor networks. We start by formalising the main static concepts: actors, the channels through which actors can interact, the knowledge that actors may have and the way it may be acquired across certain channels, and the placement of actors relative to other actors. Then, we formalize the key notion of interaction and the way an interaction can become enabled, and hence can later change the states of a network.

Section 3 is dedicated to the two-stage constrained hybridisation process, which results in three different logics: the basic logic (first level) characterizes the knowledge/data structures of actors; the second level characterizes connectivity (how actors are connected) and locality (where actors are placed) in a given state; and the third level characterizes the dynamics that result from interactions (how actors change location or their connections), i.e., how interactions reconfigure states. Each level captures a different aspect of actor networks (knowledge representation, structure, or dynamics), and each is defined as an exogenous enrichment (in the sense of [22]) of the previous level. This means that, for each step of the hybridisation process, we work with higher-level models consisting of (collections of) lower-level models – which may not be arbitrary, but subject to various semantic constraints whose role is to ensure that the higher-level models are well defined – together with an additional structure that is specific to the hybridisation.

In Section 4, we show how, at each level, those logics can be used to specify properties of protocols and to derive properties of specifications using inference rules. Throughout the paper, we use an elevator as a case study; this is simple enough in order to discuss and provide a model in a few pages, and yet sufficiently rich to illustrate the capabilities of the ANts modelling framework and of the logics that we put forward to support it.

This paper is a revised and extended version of [23]. Section 4 is new, and all the other sections were expanded with more motivation, examples, and explanations of the main results. The proof systems in Section 3 were also revised, and proofs were added for the key, ANt-related theoretical results that cannot be found in the literature.

On notations. Most of the structures we deal with in this paper are presented as tuples – whose components, in turn, may also be tuple-based structures – that satisfy certain cohesion properties. To keep the notations as simple as possible, and to avoid persistently spelling out all the components of a given structure, we make use of subscripts. For example, we may denote the set $\mathcal{N}$ of nodes of a graph $\mathcal{G}$ by $\mathcal{N}_{\mathcal{G}}$, the underlying graph $\mathcal{G}$ of an ANt schema $\mathcal{A}$ by $\mathcal{G}_{\mathcal{A}}$, and the domain $\mathcal{D}$ of an actor network ν by $\mathcal{D}_{\nu}$. When there is no risk of confusion, we overload this notation in order to refer to the hereditary components of a structure. That is, we may denote, for example, the set $\mathcal{N}$ of nodes of the underlying graph of an ANt schema $\mathcal{A}$ by $\mathcal{N}_{\mathcal{A}}$ – even if $\mathcal{N}$ is not a direct component of $\mathcal{A}$, but of its underlying graph.

2. Actor Networks

Actor Networks (ANts) were originally proposed as a framework for modelling cyber-physical-system protocols in [10]. They are based on Latour's actor-network theory [11] in recognition of the fact that such protocols involve a number of entities (called actors, which in concrete situations may correspond to people, devices, locations, etc.)

that have shared agency, and for which interaction, rather than computation, is the major concern. In this section, we provide a model theory for ANts, i.e., a number of mathematical structures for capturing the abstractions through which one can model cyber-physical-system protocols from the vantage point of connectivity and locality.

2.1. Schemas

We start by defining the structures that capture the static aspects of ANts: the actors that are involved in an ANt and the means through which they can exchange knowledge of the domain over which they operate, independently of the way in which they actually operate.

Definition 1 (Schema). An *ANt schema* $\mathcal{A}$ consists of:

- A finite directed graph $\mathcal{G} = \langle \mathcal{N}, C, \delta, \rho \rangle$, where: *(a)* $\mathcal{N}$ is a non-empty set (of nodes, called *actors*), *(b)* C is a set (of edges, called *channels*), and *(c)* δ and ρ are maps $C \to \mathcal{N}$ that give the domain (also referred to as the *origin*) and the codomain (also referred to as the *target*) of every channel.
- A partially ordered set $\mathcal{T}$ (of *channel types*), with a subtype relationship that we denote by $\leq$.
- A function $\tau \colon C \to 2^{\mathcal{T}}$ that assigns a non-empty upper set[2] of channel types to every channel, such that for all actors $n, n' \in \mathcal{N}$ and types $\kappa \in \mathcal{T}$ there is at most one channel $c \in C$ such that $\delta(c) = n$, $\rho(c) = n'$, and $\kappa \in \tau(c)$. This means that every channel $c \in C$ can be identified by the triple $\langle \delta(c), \tau(c), \rho(c) \rangle$.
- A set $\mathcal{P}$ (of *propositional symbols*), disjoint from $\mathcal{N}$.

The nodes of an ANt schema represent *actors* executing a given protocol and the edges represent the *channels* through which those actors can be connected. Channels are typed in order to account for different modes of relationships between actors. For example, channels can account for *observations* that an actor may make of another, *control* that one actor may exert on another, or *movement* of one actor inside another, say a person moving to a certain location. The fact that a channel accounts for an observation, control, movement or any combination of these is captured by the types associated with it. Channels are oriented to indicate the direction in which information, control, or movement flows.

The propositional symbols are used to represent knowledge that is held by the different actors, including data. Pieces of data (or knowledge) have by themselves no agency in the context of the protocol, otherwise they would be actors; for example, in a given protocol, money could be data but, in another protocol, bank notes could be actors, in the sense that they can change hands, be lost, and so on. Knowledge/data can be transmitted across channels as appropriate.

Example 2. Consider an elevator serving a building with two floors.[3] The static aspects of a protocol involving the elevator and a user are captured by the ANt schema $\mathsf{Elevator} = \langle \mathcal{G}_{\mathsf{Elevator}}, \mathcal{T}_{\mathsf{Elevator}}, \tau_{\mathsf{Elevator}}, \mathcal{P}_{\mathsf{Elevator}} \rangle$ defined as follows:

$\mathcal{G}_{\mathit{Elevator}}$ — The underlying graph of Elevator is depicted on the left-hand side of Figure 1. In particular, its set of nodes is $\mathcal{N}_{\mathsf{Elevator}} = \{\mathit{F0}, \mathit{F1}, E, C, \mathit{P0}, \mathit{P1}, A\}$ where:

- the nodes *F0* and *F1* correspond to the ground and first floor of a building, and *E* to the elevator proper (including the shaft and the control system), which we often refer to as Elevator unless it is ambiguous;
- the node *C* corresponds to the elevator's cabin, which we often refer to as Cabin, and *P0* and *P1* correspond to the two platforms where the cabin can be – *P0* for the ground floor, and *P1* for the first floor;
- the node *A* represents a user of the elevator, which we often refer to as Alice.

$\mathcal{T}_{\mathit{Elevator}}$ *and* τ_{Elevator} — The type hierarchy of Elevator is depicted on the right-hand side of Figure 1; its set of channel types is $\mathcal{T}_{\mathsf{Elevator}} = \{\mathit{mov}, \mathit{door}, \mathit{obs}, \mathit{ctr}, \mathit{btn}\}$, where *btn* is a subtype of *ctr*, and *door* is a subtype of *mov*.

The set C_{Elevator} of edges of $\mathcal{G}_{\mathsf{Elevator}}$ and the typing function τ_{Elevator} are depicted on the left-hand side. In order to avoid cluttering the graphic representation, we adopt the following conventions: on the one hand, we only

[2] We recall that an *upper set* of $\mathcal{T}$ is a subset U of $\mathcal{T}$ that contains all channel types $\kappa' \in \mathcal{T}$ for which there exists $\kappa \in U$ such that $\kappa \leq \kappa'$.

[3] As one would expect, the techniques we present in this paper can be used for any (fixed) number of floors. To keep the running example simple, and to avoid unnecessarily large models and long proofs in the following sections, we restrict the number of floors of the elevator system to two.

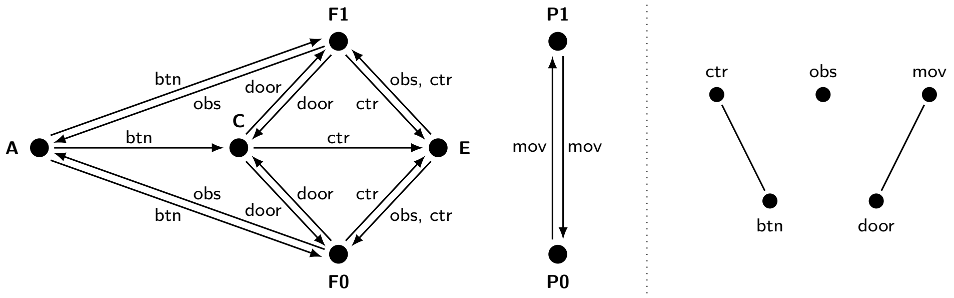

Figure 1: The graph and typing function of the ANt schema Elevator on the left, and the Hasse diagram[4]of its channel type hierarchy on the right

represent the minimal elements of the upper sets that label edges, and we omit brackets when there is only one minimal element; on the other hand, we only represent one (directed) edge between any two nodes, and we label it with the upper sets that type the edges that exist between those nodes.

For example, the *btn*-labelled edge connecting the node A to $F0$ corresponds to the triple $\langle A, \{btn, ctr\}, F0\rangle$; and the edge from E to $F0$ corresponds to the triples $\langle E, \{obs\}, F0\rangle$ and $\langle E, \{ctr\}, F0\rangle$. Notice that, since every channel c can be identified by the triple $\langle \delta(c), \tau(c), \rho(c)\rangle$, channels are not named in figures: there is no risk of ambiguity.

In particular:

- The channel type *mov* captures the movement of one actor inside another. More specifically, the channels of type *mov* between *P0* and *P1* allow the cabin to move between the two platforms (up or down). Note that this is the intended semantics of *mov*; the actual semantics is given by means of axioms such as in Example 61. The same applies to the other channel types discussed below.
- As mentioned above, the type *door* is a subtype of *mov*; the channels of type *door* between *F0* and C, and between *F1* and C, allow users to enter or exit the cabin from or to the two floors.
- The channel type *obs* captures observations that an actor may make of another (their direction corresponds to that of the information flow); the channels of type *obs* that connect E to *F0* and to *F1* account for observations of the state of Elevator at either floor (imagine a typical display at each floor that is updated every time the cabin moves), while those that connect *F0* and *F1* to A account for the observations that Alice can make of either floor (by looking at the display).
- The channel type *ctr* captures the control that one actor may exert on another; the four channels of type *ctr* between *F0* and E, and between *F1* and E, are used for transmitting requests between the two floors and Elevator (so that the cabin is moved accordingly and the doors at any of the floors are opened once the cabin arrives there); and the channel that connects C to E is meant to be used for transmitting requests from Cabin to Elevator (imagine, for example, an electric current triggered by pressing the button).
- The channel subtype *btn* of *ctr* captures a special kind a control that is exerted through a button; there are three channels of type *btn*, connecting the actor A to *F0*, to *F1*, and to C; each of them accounts for a physical button that Alice can press at either floor or at Cabin.

$\mathcal{P}_{Elevator}$ — The ANt schema Elevator has two propositional symbols, C_at_P0 and C_at_P1, that are used to capture knowledge of the platform where Cabin is; this means that $\mathcal{P}_{\mathsf{Elevator}} = \{\mathsf{C\_at\_P0}, \mathsf{C\_at\_P1}\}$.

2.2. States

Protocols are executed by actors, as a result of which their state changes. A state of an ANt schema $\mathcal{A}$ consists of an ANt structure – its physical part – and a valuation of the propositional symbols – its data/knowledge part. There is

[4] As usual, for any two channel types κ and κ', $\kappa \leq \kappa'$ whenever the Hasse diagram contains an upward line segment from κ to κ'.

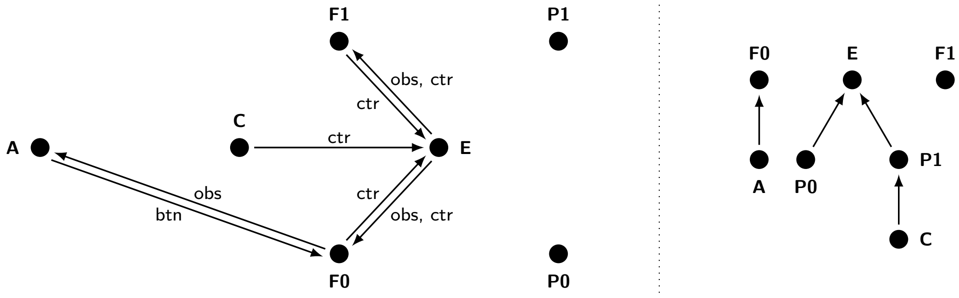

Figure 2: An ANt structure for Elevator: the graph on the left and the forest on the right

no built-in relationship between the structure and valuation of a state; instead, connections between the two parts that are relevant for specific protocols can be established through axioms as illustrated in Examples 16 and 54 for Elevator.

A structure for an ANt schema $\mathcal{A}$ consists of a subgraph of $\mathcal{G}_{\mathcal{A}}$ that captures *connectivity* (i.e., the subset of channels that connect the actors in that state) together with a forest – or *placement graph*, as in [12] – over its nodes that captures *locality* (i.e., where each actor is located in relation to the other actors in that state).

Definition 3 (Structure). A *structure* for a schema $\mathcal{A}$ is a pair $\langle \mathcal{H}, \mathcal{F} \rangle$ where:

- $\mathcal{H}$ is a subgraph of $\mathcal{G}_{\mathcal{A}}$, and
- $\mathcal{F}$ is a forest over $\mathcal{N}_{\mathcal{H}}$, meaning that every node n of $\mathcal{H}$ has either none or a unique parent (in $\mathcal{H}$), denoted $\mathcal{F}(n)$.

Example 4. An ANt structure for Elevator is depicted in Figure 2. The forest, on the right, places the two platforms inside Elevator, the Cabin inside the first-floor platform, and Alice at the ground floor; both floors are outside Elevator.

The graph, on the left, indicates the channels that are available (or active): the channel that corresponds to the button that Alice can press to call the elevator; the one that connects *F0* to Alice and allows her to observe the ground floor; the ones that connect *F0* and *F1* to *E*, and vice versa, so that requests can be transmitted to or by Elevator; the ones that connect *E* to *F0* and *F1* that allow the floors to observe Elevator; and the one that connects Cabin to Elevator and transmits calls to Elevator. Notice that, for readability, we always include the channel types in figures, even though they are not formally part of ANt structures.

To better visualise ANt structures (and similarly to [12]), we combine the graph and the forest by nesting nodes in the graph according the placement (location) hierarchy; this can be seen in Figure 3 for the ANt structure in Figure 2.

Definition 5 (Substructure). We say that $\langle \mathcal{H}_1, \mathcal{F}_1 \rangle$ is a *substructure* of, or is *included* in, $\langle \mathcal{H}_2, \mathcal{F}_2 \rangle$ if:

- $\mathcal{H}_1$ is a subgraph of $\mathcal{H}_2$, and
- for every node n of $\mathcal{H}_1$ such that the parent $\mathcal{F}_1(n)$ is defined, $\mathcal{F}_2(n)$ is also defined and equal to $\mathcal{F}_1(n)$ – in other words, the placement hierarchy is strictly preserved.

The notion of substructure defines a partial order on the set of structures for a schema $\mathcal{A}$, which we denote by $\preceq$.

A state is an ANt structure together with a *valuation* of the propositional symbols, which assigns to each node and propositional symbol the truth value of the propositional symbol at that node. For the models that we consider here, we work with a three-valued Łukasiewicz basis, which means that propositions may have the values $\oplus$ (*true*), $\ominus$ (*false*), or $\oplus$ (*undefined*). This allows us to capture states in which, for instance, Alice does not know where Cabin is.

Definition 6 (State). A *state* of an ANt schema $\mathcal{A}$ consists of a structure $\mathcal{S}$ for $\mathcal{A}$ such that $\mathcal{N}_{\mathcal{S}} = \mathcal{N}_{\mathcal{A}}$ (i.e., the structure has all the nodes of the schema) together with, for each node n, a *valuation* (map) $\mathcal{V}_n : \mathcal{P}_{\mathcal{A}} \to \{\ominus, \oplus, \oplus\}$.

