Logics for Actor Networks:
A Case Study in Constrained Hybridization

José Fiadeiro[1], Ionuţ Ţuţu[1], Antónia Lopes[2], and Dusko Pavlovic[3]

[1] Dep. of Computer Science, Royal Holloway University of London, UK
jose.fiadeiro@rhul.ac.uk, ittutu@gmail.com
[2] Dep. of Informatics, Faculty of Sciences, University of Lisbon, Portugal
malopes@ciencias.ulisboa.pt
[3] Dep. of Information and Computer Sciences, University of Hawaii, USA
dusko@hawaii.edu

Abstract. Actor Networks are a modeling 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 computational networks. We advance a logic for modeling and reasoning about such actor networks, which is obtained through a two-stage constrained hybridization process. The first stage results in a logic that captures the structure of actor networks and the way knowledge flows across them; the second addresses the dynamic aspects of actor networks, that is the way they 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.

1 Introduction

Over the past few years, there has been a renewed interest in modal logics for computer science through the family of the so-called *hybrid logics* (see [1] for a comprehensive overview). The development of hybrid logics 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 dedicated satisfaction operators $@_a$ that enable a change of perspective from the current state to the one that corresponds to the nominal a. A significant body of research exists around this class of logics, among which [3,4,5,6] are recent publications.

In this paper, we are specifically interested in the present-day applications of hybrid logics to the specification and verification of reconfigurable systems [7]. 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 hybrid logic can be developed, through a process known as hybridization [8], on top of an arbitrary logic used for expressing configuration-specific requirements. This means that, depending on the base logic, configurations can be captured, for example, as algebras, relational structures or, when the hybridization process is iterated, even as Kripke models.

Actor Networks. Our interest in this area results from a new modeling framework for cyber-physical system protocols proposed in [9] around the concept of Actor Network (or ANt), which addresses networks whose components are no longer limited to programs but can also include humans or physical artifacts as actors. ANts should therefore be understood in the wider sense of Latour's actor-network theory [10]: actors are cyber-physical entities that have shared agency—from people, to objects, and to locations; they interact through so-called channels that account, for example, for observations that an actor may make of another, of control that an actor may exert on another, or of movement of an actor inside another (say, a person to a location). Interaction, rather than computation, has become the critical source of complexity, thus giving rise to new challenges in ensuring the reliability of the systems that are now operating in cyberspace.

Contributions. The ordinary hybridization process yields logical systems that are suitable for dealing with either the static/structural aspects or with the dynamic aspects of actor networks. From a static perspective, hybrid logics can be naturally used, for example, to give 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, accommodating at the same time both the structure and the behaviour of ANts raises some difficulties because these two aspects require distinct, and possibly conflicting, interpretations of the hybrid features. For example, from a structural point of view, modalities denote channels, whereas from a behavioural point of view they stand for graphs of interactions between actors. That is, the challenge raised by ANts lies precisely in capturing the way in which the structure of such networks evolves. This leads us to propose a two-layered hybridization process, where the first level corresponds to the structure, and the second to the dynamics of actor networks.

The paper consists of two main technical sections. In Section 2 we introduce the underlying model theory of actor networks. We start by formalizing the main static concepts: actors, interaction channels, the knowledge that actors may have and the way it may be acquired across certain channels, and the placement of some actors relative to other actors. Then, we formalize the key notion of interaction and the way an interaction can change the state of a network.

Section 3 is devoted to the logics through which we can specify and reason about the states of an actor network and about the state transitions associated with interactions. These logics are obtained through a sequence of two processes of constrained hybridization, meaning that (a) the models of the hybrid logics implicitly satisfy additional semantic constraints, and (b) we actually operate across three logical levels—each level captures a different aspect of actor networks (knowledge, structure, or dynamics), and each is defined as an exogenous enrichment (with new hybrid features) of the previous level.

Notational Conventions. 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 avoid 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 $\mathcal{A}$.

2 Actor Networks

2.1 Schemas

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

- a finite directed graph $\mathcal{G} = \langle \mathcal{N}, \mathcal{C}, \delta, \rho \rangle$, where $\mathcal{N}$ is a non-empty set (of nodes, called *actors*), $\mathcal{C}$ is a set (of edges, called *channels*), and δ and ρ are maps $\mathcal{C} \to \mathcal{N}$ that give the domain (origin) and codomain (target) of every channel;
- a partially ordered set $\mathcal{T}$ (of *channel types*, with a subtype relationship);
- a function $\tau \colon \mathcal{C} \to 2^\mathcal{T}$ that assigns a non-empty upper set[4] of channel types to every channel, such that for every $n, n' \in \mathcal{N}$ and $\kappa \in \mathcal{T}$ there is at most one channel $c \in \mathcal{C}$ such that $\delta(c) = n$, $\rho(c) = n'$, and $\kappa \in \tau(c)$; and
- a set $\mathcal{P}$ (of *propositional symbols*).

The nodes of an ANt schema represent *actors* executing a given protocol and edges represent *channels* that link together those actors. Channels are typed in order to account for different modes of relationship between actors. 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 the ANt schema Elevator whose graph and typing function are depicted in Fig. 1. The nodes *F0* and *F1* correspond to the ground and first floor of a building, and E to the elevator proper, which we often refer to as Elevator unless it is ambiguous. The node C corresponds to the elevator 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 refer to as Alice.

Elevator has a number of channels of different types:

- The channel type *mov* captures the movement of one actor inside another. The two channels of type *mov* that connect *P0* and *P1* allow the cabin to move between the two platforms (up or down).

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

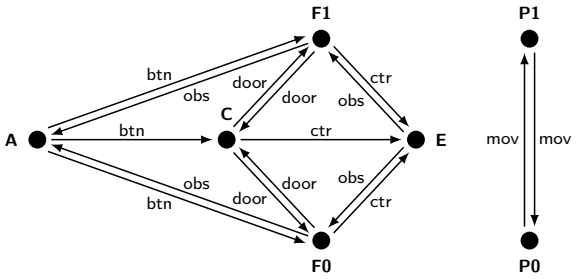

Fig. 1. The graph and typing function of the ANt schema Elevator

– The type *door* is a subtype of *mov*. The two channels of type *door* connect *F0* and *F1* to *C* in order to allow users to enter or exit the cabin from or to the floor. Although these two channels are also of type *mov*, for readability we tend to depict only the minimal types when representing ANt schemas.
– The channel type *obs* captures observations that an actor may make of another. The channels of type *obs* that connect *E* to *F0* and *F1* account for observations of the state of Elevator at either floor, while those that connect *F0* and *F1* to *A* account for observations that Alice makes of either floor.
– The channel type *ctr* captures forms of control that one actor may exert on another. The two channels of type *ctr* that connect *F0* to *E* and *F1* to *E* are for transmitting requests from floors to Elevator, and the channel that connects *C* to *E* is for transmitting requests from Cabin to Elevator.
– The channel subtype *btn* of *ctr* captures the special case of control achieved through a button. The three channels of type *btn* that connect *A* to *F0