Following the notational convention described in the introduction, we denote the structure underlying a state σ by $\mathcal{S}_\sigma$. In addition, we denote the set of all states of an ANt schema $\mathcal{A}$ by $\mathbb{S}_{\mathcal{A}}$.

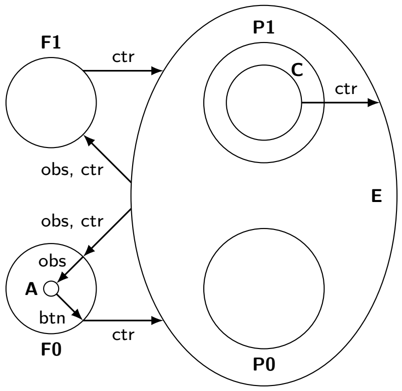

Figure 3: The underlying ᴀɴt structure of $\mathsf{elevator}_0$, a state of Elevator

Example 7. As an example, consider the state $\mathsf{elevator}_0$ whose underlying ᴀɴt structure is shown in Figure 3 and whose valuations are defined as follows:

- $\mathcal{V}_E(\mathsf{C\_at\_P0}) = \mathcal{V}_{F0}(\mathsf{C\_at\_P0}) = \mathcal{V}_{F1}(\mathsf{C\_at\_P0}) = \mathcal{V}_A(\mathsf{C\_at\_P0}) = \ominus$,
- $\mathcal{V}_E(\mathsf{C\_at\_P1}) = \mathcal{V}_{F0}(\mathsf{C\_at\_P1}) = \mathcal{V}_{F1}(\mathsf{C\_at\_P1}) = \mathcal{V}_A(\mathsf{C\_at\_P1}) = \oplus$, and
- $\mathcal{V}_n(\mathsf{C\_at\_P0}) = \mathcal{V}_n(\mathsf{C\_at\_P1}) = \circledcirc$ for all other nodes.

That is, *E*, *F0*, *F1* and *A* all know that Cabin is at *P1* (and not at *P0*); none of the other actors knows where Cabin is.

2.3. Interactions

When present in a state, specific configurations of connectivity and locality can lead to a state change. These specific configurations (parts of the states) are called *interactions*, each of which is defined in terms of an ᴀɴt structure.

Definition 8 (Interaction). An interaction in the context of an ᴀɴt schema $\mathcal{A}$ is just a structure for $\mathcal{A}$, which we usually denote by ι. The nodes of ι correspond to the actors taking part in the interaction; its channels capture the connectivity of the actors in that particular interaction; and the forest component of ι accounts for the relative placement of the actors under consideration. We denote by $\mathbb{I}_{\mathcal{A}}$ the set of all interactions of $\mathcal{A}$.

Note that interactions are defined *between actors* (in a given state), and are meant to signal or trigger a reconfiguration *between states*. Their semantics is given by the transition relation of an actor network (cf. Definition 10).

Example 9. Figure 4 depicts three interactions for the ᴀɴt schema Elevator:

a) callElevator0 involves the *btn* channel through which Alice can call the elevator at the ground floor (*F0*) and the *ctr* channel through which the request can then be transmitted to Elevator.

b) moveCabin0 involves the *mov* channel from the platform *P1* to *P0*, which enables the movement of the cabin between the two platforms, and the *ctr* channels between *F0* and *E* that mediate the opening of the doors.

c) enterCabin0 involves the channel of type *door* that connects *F0* to *C* and allows actors at the ground floor to enter the cabin, as well as the *obs* channels through which those actors can be informed that Cabin is at the ground floor (and is therefore safe to cross through).

When executed in a given state, an interaction produces another state, which we formalise in Section 2.4 below.

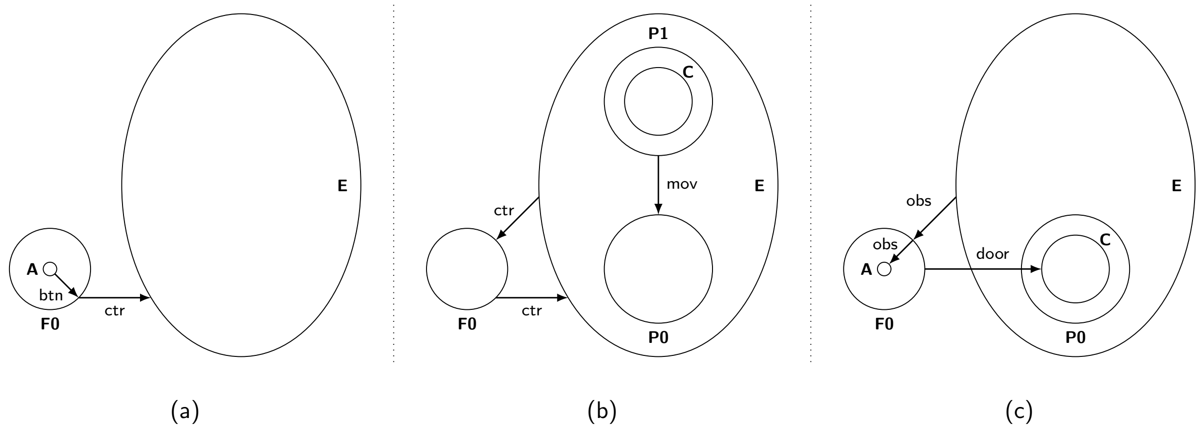

Figure 4: Interactions of Elevator: (a) callElevator0, (b) moveCabin0, and (c) enterCabin0

2.4. Actor networks

We formalise cyber-physical-system protocols as *actor networks*. For this purpose, an actor network consists of *(a)* an ANt schema, which contains all the actors and the channels available to connect them; *(b)* a set of possible worlds – each being associated with a state of the ANt schema under consideration – including a subset of designated initial worlds; *(c)* a set of possible interactions through which the actor network can evolve; and *(d)* for every such interaction, a transition relation on the set of worlds. More precisely:

Definition 10 (Actor network). An actor network $\mathfrak{N} = \langle \mathcal{A}, \mathcal{D}, \varsigma, \mathcal{D}_0, \mathcal{I}, \longrightarrow \rangle$ consists of:

- an ANt schema $\mathcal{A}$;
- a *domain* (finite set of *worlds*) $\mathcal{D}$ together with a function $\varsigma \colon \mathcal{D} \to \mathbb{S}_{\mathcal{A}}$ that assigns a state to every world;
- a non-empty subset $\mathcal{D}_0 \subseteq \mathcal{D}$ of *initial worlds* (whose labels are *initial states*);
- a (finite) set $\mathcal{I} \subseteq \mathbb{I}_{\mathcal{A}}$ of interactions of $\mathcal{A}$;[5]
- a *transition relation* $(\longrightarrow) \subseteq \mathcal{D} \times \mathcal{I} \times \mathcal{D}$ such that, for each interaction $\iota \in \mathcal{I}$ and world w, there is a transition $w \xrightarrow{\iota} w'$ to another world w' if and only if $\iota \leq \mathcal{S}_{\varsigma(w)}$ (i.e., interactions are substructures of the source states).

Therefore, an actor network over a schema $\mathcal{A}$ can be regarded as a labelled transition system over a set of states of $\mathcal{A}$, transitions being labelled with interactions. For each interaction, the ANt structure that defines it is required to be present in the source state of any transition performed by that interaction; moreover, its presence at any given state of the network determines that a transition by that interaction is available from the state.

Example 11. An actor network $\mathfrak{N}_{\mathsf{Elevator}}$ with Elevator as its schema could have, for example, $\mathsf{elevator}_0$ (labelled with the state defined in Example 7, which we also designate by $\mathsf{elevator}_0$, or by $\sigma_{\mathsf{elevator}_0}$ when there may be a risk of confusion) as its only initial world, and the worlds and transitions presented in Figure 5, among others. Note that the valuations are not included in these diagrams; an axiomatic presentation is discussed in Section 3.

The 'horizontal' transitions in Figure 5 are performed by the interaction callElevator0 – cf. Figure 4 (a). The one at the top starts at $\mathsf{elevator}_0$. Although several actors and channels are present in $\mathsf{elevator}_0$, the interaction callElevator0 indicates that the actors that are active in the transition are Alice, Elevator and *F0* (the ground floor), and that the active channels are those that connect *A* to *F0* and *F0* to *E*. That is to say, Alice presses the button at *F0* and the request is transmitted to Elevator. The transition to $\mathsf{elevator}_1$ activates the *mov* channel that connects *P1* to *P0* through which

[5] Notice that, since the schema is finite, its set of interactions is finite as well.

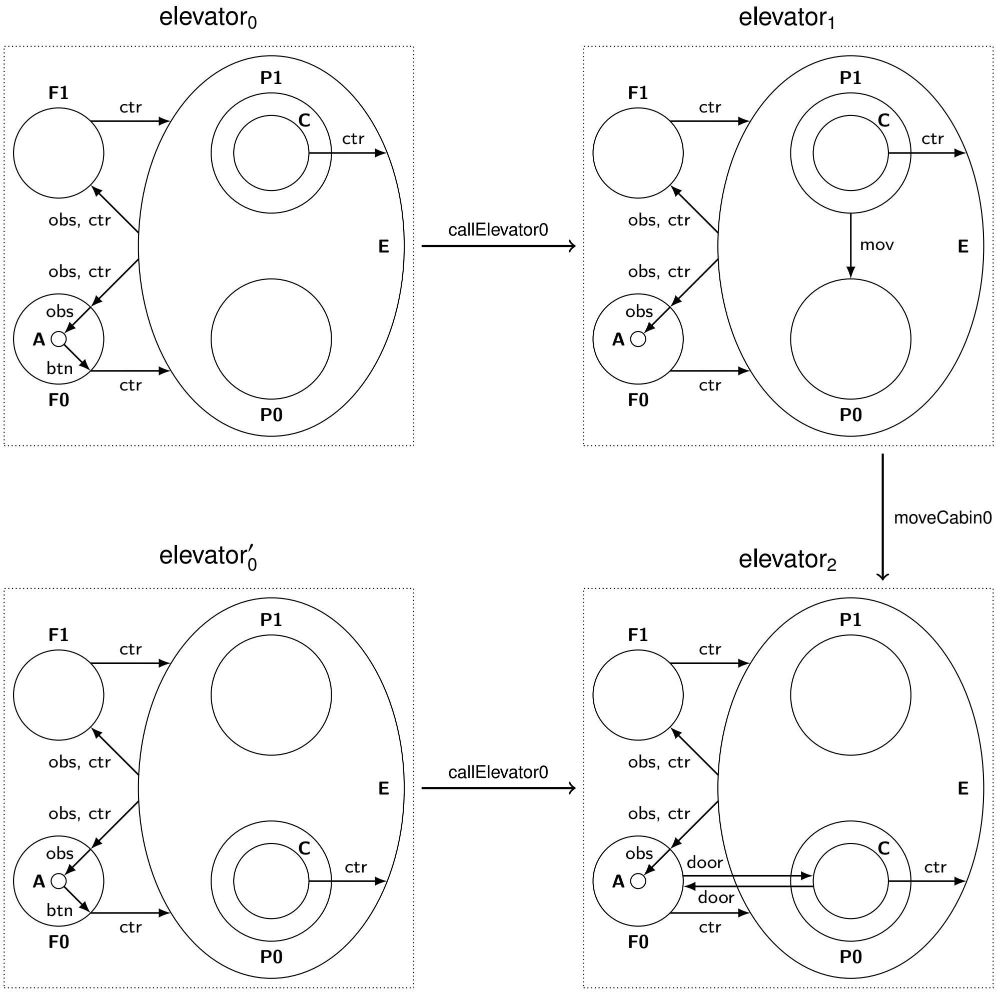

Figure 5: Transitions performed by the interactions callElevator0 and moveCabin0

Elevator can respond to the request (move the cabin), and closes the channel of type *btn* from *A* to *F0*, i.e., Alice is no longer able to call the elevator. (Naturally, one could model a scenario where the button can be repeatedly pressed, but that is not relevant for this protocol.) We denote both the resulting world and the state associated with it by $\mathsf{elevator}_1$.

The other transition performed by callElevator0 (at the bottom of the figure) starts in a different world, $\mathsf{elevator}'_0$, where Cabin is at the ground-floor platform, *P0*. It opens the two channels of type *door* between *F0* and *C* that allow users to enter or to exit the cabin. We denote the resulting world and state by $\mathsf{elevator}_2$.

The 'vertical' transition from $\mathsf{elevator}_1$ to $\mathsf{elevator}_2$ is performed by the interaction moveCabin0 – cf. Figure 4 (b). As indicated by the interaction, this computation is local to *P0* and *P1*. The transition moves the cabin from *P1* to *P0*, closes the channel of type *mov* that connects the two platforms and – just like the transition between $\mathsf{elevator}'_0$ and $\mathsf{elevator}_2$ – opens the two channels of type *door* that allow users to enter or to exit the cabin.

In the final part of the paper, when reasoning about properties of actor networks, we are particularly interested in those states that correspond to reachable worlds, i.e., to worlds that can be reached (through interactions) from initial worlds:

Definition 12 (Reachable states). Given an actor network $\mathfrak{N} = \langle \mathcal{A}, \mathcal{D}, \varsigma, \mathcal{D}_0, \mathcal{I}, \longrightarrow \rangle$, we say that:

- A world $w \in \mathcal{D}$ is *reachable under* $\mathfrak{N}$ if either $w \in \mathcal{D}_0$ (i.e., w is an initial world) or there exists a reachable world w' and an interaction $\iota \in \mathcal{I}$ such that $w' \xrightarrow{\iota} w$. We denote by $\mathcal{D}_R$ the set of reachable worlds of $\mathfrak{N}$.
- A state of $\mathcal{A}$ is *reachable under* $\mathfrak{N}$ if it is the image through ς of a reachable world under $\mathfrak{N}$. We denote by $\mathbb{R}_{\mathfrak{N}}$ the set of reachable states of $\mathfrak{N}$.
- $\mathfrak{N}$ *has reachable worlds/states* when all its worlds/states are reachable, i.e., when $\mathcal{D} = \mathcal{D}_R$, resp. $\varsigma(\mathcal{D}) = \mathbb{R}_{\mathfrak{N}}$.

3. Logics for ANts

The logics through which we can specify and reason about actor networks are obtained through an iterated process of constrained hybridisation. For every step, we discuss the syntax and the semantics of the logics involved, both of which are defined in terms of an appropriate notion of signature. As usual, signatures provide the primitive symbols used for building the sentences of the logic, which are then interpreted (assigned meaning, usually in an inductive fashion) in models; but for the hybridised logics that we consider here, signatures have a prevailing semantic role, as they also determine some of the constraints to be imposed on the models defined in the hybridisation process.

Throughout this section, we consider an arbitrary but fixed ANT schema $\mathcal{A} = \langle \mathcal{G}, \mathcal{T}, \tau, \mathcal{P} \rangle$.

3.1. The base logic

The base for the hybridised construction that we develop is the three-valued propositional Łukasiewicz logic, which we recall below. At this level, the underlying signature for the schema $\mathcal{A}$ is just the set $\mathcal{P}$ of its propositional symbols.

Definition 13 (Syntax). The set $\text{Ł}(\mathcal{P})$ of sentences of the base logic is the least set that contains all the elements of $\mathcal{P}$ (as *atomic sentences*) and is closed under *negation*, denoted $\overline{p}$, and *implication*, denoted $p \supset q$, where $p, q \in \text{Ł}(\mathcal{P})$.

Example 14. The Łukasiewicz-logic signature of Elevator is $\mathcal{P}_{\mathsf{Elevator}} = \{\mathsf{C\_at\_P0}, \mathsf{C\_at\_P1}\}$ as defined in Example 2.

Definition 15 (Semantics). The sentences of the base logic are interpreted over appropriate Łukasiewicz valuations, that is over functions $v \colon \text{Ł}(\mathcal{P}) \to \{\ominus, \text{\textcircled{$\pm$}}, \oplus\}$ that are compatible with the following two truth tables.

p	$\overline{p}$
$\ominus$	$\oplus$
$\text{\textcircled{$\pm$}}$	$\text{\textcircled{$\pm$}}$
$\oplus$	$\ominus$

$p \supset q$	$q = \ominus$	$q = \text{\textcircled{$\pm$}}$	$q = \oplus$
$p = \ominus$	$\oplus$	$\oplus$	$\oplus$
$p = \text{\textcircled{$\pm$}}$	$\text{\textcircled{$\pm$}}$	$\oplus$	$\oplus$
$p = \oplus$	$\ominus$	$\text{\textcircled{$\pm$}}$	$\oplus$

It is straightforward to see that any Łukasiewicz valuation is uniquely determined by its restriction to the propositional symbols of $\mathcal{P}$. In other words, there is a one-to-one correspondence between Łukasiewicz valuations and functions from $\mathcal{P}$ to $\{\ominus, \bigcirc\mkern-14mu\pm, \oplus\}$. We say that a proposition p is *valid*, or that it is a *tautology*, if $v(p) = \oplus$ for all valuations v.

Other useful standard connectives such as the disjunction (denoted $p \cup q$, for base-logic sentences p and q), conjunction (denoted $p \cap q$), and equivalence (denoted $p \equiv q$) can be defined through the following abbreviations:

- $p \cup q \triangleq (p \supset q) \supset q$
- $p \cap q \triangleq \overline{\overline{p} \cup \overline{q}}$
- $p \equiv q \triangleq (p \supset q) \cap (q \supset p)$

Their corresponding truth tables are:

$p \cup q$	$q = \ominus$	$q = \bigcirc\mkern-14mu\pm$	$q = \oplus$
$p = \ominus$	$\ominus$	$\bigcirc\mkern-14mu\pm$	$\oplus$
$p = \bigcirc\mkern-14mu\pm$	$\bigcirc\mkern-14mu\pm$	$\bigcirc\mkern-14mu\pm$	$\oplus$
$p = \oplus$	$\oplus$	$\oplus$	$\oplus$

$p \cap q$	$q = \ominus$	$q = \bigcirc\mkern-14mu\pm$	$q = \oplus$
$p = \ominus$	$\ominus$	$\ominus$	$\ominus$
$p = \bigcirc\mkern-14mu\pm$	$\ominus$	$\bigcirc\mkern-14mu\pm$	$\bigcirc\mkern-14mu\pm$
$p = \oplus$	$\ominus$	$\bigcirc\mkern-14mu\pm$	$\oplus$

$p \equiv q$	$q = \ominus$	$q = \bigcirc\mkern-14mu\pm$	$q = \oplus$
$p = \ominus$	$\oplus$	$\bigcirc\mkern-14mu\pm$	$\ominus$
$p = \bigcirc\mkern-14mu\pm$	$\bigcirc\mkern-14mu\pm$	$\oplus$	$\bigcirc\mkern-14mu\pm$
$p = \oplus$	$\ominus$	$\bigcirc\mkern-14mu\pm$	$\oplus$

Example 16. The base-logic sentence below captures the coherence of the actors' knowledge regarding the position of Cabin: if an actor knows that the cabin is at the ground platform, then it must also know that the cabin is not at the top platform, and vice versa; in addition, an actor that does not know whether the cabin is at the ground platform or not cannot know whether the cabin is at the top platform either.

- $\mathsf{C\_at\_P0} \equiv \overline{\mathsf{C\_at\_P1}}$

It is easy to see that, in the state $\mathsf{elevator}_0$ described in Example 7, the valuations associated with all the actors return the value $\oplus$ for this sentence.

The following Łukasiewicz modalities, which return Boolean values, are also useful:

- $\mathsf{M}\,p \triangleq \overline{p} \supset p$, meaning that p is *possibly true*, in the sense that it has value $\oplus$ or $\bigcirc\mkern-14mu\pm$;
- $\mathsf{L}\,p \triangleq \overline{\mathsf{M}\,\overline{p}}$, meaning that p is *necessarily true*, in the sense that it has value $\oplus$;
- $\mathsf{N}\,p \triangleq \overline{\mathsf{M}\,p}$ (or, equivalently, $\mathsf{L}\,\overline{p}$), meaning that p is *necessarily false*, in the sense that it has value $\ominus$;
- $\mathsf{I}\,p \triangleq \overline{\mathsf{M}\,p \supset \mathsf{L}\,p}$, meaning that p is *unknown*, in the sense that it has value $\bigcirc\mkern-14mu\pm$.

Their corresponding truth tables are:

p	$\mathsf{M}\,p$
$\ominus$	$\ominus$
$\bigcirc\mkern-14mu\pm$	$\oplus$
$\oplus$	$\oplus$

p	$\mathsf{L}\,p$
$\ominus$	$\ominus$
$\bigcirc\mkern-14mu\pm$	$\ominus$
$\oplus$	$\oplus$

p	$\mathsf{N}\,p$
$\ominus$	$\oplus$
$\bigcirc\mkern-14mu\pm$	$\ominus$
$\oplus$	$\ominus$

p	$\mathsf{I}\,p$
$\ominus$	$\ominus$
$\bigcirc\mkern-14mu\pm$	$\oplus$
$\oplus$	$\ominus$

Notice that the nature of these operators is indeed modal. This means that *(a)* for any two base-logic sentences p and q, the sentence $\mathsf{L}\,(p \supset q) \supset (\mathsf{L}\,p \supset \mathsf{L}\,q)$ is a tautology – this accounts for the standard modal axiom K, also known as the *distribution axiom*, for the operator L; *(b)* the operators M and L are *dual*, according to the very definition of L; and *(c)* the sentence $\mathsf{L}\,p$ is valid whenever p is valid – which corresponds to the *generalisation rule* for L.

3.2. The state logic

In order to capture the states of the ANt schema $\mathcal{A}$, we consider a constrained hybridisation of $Ł(\mathcal{P})$.[6] For that purpose, we start with a notion of state signature for $\mathcal{A}$. Technically, this is a countably infinite set $SVar$ whose elements are called *state variables* (following terminology from [24]); however, to emphasise the connection with the base Łukasiewicz logic and the role played by the actors and channels of $\mathcal{A}$ in the construction of state sentences, we also list $\mathcal{P}$, $\mathcal{N}$ and $\mathcal{T}$ as components of signatures.

Definition 17 (Signature). A *state signature* for $\mathcal{A}$ is a tuple

$$\Sigma = \langle \mathcal{P}, \mathcal{N}, SVar, \mathcal{T} \rangle$$

where $\mathcal{P}$ is the set of propositional symbols of $\mathcal{A}$ (i.e., a signature of the Łukasiewicz logic), $\mathcal{N}$ is the set of actor names of $\mathcal{A}$, $SVar$ is a countably infinite set disjoint from $\mathcal{P} \cup \mathcal{N}$, and $\mathcal{T}$ is the set of channel types of $\mathcal{A}$.

Example 18. The state signatures for the schema Elevator are of the form

$$\langle \mathcal{P}_{\mathsf{Elevator}}, \mathcal{N}_{\mathsf{Elevator}}, SVar, \mathcal{T}_{\mathsf{Elevator}} \rangle$$

where $(\mathcal{P}_{\mathsf{Elevator}} \cup \mathcal{N}_{\mathsf{Elevator}}) \cap SVar = \emptyset$ and, as defined in Examples 2 and 14,

- $\mathcal{P}_{\mathsf{Elevator}} = \{\mathsf{C\_at\_P0}, \mathsf{C\_at\_P1}\}$,
- $\mathcal{N}_{\mathsf{Elevator}} = \{F0, F1, E, C, P0, P1, A\}$,
- $\mathcal{T}_{\mathsf{Elevator}} = \{mov, door, obs, ctr, btn\}$.

The formulas and sentences of the state logic follow the standard definitions in the hybrid-logic literature. In this case, the set $\mathcal{P}$ determines the ordinary *propositional symbols*, $\mathcal{N} \cup SVar$ determines the *nominals* used to refer to individual nodes in a Kripke model, and $\mathcal{T}$ determines the *modalities* used in modal operators.

Definition 19 (Syntax). The formulas of the state logic for a signature $\Sigma = \langle \mathcal{P}, \mathcal{N}, SVar, \mathcal{T} \rangle$ are given by the grammar

$$\phi ::= p \mid a \mid \neg\, \phi \mid \phi \to \phi \mid \langle \kappa \rangle\, \phi \mid \langle \pi \rangle\, \phi \mid @_a\, \phi \mid \exists x\, \phi$$

where $p \in Ł(\mathcal{P})$, $a \in \mathcal{N} \cup SVar$, $x \in SVar$, $\kappa \in \mathcal{T}$, and π is a distinguished and new *parent* modality. Therefore, the state logic includes modal operators that capture connectivity – $\langle \kappa \rangle$, for $\kappa \in \mathcal{T}$ – and operators that capture locality – $\langle \pi \rangle$.

The free and the bound occurrences of nominals in formulas are defined as usual (e.g. [25, 26]) except that the only binder that we consider here is the existential quantifier ($\exists$). The Σ-sentences are formulas for Σ with no free occurrences of state variables; we denote their set by $\mathsf{State}(\Sigma)$.

Traditionally, hybrid-logic formulas are evaluated over unconstrained Kripke models, that is, over triples $\langle W, R, V \rangle$ where $\langle W, R \rangle$ is a Kripke frame – which means that W is a set of possible worlds and R is a family $R_\lambda \subseteq W \times W$ of accessibility relations indexed by modalities λ – and V is a W-indexed family of interpretations for nominals and propositional symbols. In order to enable nominals to act as names of possible worlds, every nominal is required to be satisfied (in the sense that its interpretation is true) at exactly one node of W (cf. [27]).

The semantics of specific hybrid logics often includes additional constraints; for example, in the S4 variant of hybrid propositional logic, the accessibility relations are required to be reflexive and transitive (i.e., they are preorders), while in the S5 variant they are required to be reflexive and Euclidean (i.e., they are equivalences). The constraints that we consider for the state logic follow from the underlying graph structure of the ANt schema:

- There is a one-to-one correspondence between actors and possible worlds. For notational convenience, we do not distinguish possible worlds from actors.

[6] Strictly speaking, not the set $Ł(\mathcal{P})$ is hybridised, but the Łukasiewicz logic to which it belongs. When there is no risk of confusion, we use such formulations as well to indicate the relationship between ANt schemas and the base-logic sentences used to describe their states: states of $\mathcal{A}$ are specified using hybrid-logic sentences built over $Ł(\mathcal{P})$. This extends in the latter part of the paper to ANt schemas and the networks defined over them.

- The accessibility relations conform to the channels and the channel types of the schema: for each channel type κ, R_κ consists only of pairs (n, n') of nodes that are connected through a channel of type κ.
- The interpretation of the parent (locality) modality π is functional and acyclic.

In other words, the constrained Kripke models that we consider here are states of the actor-network schema $\mathcal{A}$.

Definition 20 (Semantics: satisfaction). Let $\Sigma = \langle \mathcal{P}, \mathcal{N}, SVar, \mathcal{T} \rangle$ be a state signature for $\mathcal{A}$. Given a state $\sigma = \langle \mathcal{S}, \mathcal{V} \rangle$ of $\mathcal{A}$, an *assignment* is a map $\alpha \colon \mathcal{N} \cup SVar \to \mathcal{N}_{\mathcal{S}}$ whose restriction to the set of actor names is the identity.[7] The *satisfaction relation* between states and formulas is parameterised by assignments α and by actors n (i.e., nodes of $\mathcal{S}$):

- $\sigma, \alpha, n \vDash a$ iff $\alpha(a) = n$;
- $\sigma, \alpha, n \vDash p$ iff $v_n(p) = \oplus$, where v_n is the valuation defined by $\mathcal{V}_n$ over $\mathcal{P}$;
- $\sigma, \alpha, n \vDash \neg\, \phi$ iff $\sigma, \alpha, n \nvDash \phi$;
- $\sigma, \alpha, n \vDash \phi_1 \to \phi_2$ iff $\sigma, \alpha, n \vDash \phi_1$ implies $\sigma, \alpha, n \vDash \phi_2$;
- $\sigma, \alpha, n \vDash \langle \kappa \rangle\, \phi$ iff $\sigma, \alpha, \rho(c) \vDash \phi$ for some channel $c \in C$ such that $\delta(c) = n$ and $\kappa \in \tau(c)$;
- $\sigma, \alpha, n \vDash \langle \pi \rangle\, \phi$ iff $\mathcal{F}(n)$ is defined and $\sigma, \alpha, \mathcal{F}(n) \vDash \phi$;
- $\sigma, \alpha, n \vDash @_a\, \phi$ iff $\sigma, \alpha, \alpha(a) \vDash \phi$;
- $\sigma, \alpha, n \vDash \exists x\, \phi$ iff $\sigma, \alpha', n \vDash \phi$ for some assignment α' that agrees with α on $SVar \setminus \{x\}$.

Other propositional connectives such as the conjunction ($\wedge$), disjunction ($\vee$), and equivalence ($\leftrightarrow$) can be defined as usual. The dual modal operators ($[\kappa]$, where κ is a channel type, and $[\pi]$ for the parent modality) and the universal quantifier over state variables ($\forall$) can also be defined in the conventional way:

- $[\kappa]\, \phi \triangleq \neg\, \langle \kappa \rangle\, \neg\, \phi$

  That is, $\sigma, \alpha, n \vDash [\kappa]\, \phi$ iff $\sigma, \alpha, \rho(c) \vDash \phi$ for all channels $c \in C$ with $\delta(c) = n$ and $\kappa \in \tau(c)$.
- $[\pi]\, \phi \triangleq \neg\, \langle \pi \rangle\, \neg\, \phi$

  That is, $\sigma, \alpha, n \vDash [\pi]\, \phi$ iff $\sigma, \alpha, \mathcal{F}(n) \vDash \phi$ whenever $\mathcal{F}(n)$ is defined.
- $\forall x\, \phi \triangleq \neg\, \exists x\, \neg\, \phi$

  That is, $\sigma, \alpha, n \vDash \forall x\, \phi$ iff $\sigma, \alpha', n \vDash \phi$ for all assignments α' that agree with α on $SVar \setminus \{x\}$.

We also consider the *universal* modality, sometimes referred to as the *global* modality:

- $\mathsf{A}\, \phi \triangleq \forall x\, @_x\, \phi$, where x is a state variable that does not occur free in ϕ

  That is, $\sigma, \alpha, n \vDash \mathsf{A}\, \phi$ iff $\sigma, \alpha, m \vDash \phi$ for all nodes $m \in \mathcal{N}$. In other words, a sentence of the form $\mathsf{A}\, \phi$ is satisfied at a node when ϕ is satisfied at all the nodes of the state.

To keep the notation simple, we sometimes extend the use of the connectives and modal operators to sets of sentences. For example, when Φ is a finite set of sentences $\{\phi_1, \ldots, \phi_n\}$, we may denote by $\bigwedge \Phi$ the conjunction $\phi_1 \wedge \cdots \wedge \phi_n$. We also denote by $\mathsf{A}\, \Phi$ the set of sentences $\{\mathsf{A}\, \phi \mid \phi \in \Phi\}$, regardless of the cardinality of Φ.

Definition 21 (Semantics: validity). A state-logic formula is said to be *valid in a state* if it is satisfied, for every assignment, at every node of that state; it is *valid in an* ANt *structure* if it is satisfied in every state based on that structure; and it is *absolutely valid* (in a signature of the schema) if it is valid in every state of the schema. Formally,

- $\sigma \vDash \phi$ iff $\sigma, \alpha, n \vDash \phi$ for all assignments $\alpha \colon \mathcal{N} \cup SVar \to \mathcal{N}_\sigma$ (cf. Definition 20) and all $n \in \mathcal{N}_\sigma$;

[7] Recall that, by Definition 6, $\mathcal{N}_{\mathcal{S}} = \mathcal{N}$.

- $\mathcal{S} \vDash \phi$ iff $\sigma \vDash \phi$ for all states σ such that $\mathcal{S} \leq \mathcal{S}_\sigma$;[8]
- $\Sigma \vDash \phi$, or simply $\vDash \phi$, iff $\sigma \vDash \phi$ for all state-logic models σ of Σ, i.e., for all states $\sigma \in \mathbb{S}_{\mathcal{A}}$.[8]

These forms of validity extend to sets of state-logic formulas to mean that every formula in the set is valid. Given a set Φ of formulas, we denote by $\mathbb{S}_\Phi$ the set of states in which all the formulas in Φ are valid.

Notice that, because they have no free occurrences of state variables, the validity of sentences does not depend on the choice of the assignment: for every *sentence* ϕ, every actor n of a state σ, and for any two assignments α and α', $\sigma, \alpha, n \vDash \phi$ if and only if $\sigma, \alpha', n \vDash \phi$ – a corollary of Proposition 22 below. Therefore, we can disregard the assignments and say, for short, that a sentence is *valid* in a state when it is satisfied at every node of that state.

Proposition 22. *Let σ be a state of $\mathcal{A}$, and $\alpha_1, \alpha_2 \colon \mathcal{N} \cup SVar \to \mathcal{N}_\sigma$ two assignments for σ. For every state formula ϕ, if α_1 and α_2 agree on the free variables of ϕ, then for every actor $n \in \mathcal{N}_\sigma$ we have:*

$$\sigma, \alpha_1, n \vDash \phi \qquad \textit{if and only if} \qquad \sigma, \alpha_2, n \vDash \phi.$$

Proof. We prove the result by structural induction on ϕ. Most of the cases are straightforward, so we only consider here the two more interesting ones that correspond to satisfaction statements and to existentially quantified formulas.

For satisfaction statements, consider a formula $@_a\,\phi$ such that α_1 and α_2 agree on the free variables of ϕ and on a (assuming that a is a state variable; otherwise, if it is an actor name, the assignments agree on a by definition). Then

$$\begin{array}{lll} \sigma, \alpha_1, n \vDash @_a\,\phi & \text{iff} \quad \sigma, \alpha_1, \alpha_1(a) \vDash \phi & \text{according to Definition 20} \\ & \text{iff} \quad \sigma, \alpha_2, \alpha_1(a) \vDash \phi & \text{by the induction hypothesis} \\ & \text{iff} \quad \sigma, \alpha_2, \alpha_2(a) \vDash \phi & \text{because } \alpha_1 \text{ and } \alpha_2 \text{ agree on } a \text{ (regardless of its nature as an actor name or as a variable)} \\ & \text{iff} \quad \sigma, \alpha_2, n \vDash @_a\,\phi. & \text{according to Definition 20} \end{array}$$

For existentially quantified formulas, assume that α_1 and α_2 agree on all free variables of a state-logic formula ϕ, except an arbitrary but fixed variable $x \in SVar$. In that case, we obtain the following sequence of equivalences:

$$\begin{array}{ll} \sigma, \alpha_1, n \vDash \exists x\,\phi & \text{iff} \quad \sigma, \alpha_1', n \vDash \phi \text{ for some assignment } \alpha_1' \text{ that agrees with } \alpha_1 \text{ on } SVar \setminus \{x\} \\ & \qquad \text{according to Definition 20} \\ & \text{iff} \quad \sigma, \alpha_2', n \vDash \phi \text{ for some assignment } \alpha_2' \text{ that agrees with } \alpha_2 \text{ on } SVar \setminus \{x\} \\ & \qquad \text{by the induction hypothesis, because for every assignment } \alpha_1' \text{ as above} \\ & \qquad \text{there exists an assignment } \alpha_2'\text{, defined by } \alpha_2'(y) = \begin{cases} \alpha_1'(y) & \text{if } y \text{ occurs freely in } \phi \\ \alpha_2(y) & \text{otherwise,} \end{cases} \\ & \qquad \text{that agrees with } \alpha_1' \text{ on the free variables of } \phi\text{, and also with } \alpha_2 \text{ on } SVar \setminus \{x\} \\ & \text{iff} \quad \sigma, \alpha_2, n \vDash \exists x\,\phi. \qquad \text{according to Definition 20} \end{array}$$

□

Remark 23. A state-logic sentence ϕ is valid in a state if and only if $\mathsf{A}\,\phi$ is valid in that state.

Example 24. The following sentences are properties of (i.e., valid in) the state $\mathsf{elevator}_0$ defined in Example 7, of its underlying structure $\mathcal{S}_{\mathsf{elevator}_0}$ (according to Figures 2 and 3), or of the ANt schema Elevator itself:

- $\mathsf{elevator}_0 \vDash (E \vee F0 \vee F1 \vee A) \to \mathsf{L}\,(\mathsf{C.at.P1})$

  The actors E, $F0$, $F1$, and A know that Cabin is at the first-floor platform.

- $\mathcal{S}_{\mathsf{elevator}_0} \vDash @_A\,[btn]\,\langle ctr\rangle\,E$

  Whenever Alice calls the elevator, the request is transmitted to Elevator.

[8] Recall that, by Definition 5, $\mathcal{S} \leq \mathcal{S}_\sigma$ means that $\mathcal{S}$ is a substructure of the structure of σ; and that, by Definition 6, $\mathbb{S}_{\mathcal{A}}$ is the set of states of $\mathcal{A}$.

- $\mathsf{Elevator} \models @_{P0}\, [mov]\, P1 \wedge @_{P1}\, [mov]\, P0$[9]

  Any *mov* channel with source *P0* has target *P1* and any *mov* channel with source *P1* has target *P0*

Naturally, any property of the structure $\mathcal{S}_{\mathsf{elevator}_0}$ is also a property of $\mathsf{elevator}_0$, and any property of the schema $\mathsf{Elevator}$ is also a property of $\mathcal{S}_{\mathsf{elevator}_0}$ (and of any ANt structure of $\mathsf{Elevator}$). More generally:

Remark 25. Given a state signature Σ for $\mathcal{A}$, a structure $\mathcal{S}$, a state σ, and a sentence $\phi \in \mathsf{State}(\Sigma)$:

- If $\mathcal{S}_\sigma \models \phi$ then $\sigma \models \phi$.
- If $\Sigma \models \phi$ then $\mathcal{S} \models \phi$.

As might be expected, the converse implications do not necessarily hold. For example, considering Example 24, the first sentence is valid in $\mathsf{elevator}_0$ but not in its underlying structure, while the second sentence is valid in $\mathcal{S}_{\mathsf{elevator}_0}$ without being absolutely valid.

A useful property of the state logic is that it allows us to characterise the states that are admissible for a given structure by means of (sets of) state-logic sentences. The next result plays an important part in the following sections of the paper when dealing with the axioms of the ANt logic and with entailment.

Proposition 26. *Let $\Phi_\mathcal{S}$ be the (finite) set of all sentences of the form $@_a \langle\lambda\rangle\, b$ that are valid in a structure $\mathcal{S}$, where a and b are actor names and λ is a modality (i.e., a channel type or the parent modality π). Then $\mathbb{S}_{\Phi_\mathcal{S}} = \{\sigma \in \mathbb{S}_\mathcal{A} \mid \mathcal{S} \leq \mathcal{S}_\sigma\}$.*

Proof. Clearly, since $\Phi_\mathcal{S}$ consists only of sentences that are valid in $\mathcal{S}$, it follows that $\Phi_\mathcal{S}$ is valid in any state σ that has $\mathcal{S}$ as a substructure. For the converse, it suffices to notice, for any state σ of $\mathcal{A}$, that $\sigma \models @_a \langle\lambda\rangle\, b$ if and only if there exists a channel of type λ from a to b in $\mathcal{S}_\sigma$ (if λ is a channel type) or b is the parent of a in $\mathcal{S}_\sigma$ (if λ is the parent modality). Consequently, for all the states σ in which $\Phi_\mathcal{S}$ is valid, $\mathcal{S} \leq \mathcal{S}_\sigma$. □

The (absolute) validity of sentences can be established syntactically through a proof system, i.e., through a set of axioms and inference rules. As usual, by *theorem* of the logic we mean a sentence that is either an axiom, or can be derived (from axioms) in a finite number of steps through repeated applications of inference rules.

An example of a (Hilbert-style) proof system for the basic, unconstrained hybrid logic is given in Figure 6, which is both a simplification (because we do not make use of the binder ↓) and an extension (due to the multi-modality setting of our logic) of the axiom system given in [3, Chapter 2].[10] This system (among others, e.g., [24]) has been shown to be sound and complete in the sense that a sentence is a theorem if and only if it is valid in all Kripke frames.

Some useful theorems and admissible inference rules – in the sense that for every state-logic formula that can be derived using such a rule, there exists a derivation that does not make use of the rule (see, e.g., [3]) – include:

Theorems	
Taut	any propositional theorem
AndDistr	$@_a(\phi_1 \wedge \phi_2) \leftrightarrow (@_a \phi_1 \wedge @_a \phi_2)$
OrDistr	$@_a(\phi_1 \vee \phi_2) \leftrightarrow (@_a \phi_1 \vee @_a \phi_2)$

Admissible inference rules

$$\frac{\phi_1 \quad \phi_2}{\phi_1 \wedge \phi_2}\ (\wedge I) \qquad \frac{\phi_1}{\phi_1 \vee \phi_2}\ (\vee E) \qquad \frac{\phi}{[\lambda]\,\phi}\ (G_\lambda) \qquad \frac{\phi}{\mathsf{A}\,\phi}\ (G_\mathsf{A})$$

[9] As most unary operators, $[mov]$, $@_{P_0}$ and $@_{P_1}$ bind stronger than the binary operator $\wedge$, hence the sentence should be read as a conjunction.

[10] It should be noted that, although apparently finite, the axiomatisation in Figure 6 is actually infinite, since each of the axioms or rules (sometimes referred to in the literature as schemas) is parameterised by formulas, nominals, state variables, and other syntactic entities.

Axioms

CT	propositional theorems for $\neg$ and $\rightarrow$
Distr	$@_a(\phi_1 \rightarrow \phi_2) \leftrightarrow (@_a \phi_1 \rightarrow @_a \phi_2)$
SD	$@_a \phi \leftrightarrow \neg @_a \neg \phi$
Scope	$@_a @_b \phi \leftrightarrow @_b \phi$
Ref	$@_a a$
Intro	$(a \wedge \phi) \rightarrow @_a \phi$
λE*	$([\lambda]\phi \wedge \langle\lambda\rangle a) \rightarrow @_a \phi$
$\forall$E	$\forall x\, \phi \rightarrow \phi[a/x]$

Inference rules

$$\frac{\phi_1 \qquad \phi_1 \rightarrow \phi_2}{\phi_2}\ (\textsc{mp}) \qquad \frac{\phi}{@_a \phi}\ (@\textsc{i}) \qquad \frac{@_x \phi}{\phi}\ (@\textsc{e})^{\dagger}$$

$$\frac{(\phi_1 \wedge \langle\lambda\rangle x) \rightarrow @_x \phi_2}{\phi_1 \rightarrow [\lambda]\phi_2}\ (\lambda\textsc{i})^{\dagger} \qquad \frac{\phi_1 \rightarrow \phi_2[x/y]}{\phi_1 \rightarrow \forall y\, \phi_2}\ (\forall\textsc{i})^{\dagger}$$

∗ Here, λ stands both for the regular modalities defined by channel types and for π.

† x does not occur free in ϕ ($@$E), in ϕ_1 or ϕ_2 (λI), or in ϕ_1 or $\forall y\, \phi_2$ ($\forall$I).

Figure 6: Hilbert-style axioms and rules for basic hybrid logic

Remark 27. Because any ANt state has an underlying Kripke frame (corresponding to the structure of the state; see Definitions 6 and 3), and because the satisfaction of state-logic sentences (Definition 20) is consistent with the hybrid-logic concept of satisfaction of sentences, it follows that any sound and complete proof system for the basic hybrid logic is also sound for the state logic: sentences that can be proved using the axioms and inference rules in Figure 6 are valid in all Kripke frames and, therefore, they are absolutely valid in the sense of Definition 21.

Unsurprisingly, hybrid-logic proof systems such as the one in Figure 6 are no longer complete for the state logic in the sense that there are sentences of the state logic that are absolutely valid but cannot be derived as theorems. This is due to two major factors:

1. As a hybrid formalism, the state logic is built over a Łukasiewicz logic. Notice that, in order to syntactically mark the difference between the two logical levels, the symbols used for the negation ($\overline{\ }$) and implication ($\supset$) in the base Łukasiewicz logic are different from those used in the state logic ($\neg$ and $\rightarrow$, respectively). Their semantics is also different. For example, given two sentences $p, q \in \text{Ł}(\mathcal{P})$, we have that

   - $\vDash \overline{p} \rightarrow \neg p$, and
   - $\vDash (p \supset q) \rightarrow (p \rightarrow q)$,

   but not the other way around. These sentences, as well as any tautology of the base logic, cannot be derived from an axiom system (as in Figure 6) that does not take into account the structure of the base-logic sentences.

   We address this issue by following the general technique described in [7], where the authors show how (sound and complete) proof systems for the base logic can be integrated within a hybrid axiomatisation like the one described in Figure 6 to produce a (sound and complete) proof system for the resulting hybridised logic. All that we need to ensure in order to use this result – besides the existence of a sound and complete proof system for the base logic, which is known [28] – is that the three-valued Łukasiewicz logic that we consider here admits all semantic Boolean connectives[11] (cf. Figure 8), which is established in Proposition 29 below.

[11] Intuitively, this means that the logic is expressive enough to capture the semantics of the standard Boolean connectives, possibly through sentences that are considerably more complex than their Boolean counterparts – a property also known as Boolean completeness (see Proposition 29).

N1	$\bigvee\{a \mid a \in \mathcal{N}\}$
N2	$\neg\, @_a\, b$ for $a \neq b \in \mathcal{N}$
c1	$\neg\, @_a\, \langle\kappa\rangle\, b$ for $a, b \in \mathcal{N}$ such that there is no $c \in C$ with $\delta(c) = a$, $\rho(c) = b$, and $\kappa \in \tau(c)$
c2	$\langle\kappa\rangle\, a \to \langle\kappa'\rangle\, a$ for $\kappa \leq \kappa'$ and $a \in \mathcal{N}$
π1	$\langle\pi\rangle\, a \to [\pi]\, a$ for $a \in \mathcal{N}$
π2	$\neg\, @_a\, \langle\pi\rangle^n a$ for $1 \leq n \leq \lvert\mathcal{N}\rvert$ and $a \in \mathcal{N}$

Figure 7: Additional axioms for the state logic – $\mathrm{Ax}_{\mathsf{State}(\Sigma)}$

BLT	all base-logic theorems
BLI	$(p \to q) \leftrightarrow ((q \supset \overline{q}) \supset (p \supset \overline{p}))$
BLN	$\neg\, p \leftrightarrow (p \supset \overline{p})$

Figure 8: Additional axioms for the hybridised Łukasiewicz logic

2. As discussed above, the models of the state logic are not arbitrary (unconstrained) Kripke models but states of an ANt schema. Because of this, there are two main categories of new tautologies: those that arise from the ANt schema used, and those that are innate to the state logic. The former category includes, for example in the case of Elevator, sentences like $\neg\, @_A\, \langle ctr\rangle\, E$ (meaning that Alice cannot control the elevator directly) that derive from the underlying graph of the ANt schema over which the connectivity modalities are interpreted. The latter contains sentences such as $\neg\, @_a\, b$, where a and b are distinct actor names, and $\langle\pi\rangle\, \phi \to [\pi]\, \phi$, which derives from the forest structure over which the parent modality is interpreted.

   In order to regain completeness, for any given state signature Σ for the ANt schema $\mathcal{A}$, we consider the axioms presented in Figure 7, the set of which we denote by $\mathrm{Ax}_{\mathsf{State}(\Sigma)}$. The axioms *N1* and *N2* ensure that all possible worlds correspond to actor names, and that no two distinct names are interpreted in the same way. The axiom *c1* rules out those channels that are not defined in the ANt schema, while *c2* captures the channel subtyping relation. Lastly, *π1* and *π2* specify that the interpretations of the distinguished parent modality are functional and acyclic, respectively. Notice that, because ANt schemas are finite, so is $\mathrm{Ax}_{\mathsf{State}(\Sigma)}$.

Definition 28 (Proof system). The *proof system* $\mathrm{PS}_{\mathsf{State}(\Sigma)}$ for the state logic is the extension of the axiomatisation of the basic hybrid logic (as in Figure 6) with all the theorems of the Łukasiewicz logic and the statements that establish its semantic connectives (as in Figure 8), and with the axiomatic constraints $\mathrm{Ax}_{\mathsf{State}(\Sigma)}$ of the state logic (as in Figure 7).

Note that the axiomatisation in Figure 7 depends on the ANt schema under consideration. For instance, axiom *c1* depends on the connections between actors that are declared in the schema, while *c2* depends on the typing hierarchy. Therefore, the size and complexity of the axiomatisation is correlated with the level of detail of the ANt schema. This would not compromise potential implementation efforts though, because the schema is always finite and the number of state-logic axioms increases only quadratically with the number of actors in the ANt schema.

The completeness of $\mathrm{PS}_{\mathsf{State}(\Sigma)}$ relies on the following two elementary properties, each of which addresses one of the major factors mentioned above. The first one, Proposition 29, warrants the axioms in Figure 8; the second, Proposition 30 (which is, in fact, a direct consequence of the interpretation of the sentences in Figure 7 in Kripke models), establishes the necessary connection between the state logic and the hybridisation of the Łukasiewicz logic.

Proposition 29. *The three-valued propositional Łukasiewicz logic is Boolean complete, in the sense that, for any two sentences $p, q \in$ Ł$(\mathcal{P})$, there exist $piq, np \in$ Ł$(\mathcal{P})$ such that, for every valuation $v\colon$ Ł$(\mathcal{P}) \to \{\ominus, \textcircled{$\pm$}, \oplus\}$,*

- *$v(piq) = \oplus$ if and only if $v(p) = \oplus$ implies $v(q) = \oplus$;*
- *$v(np) = \oplus$ if and only if $v(p) \in \{\ominus, \textcircled{$\pm$}\}$.*

Proof. The sentence piq, which corresponds to the semantic Boolean implication of p and q, can be defined as $\mathsf{L}\, p \supset \mathsf{L}\, q$ or, equivalently, using only primitive connectives, as $(q \supset \overline{q}) \supset (p \supset \overline{p})$. For the semantic negation, we can define np as $\mathsf{N}\, p \cup \mathsf{I}\, p$ or, equivalently, as $p \supset \overline{p}$. To check the two properties above, it suffices to consider the following truth-tables:

$(q \supset \overline{q}) \supset (p \supset \overline{p})$	$q = \ominus$	$q = \textcircled{\pm}$	$q = \oplus$
$p = \ominus$	$\oplus$	$\oplus$	$\oplus$
$p = \textcircled{\pm}$	$\oplus$	$\oplus$	$\oplus$
$p = \oplus$	$\ominus$	$\ominus$	$\oplus$

p	$p \supset \overline{p}$
$\ominus$	$\oplus$
$\textcircled{\pm}$	$\oplus$
$\oplus$	$\ominus$

□

Proposition 30. *Every state signature Σ for $\mathcal{A}$ can be treated as a hybrid-logic signature. Under this view, the mapping of ANt states to their underlying Kripke models gives rise to a one-to-one correspondence (up to model isomorphism)*[12] *between the states of $\mathcal{A}$ and the (unconstrained) Kripke models over Σ that validate $Ax_{\mathsf{State}(\Sigma)}$. Moreover, for every state σ of $\mathcal{A}$ and every state sentence ϕ of Σ, ϕ is valid at σ if and only if it is valid in its underlying Kripke model.* □

Lemma 31. *Given an arbitrary but fixed state signature Σ for $\mathcal{A}$ and a sentence $\phi \in \mathsf{State}(\Sigma)$, if ϕ is absolutely valid then $\bigwedge \mathsf{A}\, Ax_{\mathsf{State}(\Sigma)} \to \phi$ is a theorem of the unconstrained hybridisation of the Łukasiewicz logic, i.e., it can be derived from the proof system in Figure 6 enriched with the axioms in Figure 8.*

Proof. Let ϕ be an absolutely valid state-logic sentence. It follows from Proposition 30 and Remark 23 that any model of the unconstrained hybridisation of the Łukasiewicz logic validates $\bigwedge \mathsf{A}\, \mathrm{Ax}_{\mathsf{State}(\Sigma)} \to \phi$. By the general soundness & completeness result of [7], since the base Łukasiewicz logic is Boolean complete (according to Proposition 29), we have that the axioms and proof rules in Figures 6 and 8 (which include all the theorems of the base logic – obtained, for example, by using a proof system as in [28]) are sound and complete for the unconstrained hybridisation of the Łukasiewicz logic. Consequently, the implication $\bigwedge \mathsf{A}\, \mathrm{Ax}_{\mathsf{State}(\Sigma)} \to \phi$ is a hybrid-logic theorem. □

Theorem 32. *A state-logic sentence (over a signature Σ) is absolutely valid if and only if it is derivable in $PS_{\mathsf{State}(\Sigma)}$.*

Proof. The soundness property follows trivially by induction on the structure of the derivation, so we focus only on completeness. Let ϕ be an absolutely valid state-logic sentence. By Lemma 31, this means that $\bigwedge \mathsf{A}\, \mathrm{Ax}_{\mathsf{State}(\Sigma)} \to \phi$ is a theorem of the unconstrained hybridisation of the Łukasiewicz logic. It follows that ϕ can be derived in $\mathrm{PS}_{\mathsf{State}(\Sigma)}$ by extending the derivation of the above-mentioned implication with multiple applications of the generalisation rule for the global modality to $\mathrm{Ax}_{\mathsf{State}(\Sigma)}$, of the introduction rule for the conjunction ($\wedge I$), and by modus ponens (MP). □

3.3. The ANt logic

The logic through which we can reason about the networks of an ANt schema $\mathcal{A}$ requires a further level of hybridisation. Concepts like those of signature, formula/sentence, model, and satisfaction are all defined as in the case of the state logic. The main difference is that, instead of the Łukasiewicz logic, the functionality assumed for the base logic is now provided by the state logic itself. There are no surprises on the syntactic side, where formulas are once again ordinary hybrid-logic sentences (built by regarding state-logic sentences as atoms). The semantics, however, deserves more attention because we are considering new constraints that are specific to the actor networks.

Definition 33 (Signature). An *ANt signature* for $\mathcal{A}$ is a tuple

$$\Omega = \langle \Sigma, Init, NVar, \mathcal{I} \rangle$$

consisting of a signature $\Sigma = \langle \mathcal{P}, \mathcal{N}, SVar, \mathcal{T} \rangle$ of the state logic, a (finite) set $Init$ whose elements are called *names of initial states*, a countably infinite set $NVar$ of *network variables* such that $\mathcal{P}$, $\mathcal{N}$, $SVar$, $Init$ and $NVar$ are pairwise disjoint, and a (finite) set $\mathcal{I}$ of interactions for $\mathcal{A}$.

[12] The correspondence is only up to isomorphism because the state-logic axioms ensure that the interpretation of actor names in Kripke models is bijective – it is not necessarily an identity mapping, as in the case of ANt states, where there is no distinction between actors and possible worlds.

Example 34. An ANT signature for Elevator that corresponds to the network discussed in Example 11 is of the form

$$\langle \Sigma, \mathit{Init}, \mathit{NVar}, \mathcal{I} \rangle$$

where $\Sigma = \langle \mathcal{P}_{\mathsf{Elevator}}, \mathcal{N}_{\mathsf{Elevator}}, \mathit{SVar}, \mathcal{T}_{\mathsf{Elevator}} \rangle$ is a state-logic signature as described in Example 18, *Init* is a singleton, and $\mathcal{I} \subseteq \mathbb{I}_{\mathsf{Elevator}}$ is a set of interactions defined over Elevator (as in Definition 8) that contains at least the three interactions given in Example 9: callElevator0, moveCabin0, and enterCabin0.

The sentences of the ANT logic (over a signature as described above) are defined as follows:

Definition 35 (Syntax). The syntax of the ANT logic for a signature $\Omega = \langle \Sigma, \mathit{Init}, \mathit{NVar}, \mathcal{I} \rangle$ is given by the grammar

$$\psi ::= \phi \mid i \mid \neg\psi \mid \psi \Rightarrow \psi \mid \langle\!\langle \iota \rangle\!\rangle \psi \mid i : \psi \mid \exists s\, \psi$$

where $\phi \in \mathsf{State}(\Sigma)$, $i \in \mathit{Init} \cup \mathit{NVar}$ and $s \in \mathit{NVar}$ are seen as ANT-logic *nominals*, and $\iota \in \mathcal{I}$ is an ANT-logic *modality*.

The free/bound occurrences of nominals in formulas and the notion of ANT-logic sentence are defined similarly to their state-logic counterparts. We denote by $\mathrm{ANT}(\Omega)$ the set of ANT-logic sentences defined over the signature Ω. For simplicity, we sometimes use $\mathsf{State}(\Omega)$ instead of $\mathsf{State}(\Sigma)$ to denote the set of sentences that belong to the base logic.

Notice that, in order to avoid ambiguity, we use double symbols for the connectives of the ANT logic, and that the satisfaction operators (represented as $@_a$ for the state logic) are denoted in this case by means of a colon. We extend the use of the double-symbol notation to other Boolean connectives ($\wedge$, $\vee$, $\Leftrightarrow$), to the dual modal operators ($[\![\_]\!]$), and to the universal quantifier ($\forall$), which are defined as in Section 3.2. In addition, we also consider the global modality $\mathbb{A}\,\psi \triangleq \forall s\,(s : \psi)$, where s is a network variable that has no free occurrences in ψ.

Example 36. We can now write sentences about the dynamics of Elevator like

$$@_C\, \langle \pi \rangle\, \mathit{P1} \Rightarrow [\![\mathsf{callElevator0}]\!]\; @_{\mathit{P1}}\, \langle \mathit{mov} \rangle\, \mathit{P0}$$

meaning that, if the interaction callElevator0 takes place when the cabin is at the first floor, then in the resulting state there will be an active channel of type *mov* between the two platforms – which could then enable, through the execution of different interactions, the actual movement of the cabin from *P1* to *P0*.

Similarly to the level of the state logic, the semantics of the ANT logic is defined once more by means of constrained Kripke models. In this case, the models of a signature $\Omega = \langle \Sigma, \mathit{Init}, \mathit{NVar}, \mathcal{I} \rangle$ for the schema $\mathcal{A}$ are actor networks $\mathfrak{N} = \langle \mathcal{A}, \mathcal{D}, \varsigma, \mathcal{D}_0, \mathcal{I}, \longrightarrow \rangle$ as in Definition 10 together with an onto mapping from the set *Init* (of names of initial states of Ω) to the set $\mathcal{D}_0$ (of initial worlds of $\mathfrak{N}$); to keep the notation simple, when there is no danger of confusion we denote by $\mathfrak{N}_i$ the initial world associated to a name $i \in \mathit{Init}$. The ANT-logic constraints are significantly simpler: all that we need to ensure is that the interactions $\iota \in \mathcal{I}$ are enabled precisely at those states for which ι is a substructure.

Definition 37 (Semantics: satisfaction). Consider an ANT signature $\Omega = \langle \Sigma, \mathit{Init}, \mathit{NVar}, \mathcal{I} \rangle$ for $\mathcal{A}$, an actor-network model $\mathfrak{N} = \langle \mathcal{A}, \mathcal{D}, \varsigma, \mathcal{D}_0, \mathcal{I}, \longrightarrow \rangle$ for Ω, and an assignment (function) $\alpha \colon \mathit{Init} \cup \mathit{NVar} \to \mathcal{D}$ such that $\alpha(i) = \mathfrak{N}_i$ for every $i \in \mathit{Init}$. For every world $w \in \mathcal{D}$, the *satisfaction* of ANT sentences at w is inductively defined as follows:

- $\mathfrak{N}, \alpha, w \Vdash i$ iff $\alpha(i) = w$;
- $\mathfrak{N}, \alpha, w \Vdash \phi$ iff $\varsigma_{\mathfrak{N}}(w) \vDash \phi$;
- $\mathfrak{N}, \alpha, w \Vdash \neg\psi$ iff $\mathfrak{N}, \alpha, w \nVdash \psi$;
- $\mathfrak{N}, \alpha, w \Vdash \psi_1 \Rightarrow \psi_2$ iff $\mathfrak{N}, \alpha, w \Vdash \psi_1$ implies $\mathfrak{N}, \alpha, w \Vdash \psi_2$;
- $\mathfrak{N}, \alpha, w \Vdash \langle\!\langle \iota \rangle\!\rangle \psi$ iff there is a transition $w \xrightarrow{\iota} w'$ in $\mathfrak{N}$ such that $\mathfrak{N}, \alpha, w' \Vdash \psi$;
- $\mathfrak{N}, \alpha, w \Vdash i : \psi$ iff $\mathfrak{N}, \alpha, \alpha(i) \Vdash \psi$;
- $\mathfrak{N}, \alpha, w \Vdash \exists s\, \psi$ iff $\mathfrak{N}, \alpha', w \Vdash \psi$ for some assignment α' that agrees with α on $\mathit{NVar} \setminus \{s\}$.

SLT	all state-logic theorems
SLI	$(\phi_1 \Rightarrow \phi_2) \Leftrightarrow (\mathsf{A}\,\phi_1 \rightarrow \phi_2)$
SLN	$\neg\,\phi \Leftrightarrow \exists x\ @_x \neg\,\phi$

Figure 9: Additional axioms for the hybridisation of the state logic

Inter[13]	$\bigwedge \Phi_\iota \Leftrightarrow \langle\!\langle \iota \rangle\!\rangle\, \mathsf{true}$, where $\iota \in \mathcal{I}$

Figure 10: Additional axioms for the ANt logic – $\mathrm{Ax}_{\mathsf{ANt}(\Omega)}$

Definition 38 (Semantics: validity). Similarly to the first level of hybridisation, we say that an ANt-logic sentence ψ defined over $\Omega = \langle \Sigma, \mathit{Init}, \mathit{NVar}, \mathcal{I} \rangle$ is *valid in an actor network (model)* if it is satisfied, for every assignment, at every world of the network, and that a sentence ψ is *absolutely valid* if it is valid in every actor-network model:

- $\mathfrak{N} \Vdash \psi$ iff $\mathfrak{N}, \alpha, w \Vdash \psi$ for all assignments $\alpha \colon \mathit{Init} \cup \mathit{NVar} \rightarrow \mathcal{D}_{\mathfrak{N}}$ and all $w \in \mathcal{D}_{\mathfrak{N}}$;
- $\Omega \Vdash \psi$, or simply $\Vdash \psi$, iff $\mathfrak{N} \Vdash \psi$ for all ANt models $\mathfrak{N}$ over Ω.

Given a set Ψ of ANt sentences, we denote by $\mathbb{N}_\Psi$ the set of actor networks over which all the sentences in Ψ are valid.

Example 39. Let Ω be an ANt signature for Elevator as described in Example 34, and let $\mathfrak{N}_{\mathsf{Elevator}}$ be the ANt model that corresponds to the actor network outlined in Example 11 – where the only initial-state name of Ω is interpreted as $\sigma_{\mathsf{elevator}_0}$ (the only initial world of $\mathfrak{N}_{\mathsf{Elevator}}$). Then we obtain the following validity statements:

- $\mathfrak{N}_{\mathsf{Elevator}} \Vdash [\![\mathsf{moveCabin0}]\!]\ @_C \langle \pi \rangle\, \mathit{P0}$

  Any execution of the interaction moveCabin0 results in a state where Cabin is at the ground floor.
- $\Omega \Vdash \langle\!\langle \mathsf{moveCabin0} \rangle\!\rangle\, \mathsf{true} \Rightarrow @_{\mathit{P1}} \langle \mathit{mov} \rangle\, \mathit{P0}$

  Moreover, the moveCabin0 transitions are enabled only at states where there is a *mov* channel from *P1* to *P0*.

Notice that, unlike the second sentence above, the first one cannot be inferred from the general properties of the ANt models over the signature Ω, and thus it is valid in $\mathfrak{N}_{\mathsf{Elevator}}$ without being absolutely valid.

Having the development of the proof system for the state logic as a reference, we can easily define a similar proof system for the ANt logic, which would be adequate for establishing (syntactically) the absolute validity of ANt-logic sentences. To that end, as argued in Section 3.2, we need to ensure that:

1. The base logic – in this case, the state logic – admits all semantic Boolean connectives.
2. The constraints of the ANt logic can be captured through a (finite) set of hybrid-logic sentences.

In order to address these concerns, we consider a different extension of the axiomatisation of hybrid logic.

Definition 40 (Proof system). The *proof system* $\mathrm{PS}_{\mathsf{ANt}(\Omega)}$ for the ANt logic is the extension of the axiomatisation of the basic hybrid logic (as in Figure 6, but using the connectives, hybrid-logic operators and quantifiers of the ANt logic) with all the theorems of the state logic and the statements that establish its semantic connectives (as in Figure 9), and with the axiomatic constraints $\mathrm{Ax}_{\mathsf{ANt}(\Omega)}$ of the ANt logic (as in Figure 10).

The adequacy of $\mathrm{PS}_{\mathsf{ANt}(\Omega)}$ follows from the next two properties.

Proposition 41. *The state logic is Boolean complete: for any state signature Σ for $\mathcal{A}$ and sentences $\phi, \phi_1, \phi_2 \in \mathsf{State}(\Sigma)$, there exist $\upsilon, \mu \in \mathsf{State}(\Sigma)$ such that, for every ANt state σ for $\mathcal{A}$,*

[13] Recall that, by Proposition 26, Φ_ι is set of sentences of the form $@_a \langle \lambda \rangle\, b$ that are valid in ι.

- *$\sigma \vDash \upsilon$ if and only if $\sigma \vDash \phi_1$ implies $\sigma \vDash \phi_2$;*
- *$\sigma \vDash \mu$ if and only if $\sigma \nvDash \phi$.*

Proof. Let υ and μ be the sentences $\mathsf{A}\,\phi_1 \to \phi_2$ and $\exists x\ @_x \neg\phi$, respectively (cf. Figure 9).

By Definition 20 we have that, for every state σ for $\mathcal{A}$, $\mathsf{A}\,\phi_1 \to \phi_2$ is valid in σ if and only if σ satisfies ϕ_2 at every node when it satisfies ϕ_1 at every node, i.e., ϕ_2 is valid in σ if ϕ_1 is valid in σ. Notice that the global modality is needed because the semantics of the plain state-logic implication $\phi_1 \to \phi_2$ is significantly different; for example, taking a sentence of the form $a \to p$ in the role of ϕ_1 and the sentence $@_a\, p$ in the role of ϕ_2, it is easy to see that any state that validates $a \to p$ also validates $@_a\, p$ but $(a \to p) \to @_a\, p$ is not a tautology.

For the negation, $\exists x\ @_x \neg\phi$ is valid in a state σ if and only if there is a node n of σ where $\neg\phi$ is satisfied; that is, if and only if ϕ is not valid in σ. Notice that, since the nodes of σ are actors, for the semantic negation of ϕ we could also consider the (finite) disjunction of all sentences of the form $@_a \neg\phi$, where $a \in \mathcal{N}$. □

Proposition 42. *The* ANt *logic is equivalent to a fragment of the hybridisation of the state logic, as follows: every* ANt *signature Ω for $\mathcal{A}$ is also a signature of the hybridisation of the state logic, and every* ANt *model (of a signature Ω) uniquely corresponds to a Kripke model that validates the sentences $Ax_{\mathsf{ANt}(\Omega)}$ in Figure 10.* □

Theorem 43. *An* ANt*-logic sentence (over a signature Ω) is absolutely valid if and only if it is derivable in $PS_{\mathsf{ANt}(\Omega)}$.*

Proof. Analogous to the proof of Theorem 32, by Propositions 41 and 42. □

4. Specifying and Reasoning about ANts

In this section, we show how the logics defined previously can be used for specifying and reasoning about actor networks. Throughout the section, we consider a fixed ANt schema $\mathcal{A} = \langle \mathcal{G}, \mathcal{T}, \tau, \mathcal{P} \rangle$ and use Elevator as an example.

4.1. Entailment

The methodological approach that we consider for reasoning about ANts is that of the algebraic-specification tradition for the formal development of systems. This means that we consider two essential, and interrelated, processes:

Specification where classes of models of interest (ANt states or networks) are described by means of sets of sentences.

Verification where we determine (e.g., through a proof system) that a specification guarantees certain properties.

The following notions of semantic entailment are essential in formalizing the verification process.

Definition 44 (Entailment). Given a set Φ of sentences and a sentence ϕ, all over the same state signature, we say that

- ϕ is a *(global) semantic consequence* of Φ, or that Φ *(globally) entails* ϕ, and we write $\Phi \vDash \phi$, when ϕ is valid in all the states where the sentences of Φ are valid – in other words, when $\mathbb{S}_\Phi \subseteq \mathbb{S}_{\{\phi\}}$.
- ϕ is a *local semantic consequence* of Φ, or that Φ *locally entails* ϕ, when ϕ is satisfied at all the nodes (of all states) where the sentences of Φ are satisfied.[14]

ANt-logic counterparts of the notions of global and local semantic consequence can be defined analogously; in that case, we denote the fact that ψ is a global semantic consequence of a set Ψ of ANt sentences by $\Psi \Vdash \psi$. In addition,

- ψ is a *semantic consequence of* Ψ *with respect to reachable states*, or Ψ *entails* ψ *on reachable states*, when ψ is valid in all networks with reachable states (according to Definition 12) where the sentences of Ψ are valid – more formally, if $\mathfrak{N} \Vdash \psi$ for all networks $\mathfrak{N}$ whose worlds/states are all reachable and such that $\mathfrak{N} \Vdash \Psi$.[15]

[14] Equivalently, one could say that ϕ is valid in all so-called *pointed* Kripke models where Φ is valid (see, e.g. [29]).

[15] This kind of entailment is reminiscent of the *master modality* from [27, 30].

Remark 45. Any local consequence of a set Φ of state sentences (or, similarly, of ANt sentences) is also a global consequence of Φ. The converse does not necessarily hold. As a (counter)example, it would be enough to consider the singleton $\{p\}$ in the role of Φ, where p is an atomic sentence, and $[\kappa]\, p$ in the role of ϕ.

Besides absolute validity, the proof systems $\mathrm{PS}_{\mathsf{State}(\Sigma)}$ and $\mathrm{PS}_{\mathsf{ANt}(\Omega)}$ discussed in Section 3 can also be used to syntactically establish the entailment of state- and ANt-logic sentences.

Definition 46 (Syntactic entailment). Under the notations and assumptions of Definition 44, we say that the sentence ϕ is *provable* from Φ, and we write $\Phi \vdash \phi$, if ϕ can be derived in $\mathrm{PS}_{\mathsf{State}(\Sigma)}$ (i.e., ϕ is a state theorem), or there are sentences $\phi_1, \dots, \phi_n \in \Phi$ such that the implication $\mathsf{A}\,(\phi_1 \wedge \cdots \wedge \phi_n) \to \phi$ is derivable in $\mathrm{PS}_{\mathsf{State}(\Sigma)}$.

Correspondingly, an ANt-logic sentence ψ is *provable* from Ψ, denoted $\Psi \Vdash \psi$, if ψ can be derived in $\mathrm{PS}_{\mathsf{ANt}(\Omega)}$, or there are sentences $\psi_1, \dots, \psi_n \in \Psi$ such that $\mathbb{A}\,(\psi_1 \mathbin{\mathbb{\wedge}} \cdots \mathbin{\mathbb{\wedge}} \psi_n) \Rightarrow \psi$ is derivable in $\mathrm{PS}_{\mathsf{ANt}(\Omega)}$.

The completeness results that we consider in this section are based on the following elementary properties of the global and local semantic-entailment relations, which are well known in the hybrid-logic literature (see, e.g. [27]).

Proposition 47.

1. *A sentence ϕ is a global semantic consequence of Φ if and only if it is a local semantic consequence of $\mathsf{A}\,\Phi$.*
2. *The local semantic-entailment relation is* compact*, in the sense that, if Φ locally entails ϕ, then there exists a finite subset $\Phi_0 \subseteq \Phi$ such that Φ_0 locally entails ϕ.*
3. *The local semantic-entailment relation has the* deduction theorem*, i.e., Φ locally entails $\phi_1 \to \phi_2$ if and only if $\Phi \cup \{\phi_1\}$ locally entails ϕ_2. In particular, $\phi_1 \to \phi_2$ is a tautology if and only if $\{\phi_1\}$ locally entails ϕ_2.* □

Theorem 48. *A state-logic sentence ϕ is a (global) semantic consequence of Φ if and only if it is provable from Φ.*

$$\Phi \models \phi \qquad \textit{iff} \qquad \Phi \vdash \phi$$

Proof. Similarly to the proof of Theorem 32, we only consider the completeness of the state-logic provability relation. Notice, however, that in this case the converses of all of the following implications are valid as well.

Suppose that ϕ is a semantic consequence of Φ. Then, by Proposition 47, ϕ is a local semantic consequence of $\mathsf{A}\,\Phi$. Since the local semantic-entailment relation is compact (also by Proposition 47), it follows that there exists a finite subset $\Phi_0 \subseteq \Phi$ such that ϕ is a local semantic consequence of $\mathsf{A}\,\Phi_0$. Therefore, by the deduction theorem, we have that the sentence $\bigwedge \mathsf{A}\,\Phi_0 \to \phi$ is absolutely valid. Finally, Theorem 32 ensures that $\bigwedge \mathsf{A}\,\Phi_0 \to \phi$ is derivable in $\mathrm{PS}_{\mathsf{State}(\Sigma)}$; in other words, given that A and $\bigwedge$ commute, and according to Definition 46, ϕ is provable from $\Phi_0 \subseteq \Phi$. □

In concrete situations, the following corollary allows us to write formal proofs that are much shorter than those that would result from the direct application of Definition 46. A similar result can be obtained for the ANt logic based on Theorem 50 below, but to avoid repetition, we do not state it explicitly.

Corollary 49. *A state-logic sentence ϕ is a (global) semantic consequence of Φ if and only if*

1. *it is a state-logic axiom,*
2. *it is a sentence in Φ, or*
3. *it can be derived (based on 1 and 2) through repeated applications of inference rules.*

Proof. The soundness part follows trivially from the soundness of the inference rules of the state logic. For completeness, notice that, by Theorem 48, we know there exists a finite subset $\{\phi_1, \dots, \phi_n\} \subseteq \Phi$ such that $\mathsf{A}\,(\phi_1 \wedge \cdots \wedge \phi_n) \to \phi$ is a theorem. Therefore, ϕ can be derived from $\{\phi_1, \dots, \phi_n\}$ through further applications of the introduction rule for the conjunction ($\wedge I$), of the generalisation rule for the global modality (G_{A}), and of modus ponens (MP). □

In an analogous manner to Theorem 48, for the ANt logic we have:

Theorem 50. *An* ANt*-logic sentence ψ is a (global) semantic consequence of Ψ if and only if it is provable from Ψ.*

$$\Psi \Vdash\!\!\models \psi \qquad \textit{iff} \qquad \Psi \Vdash\!\!- \psi \qquad \Box$$

Properties of networks that have reachable states can be proved through an induction schema.

Corollary 51. *Let Ψ and ψ be (sets of)* ANt *logic sentences over a signature Ω as in Definition 44 such that*

1. *$\Psi \Vdash\!\!- i : \psi$, for all names of initial states i of Ω;*
2. *$\Psi \Vdash\!\!- \psi \Rightarrow [\![\iota]\!]\, \psi$, for all interactions ι of Ω.*

Then ψ is a global semantic consequence of Ψ with respect to reachable states.

Proof. Suppose $\mathfrak{N}$ is an ANt model for Ω that has reachable states and validates Ψ. By Theorem 50, we know that, for all names of initial states i and interactions ι of Ω, both $i : \psi$ and $\psi \Rightarrow [\![\iota]\!]\, \psi$ are global consequences of Ψ. Consequently, $i : \psi$ and $\psi \Rightarrow [\![\iota]\!]\, \psi$ are valid at $\mathfrak{N}$, from which we deduce that *(a)* ψ is satisfied at all the initial states of $\mathfrak{N}$, and *(b)* if ψ is satisfied at a state σ of $\mathfrak{N}$, then it also satisfied at all successor states of σ, for all interaction ι of Ω. It follows that ψ is satisfied at all (reachable) states of $\mathfrak{N}$ or, equivalently, that ψ is valid in $\mathfrak{N}$. □

We are particularly interested in applying the induction schema described above to state properties, in which case the induction step is of the form

$$\phi \Rightarrow [\![\iota]\!]\, \phi$$

where ϕ is a state-logic sentence and ι is an interaction of the ANt-logic signature Ω. Whenever that is the case, we say that ϕ is an *invariant for ι with respect to* Ψ; moreover, when this property holds for all interactions ι of Ω, and when Ψ can be easily inferred from the context, we simply say that ϕ is an *invariant*.

More generally, notice that sentences of the form $\phi_1 \Rightarrow [\![\iota]\!]\, \phi_2$ correspond to Hoare triples [31], and express properties of the transitions performed by interactions: intuitively, the sentence ϕ_1 is a (pre)condition under which the execution of the interaction ι ensures the (post)condition ϕ_2. This connection with the Hoare logic goes back to Vaughan Pratt's work on propositional dynamic logic in the late 1970s [32, 33].

4.2. State specifications

As mentioned above, we consider a specification to be a set of sentences over a given signature of a logic.

Definition 52 (State specification). A *state specification* for $\mathcal{A}$ consists of a signature Σ for $\mathcal{A}$ (i.e., with the same actor names, channel types, and propositional symbols as the schema) and a set of sentences in $\mathsf{State}(\Sigma)$.

We denote by Φ_{Elevator} the state specification for the schema Elevator. This set contains, for instance, the base-logic sentence $\mathsf{C\_at\_P0} \equiv \overline{\mathsf{C\_at\_P1}}$ from Example 16, and the sentence $@_A\, [btn]\, \langle ctr \rangle\, E$ from Example 24. Other examples of sentences that we would expect to find in Φ_{Elevator} are given throughout this section.

Example 53 (Observation channels). We can include in Φ_{Elevator} sentences that give a meaning to *obs*-channels:

E1 $p \rightarrow [obs]\, p$ for every $p \in Ł(\mathcal{P})$

Intuitively, this means that knowledge is propagated through observation channels.

Example 54 (Knowledge). We can also specify what knowledge should be available at specific nodes, establishing in this way a link between the base and the state level of the specification. For example:

E2 $(@_C \langle \pi \rangle\, P0 \leftrightarrow @_E\, \mathsf{L}\,(\mathsf{C\_at\_P0})) \wedge (@_C \langle \pi \rangle\, P1 \leftrightarrow @_E\, \mathsf{L}\,(\mathsf{C\_at\_P1}))$

The elevator proper always knows at which platform the cabin is; notice that this sentence alone has no impact on the knowledge at other nodes of the whereabouts of the cabin, which might vary.

E3 $@_a\, p \to \bigvee_{b \in \mathcal{N}\setminus\{a\}} @_b\,(p \wedge \langle obs\rangle\, a)$
for $a \in \mathcal{N} \setminus \{E\}$ and $p \in \{\mathsf{L}\,(\mathsf{C\_at\_P0}), \mathsf{N}\,(\mathsf{C\_at\_P0}), \mathsf{L}\,(\mathsf{C\_at\_P1}), \mathsf{N}\,(\mathsf{C\_at\_P1})\}$

Knowledge about the location of the cabin can be acquired by actors other than E only through observation channels.

That is, these sentences of Φ_{Elevator} determine what knowledge nodes have about Cabin: the elevator proper always knows where the cabin is, and the other nodes can only acquire that information through observation channels.

Through entailment, state-logic specifications allow us to determine properties of the ANt structures and schemas in which the sentences of the specification are valid. The following corollary of Proposition 26 provides a characterisation of the validity of a sentence at a structure in terms of global (semantic) entailment.

Corollary 55. *Given an* ANt *structure $\mathcal{S}$ for $\mathcal{A}$ and a sentence ϕ, $\mathcal{S} \models \phi$ if and only if $\Phi_{\mathcal{S}} \models \phi$.* □

Example 56 (callElevator0). It is trivial to see that $\Phi_{\mathsf{callElevator0}}$ consists of:

c1 $@_A\,\langle\pi\rangle\, F0$

Alice is at the ground floor.

c2 $@_A\,\langle btn\rangle\, F0$

A *btn* channel connects Alice to the ground floor.

c3 $@_A\,\langle ctr\rangle\, F0$

A *ctr* channel connects Alice to the ground floor. Notice that this sentence is a consequence of c2, since $btn \leq ctr$.

c4 $@_{F0}\,\langle ctr\rangle\, E$

A *ctr* channel connects the ground floor to Elevator.

Example 57 ($\mathcal{S}_{\mathsf{elevator}_0}$). It is trivial to see that the following sentences belong to $\Phi_{\mathcal{S}_{\mathsf{elevator}_0}}$:

s1 $@_C\,\langle\pi\rangle\, P1$

The cabin is at the first platform.

s2 $@_E\,\langle obs\rangle\, F0$

An *obs* channel connects the elevator to the ground floor.

s3 $@_{F0}\,\langle obs\rangle\, A$

An *obs* channel connects the ground foor to Alice.

Given that callElevator0 is a substructure of $\mathcal{S}_{\mathsf{elevator}_0}$, we also have that $\Phi_{\mathcal{S}_{\mathsf{elevator}_0}}$ includes $\Phi_{\mathsf{callElevator0}}$.

One can use Corollary 55 to infer properties of the underlying structures of given states. For example, from the specification $\Phi_{\mathsf{Elevator}} \cup \Phi_{\mathcal{S}_{\mathsf{elevator}_0}}$ we can infer, by syntactic entailment, properties like $@_A\, \mathsf{L}\,(\mathsf{C\_at\_P1})$ – i.e., at $\sigma_{\mathsf{elevator}_0}$, Alice knows that the cabin is at the first platform. For simplicity, we tend to write $\Phi \vdash_{\mathsf{Elevator}} \phi$ instead of $\Phi_{\mathsf{Elevator}} \cup \Phi \vdash \phi$.

Let us first consider the following intermediary property:

Example 58 (Observation channels). The schema:

E4 $@_a\,(p \wedge \langle obs\rangle\, b) \to @_b\, p$

can be derived from Φ_{Elevator}, i.e., for any two actors $a, b \in \mathcal{N}_{\mathsf{Elevator}}$ and any base-logic sentence $p \in \text{Ł}(\mathcal{P})$,

$$\Phi_{\mathsf{Elevator}} \vdash @_a\,(p \wedge \langle obs\rangle\, b) \to @_b\, p.$$

The schema states that, if actors a and b are connected through an observation channel, then b acquires all the knowledge that is available at a. A possible derivation (according to Corollary 49) is as follows:

1	$@_a\,(p \to [obs]\,p)$	by $@_I$, based on E1
2	$@_a\,((p \wedge \langle obs\rangle\, b) \to ([obs]\,p \wedge \langle obs\rangle\, b))$	propositional equivalence, based on 1
3	$@_a\,((p \wedge \langle obs\rangle\, b) \to @_b\,p)$	by propositional calculus, based on 2 and λE
4	$@_a\,(p \wedge \langle obs\rangle\, b) \to @_a\,@_b\,p$	by *Distr*, from 3
5	$@_a\,(p \wedge \langle obs\rangle\, b) \to @_b\,p$	by *Scope*, from 4

Example 59. We can now use E4 to derive:

$$\{@_C\,\langle\pi\rangle\,P1,\ @_E\,\langle obs\rangle\,F0,\ @_{F0}\,\langle obs\rangle\,A\} \vdash_{\mathsf{Elevator}} @_A\,\mathsf{L}\,(\mathsf{C\_at\_P1})$$

That is, if Cabin is at the first platform, the ground floor can observe Elevator and, in turn, Alice can observe the ground floor, then Alice knows that Cabin is at the first platform.

1	$@_C\,\langle\pi\rangle\,P1$	by hypothesis (s1)
2	$@_C\,\langle\pi\rangle\,P1 \to @_E\,\mathsf{L}\,(\mathsf{C\_at\_P1})$	from E2
3	$@_E\,\mathsf{L}\,(\mathsf{C\_at\_P1})$	by MP, based on 1 and 2
4	$@_E\,\langle obs\rangle\,F0$	by hypothesis (s2)
5	$@_E\,\mathsf{L}\,(\mathsf{C\_at\_P1}) \wedge @_E\,\langle obs\rangle\,F0$	by $\wedge I$, based on 3 and 4
6	$@_E\,(\mathsf{L}\,(\mathsf{C\_at\_P1}) \wedge \langle obs\rangle\,F0)$	from *AndDistr*, based on 5
7	$@_E\,(\mathsf{L}\,(\mathsf{C\_at\_P1}) \wedge \langle obs\rangle\,F0) \to @_{F0}\,\mathsf{L}\,(\mathsf{C\_at\_P1})$	from E4
8	$@_{F0}\,\mathsf{L}\,(\mathsf{C\_at\_P1})$	by MP, based on 6 and 7
9	$@_{F0}\,\langle obs\rangle\,A$	by hypothesis (s3)
10	$@_{F0}\,\mathsf{L}\,(\mathsf{C\_at\_P1}) \wedge @_{F0}\,\langle obs\rangle\,A$	by $\wedge I$, based on 8 and 9
11	$@_{F0}\,(\mathsf{L}\,(\mathsf{C\_at\_P1}) \wedge \langle obs\rangle\,A)$	from *AndDistr*, based on 10
12	$@_{F0}\,(\mathsf{L}\,(\mathsf{C\_at\_P1}) \wedge \langle obs\rangle\,A) \to @_A\,\mathsf{L}\,(\mathsf{C\_at\_P1})$	from E4
13	$@_A\,\mathsf{L}\,(\mathsf{C\_at\_P1})$	by MP, based on 11 and 12

There are other general properties of Elevator that we might want to prove; for example, $@_C\,\langle\pi\rangle\,(P0 \vee P1)$ (from Example 65) – the cabin is (always) either at *P0* or at *P1*. Because such properties are not structural, in the sense that they do not hold at every state of Elevator, they should be proved instead at the level of actor networks, which define the way states can evolve through repeated interactions. For that purpose, we need to work with ANt specifications.

4.3. ANt specifications

Similar to state specifications, we define:

Definition 60 (ANt specification). An *ANt specification* for $\mathcal{A}$ consists of a signature Ω for $\mathcal{A}$ and a set of sentences in ANt(Ω). Given an ANt specification Ψ, we define $\mathsf{State}(\Psi) = \{\phi \in \mathsf{State}(\Omega) \mid \Psi \Vdash \phi\}$, i.e., the sentences of the state logic that are entailed by Ψ, which we call the *underlying state specification* of Ψ.

This means that, if Ψ is an ANt specification for $\mathcal{A}$, then for every actor network $\mathfrak{N}$ over $\mathcal{A}$ that validates Ψ ($\mathfrak{N} \Vdash \Psi$), we have $\varsigma_{\mathfrak{N}}(\mathcal{D}_{\mathfrak{N}}) \subseteq \mathbb{S}_{\mathsf{State}(\Psi)}$, i.e., all the states of $\mathfrak{N}$ satisfy the state properties entailed by Ψ; in particular, $\mathbb{R}_{\mathfrak{N}} \subseteq \mathbb{S}_{\mathsf{State}(\Psi)}$.[16]

The ANt logic allows us to axiomatise specific types of channels that influence state transitions, much in the same way that in Example 53 we axiomatised observation channels:

Example 61 (Move channels). Consider an ANt signature $\Omega_{\mathsf{Elevator}}$ as in Example 34. One could axiomatise *mov* channels as follows. For every interaction $\iota \in \mathcal{I}_{\mathsf{Elevator}}$ and actors $a, o, t \in \mathcal{N}_{\mathsf{Elevator}}$:

T1 $@_a\,\langle\pi\rangle\,o \Rightarrow [\![\iota]\!]\,@_a\,\langle\pi\rangle\,t$ if $\iota \vDash @_o\,\langle mov\rangle\,t$

[16] Recall that, by Definition 12, $\mathbb{R}_{\mathfrak{N}}$ is the set of reachable states of $\mathfrak{N}$.

T2 $@_a \langle \pi \rangle\, o \Rightarrow [\![\iota]\!]\, @_a \langle \pi \rangle\, o \quad \text{if } \iota \not\models @_o \langle mov \rangle\, \mathsf{true}$

This means that any interaction that involves a channel of type *mov* between the actors o (for *origin*) and t (for *target*), regarded as locations, determines the movement to t of any actor located in o; on the other hand, if an interaction does not involve a *mov* channel starting at o, then the actors in o maintain their location. For example, because $\mathsf{moveCabin0} \models @_{P1} \langle mov \rangle\, P0$ (according to Proposition 26 and Corollary 55) and $\mathsf{moveCabin0} \not\models @_{F0} \langle mov \rangle\, \mathsf{true}$ (easy to prove, given that moveCabin0 has no *mov* channel starting at *F0*), we obtain

$$@_C \langle \pi \rangle\, P1 \Rightarrow [\![\mathsf{moveCabin0}]\!]\, @_C \langle \pi \rangle\, P0$$
$$@_A \langle \pi \rangle\, F0 \Rightarrow [\![\mathsf{moveCabin0}]\!]\, @_A \langle \pi \rangle\, F0$$

That is, a transition performed by moveCabin0 on a state where Cabin is at the first platform moves Cabin to the ground platform and, if Alice is at the ground floor, she will remain there.

Example 62 (Control channels). Whereas *mov* channels are responsible for changes operated by interactions on the placement graph of a state, channels of type *ctr* are responsible for changes operated on the sub-graph, i.e., they control which channels can be made available as a result of the interaction.

One could axiomatise *ctr* channels as follows. For every $\iota \in \mathcal{I}_{\mathsf{Elevator}}$, $\kappa \in \mathcal{T}_{\mathsf{Elevator}}$ and $a, b \in \mathcal{N}_{\mathsf{Elevator}}$:

T3 $\langle\!|\iota|\!\rangle\, @_a \langle \kappa \rangle\, b \Rightarrow @_a \langle \kappa \rangle\, b \quad \text{if } \iota \models \forall x\, (@_a \langle \pi^* \rangle\, x \to \neg\, \exists y\, @_y \langle ctr^+ \rangle\, x)$

where π^* denotes the reflexive and transitive closure of the parent modality, and ctr^+ the transitive closure of the *ctr*-channel modality. Notice that, because the ANt schema is finite, both closures can be defined using disjunctions:

- $\langle \pi^* \rangle\, \phi \;\triangleq\; \phi \vee \langle \pi \rangle\, \phi \vee \langle \pi \rangle^2 \phi \vee \cdots \vee \langle \pi \rangle^{n-1} \phi$
- $\langle ctr^+ \rangle\, \phi \;\triangleq\; \langle ctr \rangle\, \phi \vee \langle ctr \rangle^2 \phi \vee \cdots \vee \langle ctr \rangle^n \phi$

where $n = |\mathcal{N}_{\mathsf{Elevator}}|$. In practice, however, the disjunction-based approach to reflexive and transitive closures may prove to be inadequate for theorem-proving purposes. This suggests further research into connections between hybrid logic (and, in particular, the state logic) and the much more expressive dynamic logic (see, for example, [34]).

The intuitive interpretation of the axiom T3 is that, if an interaction makes available a channel κ between two nodes a and b when applied to a given state, then the interaction must include a *ctr* channel (or a path of such channels) whose target is either a or an ancestor of a. This means that there must be an actor (corresponding to y) that exerts control over either a or an ancestor of a (which implicitly controls its descendants). This captures part of the notion of agency that is intrinsic to actor networks.

Example 63 (callElevator0 *&* moveCabin0). One can also specify the effects of specific interactions:

T4 $@_C \langle \pi \rangle\, P1 \Rightarrow [\![\mathsf{callElevator0}]\!]\, @_{P1} \langle mov \rangle\, P0$

When the elevator is called (at the ground floor), a request to move the cabin to the ground platform is issued if the cabin is at the first-floor platform.

T5 $[\![\mathsf{moveCabin0}]\!]\, @_{F0} \langle door \rangle\, (C \wedge \langle door \rangle\, F0)$

The doors are opened whenever the cabin moves to the (ground) platform.

We now have all the necessary ingredients for illustrating the proofs of invariants and of inductive properties.

Example 64 (Invariants). Let Ψ_{Elevator} be the extension of the specification Φ_{Elevator} from Section 4.2 with the sentences discussed in Examples 61, 62 and 63, and with all sentences of the form ($\mathsf{elevator0} : \phi$) where elevator0 is the only initial-state name of the ANt signature of Elevator and ϕ is a state sentence valid in $\sigma_{\mathsf{elevator}_0}$ (which includes those presented in Example 57). Then, the sentence $@_C \langle \pi \rangle\, (P0 \vee P1)$ is an invariant with respect to the specification Ψ_{Elevator} in the sense that Ψ_{Elevator} entails $@_C \langle \pi \rangle\, (P0 \vee P1) \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle\, (P0 \vee P1)$ for all interactions $\iota \in \mathcal{I}_{\mathsf{Elevator}}$.

We start by proving that $\Psi_{\mathsf{Elevator}} \Vdash @_C \langle \pi \rangle\, P0 \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle\, (P0 \vee P1)$.

1. From T2, if $\iota \nvDash @_{P0} \langle mov \rangle \mathsf{true}$ then $@_C \langle \pi \rangle P0 \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle P0$ is a sentence in Ψ_{Elevator}; therefore $\Psi_{\mathsf{Elevator}} \Vdash @_C \langle \pi \rangle P0 \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle (P0 \vee P1)$.
2. Suppose now that $\iota \vDash @_{P0} \langle mov \rangle \mathsf{true}$.
   (a) From Example 24, $\iota \vDash @_{P0} [mov]\, P1$, and thus $\iota \vDash @_{P0} \langle mov \rangle P1$.
   (b) From T1, it follows that $@_C \langle \pi \rangle P0 \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle P1$ is a sentence in Ψ_{Elevator}.
   (c) Therefore, $\Psi_{\mathsf{Elevator}} \Vdash @_C \langle \pi \rangle P0 \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle (P0 \vee P1)$.

Following the same reasoning, $\Psi_{\mathsf{Elevator}} \Vdash @_C \langle \pi \rangle P1 \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle (P0 \vee P1)$. Hence we conclude that

$$\Psi_{\mathsf{Elevator}} \Vdash @_C \langle \pi \rangle (P0 \vee P1) \Rightarrow [\![\iota]\!]\, @_C \langle \pi \rangle (P0 \vee P1)$$

Example 65. We can take the sentence from Example 64 further and show that it is not only an invariant, but a global semantic consequence of Ψ_{Elevator} with respect to reachable states. Building on Example 57 (S1), we know that the sentence ($\mathsf{elevator0} : @_C \langle \pi \rangle P1$) is part of the specification Ψ_{Elevator}. Therefore, the following entailment holds:

$$\Psi_{\mathsf{Elevator}} \Vdash \mathsf{elevator0} : @_C \langle \pi \rangle (P0 \vee P1)$$

By the induction schema of Corollary 51, it follows that Ψ_{Elevator} entails $@_C \langle \pi \rangle (P0 \vee P1)$ on reachable states. That is, the fact that Cabin is either at the first platform or at the ground platform is a property of all the reachable states of the actor networks that validate Ψ_{Elevator} – and, in particular, of the actor network described in Example 11.

The proofs of invariants rely essentially on Hoare triples, i.e., sentences of the form $\phi_1 \Rightarrow [\![\iota]\!]\, \phi_2$. Another kind of property concerns the availability of transitions performed by interactions when certain conditions hold, which can be done through sentences of the form $\phi \Rightarrow \langle\!|\iota|\!\rangle \mathsf{true}$. These are called *progress* properties, in the sense that they ensure that computations can proceed. Examples of such sentences are available through the axiom schema *Inter* in Figure 10.

Example 66 (Progress). For instance, from Example 56 we can conclude, based on *Inter*, that

$$\Psi_{\mathsf{Elevator}} \Vdash (@_A \langle \pi \rangle F0 \wedge @_A \langle btn \rangle F0 \wedge @_{F0} \langle ctr \rangle E) \Rightarrow \langle\!|\mathsf{callElevator0}|\!\rangle \mathsf{true}$$

Because, from Example 56, the sentences in the antecedent are also valid for $\mathcal{S}_{\mathsf{elevator}_0}$, we can also conclude that

$$\Psi_{\mathsf{Elevator}} \Vdash \mathsf{elevator0} : \langle\!|\mathsf{callElevator0}|\!\rangle \mathsf{true}$$

which means that progress can be made from the initial state by performing callElevator0.

This example also illustrates how the axiomatisation of *ctr* channels can interfere with progress properties, possibly generating inconsistencies. Therefore, some care must be taken when defining Ψ_{Elevator}. For example, by T4, we have

$$\Psi_{\mathsf{Elevator}} \Vdash @_C \langle \pi \rangle P1 \Rightarrow [\![\mathsf{callElevator0}]\!]\, @_{P1} \langle mov \rangle P0$$

Because, as seen in Example 64, ($\mathsf{elevator0} : @_C \langle \pi \rangle P1$) belongs to Ψ_{Elevator} we can conclude that

$$\Psi_{\mathsf{Elevator}} \Vdash \mathsf{elevator0} : [\![\mathsf{callElevator0}]\!]\, @_{P1} \langle mov \rangle P0$$

Using now the progress property above, we obtain

$$\Psi_{\mathsf{Elevator}} \Vdash \mathsf{elevator0} : \langle\!|\mathsf{callElevator0}|\!\rangle\, @_{P1} \langle mov \rangle P0$$

In order to see how interference could arise, notice that the sentence $@_{P1} [mov]\, \mathsf{false}$ is valid in the state $\sigma_{\mathsf{elevator}_0}$ discussed in Examples 64 and 11. Therefore, ($\mathsf{elevator0} : @_{P1} [mov]\, \mathsf{false}$) belongs to Ψ_{Elevator} as well.

On the other hand, by T3 it follows that, if $\mathsf{callElevator0} \vDash \forall x\, (@_{P1} \langle \pi^* \rangle x \rightarrow \neg\, \exists y\, @_y \langle ctr^+ \rangle x)$, then

$$\Psi_{\mathsf{Elevator}} \Vdash \langle\!|\mathsf{callElevator0}|\!\rangle\, @_{P1} \langle mov \rangle P0 \Rightarrow @_{P1} \langle mov \rangle P0$$

which would raise a contradiction:

$$\mathsf{elevator0} : @_{\mathit{P1}}\,[\mathit{mov}]\,\mathsf{false} \qquad \text{and} \qquad \mathsf{elevator0} : @_{\mathit{P1}}\,\langle \mathit{mov}\rangle\, \mathit{P0}$$

However, a contradiction does not arise here because, using again the axiom schema *Inter* but in the other direction, we can derive from Example 56 that $\langle\!|\mathsf{callElevator0}|\!\rangle\,\mathsf{true} \Rightarrow (@_A\,\langle \mathit{ctr}\rangle\, \mathit{F0} \wedge @_{\mathit{F0}}\,\langle \mathit{ctr}\rangle\, E)$ is an ANt-logic tautology, which shows that $\forall x\,(@_{\mathit{P1}}\,\langle \pi^{*}\rangle\, x \rightarrow \neg\,\exists y\; @_y\,\langle \mathit{ctr}^{+}\rangle\, x)$ is not valid for $\mathsf{callElevator0}$ – there is a chain of *ctr* channels that connects Alice to $\mathsf{Elevator}$ – the parent of *P1* – which allows for a new channel to be created connecting *P1* to *P0*.

That is, when modelling actor networks we need to ensure that appropriate control channels are present whenever the interactions create new connections.

5. Concluding remarks

In this paper, we have shown how, through a two-stage constrained-hybridisation process, a suite of logics can be developed that support the specification and verification of cyber-physical-system protocols modelled as actor networks (ANts) in the sense of [10]. The first stage of the hybridisation process results in a logic that captures the structure of actor networks and the way knowledge flows across such networks; the second addresses the dynamic aspects of actor networks, i.e., the way they can evolve as a result of the interactions that occur within them.

One of the main novel aspects of the paper in relation to the state of the art in hybridisation is in the fact that we rely on unconventional semantic constraints derived from the structural characteristics of ANt states or from the general properties of state transitions. This results in faithful representations, at a logical level, of the way actor networks perform computations. That is, constrained models capture the relationship between the higher-level reconfigurations of networks and the lower-level interactions between actors that trigger them. Besides expressivity, a key property of these constraints is that they can be axiomatised within hybrid logic. This enables the use of conventional (sound and complete) proof systems for hybrid logic as a tool through which we can formally verify properties of actor networks.

The semantic model of ANts is a bigraph-like structure in the sense of [12], i.e., it captures both connectivity (actors can be connected to other actors, and these connections can change in time) and locality (actors have a location and can move in space). This is achieved in the first-stage hybridisation (which produces the state logic – Section 3.2) through two different kinds of modalities: $\langle \kappa \rangle$, where κ is a channel type, for connectivity; and $\langle \pi \rangle$ for locality. Through these modalities, we can model the way information (such as data or knowledge) is transmitted among actors based on the connectivity that is available between them and where they are located – which is not intrinsic to bigraphs. Although we used a rather simple model of knowledge based on a three-valued propositional Łukasiewicz logic, the first-stage hybridisation could have been applied to a more expressive logic if relevant for a given application domain.

The second level of hybridisation (which produces the ANt logic – Section 3.3) allows us to reason about the dynamics of ANts, i.e., the way connectivity and locality change when certain configurations, called interactions, are present in a state. This sort of reasoning is akin to the use of dynamic logic for reasoning about imperative programs (e.g., [35]) or of temporal logic for reasoning about concurrent computations (e.g., [21]); as a proof of concept, we gave an example of a typical proof of an invariant. We also showed how certain types of channels that are intended to capture movement (i.e., changes to locality) or control (i.e., changes to connectivity) can be axiomatised in the ANt logic, which leads us to conclude that the two-stage constrained-hybridisation process that we proposed is particularly well-suited for dealing with both the structural and the dynamic aspects of the semantic domains that capture connectivity and locality. Part of our planned future work is to use the expressive power of this formalism to reason about security protocols in cyber-physical systems, and in particular about non-interference and the existence of covert channels, which is part of the original motivation for the study of ANts as proposed in [10].

This approach to modelling and analysis is declarative in the sense that we proposed logics whose sentences express, declaratively, properties of a system, and inference rules that can be used to infer properties from a system specification. This is different from the logics that have been proposed for bigraphs (e.g., [19], [20]), which are term-based process calculi through which one can give an operational semantics based on term rewriting. One of the research directions that we are currently pursuing is precisely towards an operational semantics of ANts based on graph transformation systems in the sense of [36], and its integration with the declarative approach proposed herein.

ANts are suitable for the kinds of closed protocols that arise in many classes of cyber-physical systems where security is a concern; an example of such systems are the authentication protocols used nowadays in online banking, which often involve bank customers, smart cards, smart-card readers and mobile phones [10]. Therefore, it is important to investigate how notions related to information flow such as non-interference or non-deducibility (see, e.g., [37]) can be formalised and proved through suitable rules. To that end, we are investigating how a connection can be established with the work reported in [17], which proposes a hybrid-logic formalism and proof calculus for modelling and reasoning about hybrid-dynamic information flow in cyber-physical systems.

The systems we have considered in this work are closed in the sense that connectivity and locality are bound by a fixed space of possible locations and also by a fixed set of possible connections between the actors at those locations. Bigraphs are more general in that they address open systems, i.e., systems in which locality and connectivity are not bound at design time. Therefore, another research direction that we are pursuing aims at investigating how the formal approach that we put forward in [38, 39] for the run-time interconnection of asynchronous systems, which like bigraphs are based on hypergraphs, could be used to extend the framework that we have proposed in this paper.

Acknowledgements

The authors would like to thank Fernando Orejas for his insightful feedback and for many helpful discussions on the topic, as well as the two anonymous referees for having read the paper so carefully and for their detailed comments and suggestions. The work of Dusko Pavlovic was partially supported by NSF and AFOSR; José Luiz Fiadeiro and Antónia Lopes received support from AFOSR for research visits to the University of Hawaii.

References

[1] P. Blackburn, Representation, reasoning, and relational structures: A hybrid logic manifesto, Logic Journal of the IGPL 8 (3) (2000) 339–365.

[2] A. Prior, Past, Present and Future, Oxford University Press, 1967.

[3] T. Braüner, Hybrid Logic and its Proof-Theory, Vol. 37 of Applied Logic Series, Springer, 2011.

[4] R. Diaconescu, Quasi-varieties and initial semantics for hybridized institutions, Journal of Logic and Computation 26 (3) (2016) 855–891.

[5] D. Găină, Foundations of logic programming in hybrid logics with user-defined sharing, Theoretical Computer Science 686 (2017) 1–24.

[6] D. Găină, Birkhoff style calculi for hybrid logics, Formal Aspects of Computing 29 (5) (2017) 805–832.

[7] R. Neves, A. Madeira, M. A. Martins, L. S. Barbosa, Proof theory for hybrid(ised) logics, Science of Computer Programming 126 (2016) 73–93.

[8] A. Madeira, R. Neves, L. S. Barbosa, M. A. Martins, A method for rigorous design of reconfigurable systems, Science of Computer Programming 132 (2016) 50–76.

[9] M. A. Martins, A. Madeira, R. Diaconescu, L. S. Barbosa, Hybridization of institutions, in: A. Corradini, B. Klin, C. Cîrstea (Eds.), Algebra and Coalgebra in Computer Science – 4th International Conference, CALCO 2011, Winchester, UK, August 30 – September 2, 2011. Proceedings, Vol. 6859 of Lecture Notes in Computer Science, Springer, 2011, pp. 283–297.

[10] D. Pavlovic, C. A. Meadows, Actor-network procedures (extended abstract), in: R. Ramanujam, S. Ramaswamy (Eds.), Distributed Computing and Internet Technology, Vol. 7154 of Lecture Notes in Computer Science, Springer, 2012, pp. 7–26.

[11] B. Latour, Reassembling the Social: An Introduction to Actor-Network Theory, Oxford University Press, 2005.

[12] R. Milner, The Space and Motion of Communicating Agents, Cambridge University Press, 2009.

[13] A. Platzer, Logical Foundations of Cyber-Physical Systems, Springer, 2018.

[14] A. Deshpande, A. Göllü, P. Varaiya, SHIFT: A formalism and a programming language for dynamic networks of hybrid automata, in: P. J. Antsaklis, W. Kohn, A. Nerode, S. Sastry (Eds.), Hybrid Systems IV, Proceedings of the Fourth International Workshop on Hybrid Systems, Ithaca, NY, USA, October 1996, Vol. 1273 of Lecture Notes in Computer Science, Springer, 1996, pp. 113–133.

[15] R. Banach, M. J. Butler, S. Qin, N. Verma, H. Zhu, Core hybrid Event-B I: Single hybrid Event-B machines, Science of Computer Programming 105 (2015) 92–123.

[16] E. M. Clarke, P. Zuliani, Statistical model checking for cyber-physical systems, in: T. Bultan, P. Hsiung (Eds.), Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11–14, 2011. Proceedings, Vol. 6996 of Lecture Notes in Computer Science, Springer, 2011, pp. 1–12.

[17] B. Bohrer, A. Platzer, A hybrid, dynamic logic for hybrid-dynamic information flow, in: A. Dawar, E. Grädel (Eds.), Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09–12, 2018, ACM, 2018, pp. 115–124.

[18] A. Madeira, R. Neves, M. A. Martins, L. S. Barbosa, Hierarchical hybrid logic, Electronic Notes in Theoretical Computer Science 338 (2018) 167–184.

[19] G. Conforti, D. Macedonio, V. Sassone, Spatial logics for bigraphs, in: L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11–15, 2005, Proceedings, Vol. 3580 of Lecture Notes in Computer Science, Springer, 2005, pp. 766–778.

[20] R. Milner, Axioms for bigraphical structure, Mathematical Structures in Computer Science 15 (6) (2005) 1005–1032.

[21] A. Pnueli, The temporal logic of programs, in: 18th Annual Symposium on Foundations of Computer Science, IEEE CS, 1977, pp. 46–57.

[22] P. Mateus, A. Sernadas, C. Sernadas, Exogenous semantics approach to enriching logics, in: Essays on the Foundations of Mathematics and Logic, Vol. 1 of Advanced Studies in Mathematics and Logic, Polimetrica, 2005, pp. 165–194.

[23] J. L. Fiadeiro, I. Tutu, A. Lopes, D. Pavlovic, Logics for actor networks: A case study in constrained hybridization - A case study in constrained hybridization, in: A. Madeira, M. R. F. Benevides (Eds.), Dynamic Logic. New Trends and Applications – First International Workshop, DALI 2017, Brasilia, Brazil, September 23–24, 2017, Proceedings, Vol. 10669 of Lecture Notes in Computer Science, Springer, 2017, pp. 98–114.

[24] P. Blackburn, M. Tzakova, Hybrid languages and temporal logic, Logic Journal of the IGPL 7 (1) (1999) 27–54.

[25] P. Blackburn, J. Seligman, Hybrid languages, Journal of Logic, Language and Information 4 (3) (1995) 251–272.

[26] V. Goranko, Hierarchies of modal and temporal logics with reference pointers, Journal of Logic, Language and Information 5 (1) (1996) 1–24.

[27] P. Blackburn, M. de Rijke, Y. Venema, Modal Logic, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2002.

[28] G. Malinowski, Many-Valued Logics, Oxford Logic Guides, Clarendon Press, 1993.

[29] B. Renne, J. Sack, A. Yap, Dynamic epistemic temporal logic, in: X. He, J. F. Horty, E. Pacuit (Eds.), Logic, Rationality, and Interaction, Second International Workshop, LORI 2009, Chongqing, China, October 8–11, 2009. Proceedings, Vol. 5834 of Lecture Notes in Computer Science, Springer, 2009, pp. 263–277.

[30] T. Braüner, Modal logic, truth, and the master modality, Journal of Philosophical Logic 31 (4) (2002) 359–386.

[31] C. Hoare, An axiomatic basis for computer programming, Communications of the ACM 12 (10) (1969) 576–580.

[32] V. R. Pratt, Semantical considerations on floyd-hoare logic, in: 17th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1976, pp. 109–121.

[33] V. R. Pratt, Application of modal logic to programming, Studia Logica 39 (2) (1980) 257–274.

[34] D. Harel, D. Kozen, J. Tiuryn, Dynamic Logic, MIT Press, 2000.

[35] R. Goldblatt, Axiomatising the logic of computer science, Vol. 130 of Lecture Notes in Computer Science, Springer Verlag, 1982.

[36] H. Ehrig, C. Ermel, U. Golas, F. Hermann, Graph and Model Transformation – General Framework and Applications, Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2015.

[37] P. G. Allen, A comparison of non-interference and non-deducibility using CSP, in: 4th IEEE Computer Security Foundations Workshop – CSFW'91, Franconia, New Hampshire, USA, June 18–20, 1991, Proceedings, IEEE Computer Society, 1991, pp. 43–54.

[38] J. L. Fiadeiro, A. Lopes, Heterogeneous and asynchronous networks of timed systems, Theoretical Computer Science 663 (2017) 1–33.

[39] I. Ţuţu, J. L. Fiadeiro, Service-oriented logic programming, Logical Methods in Computer Science 11 (3) (2015) 1–37.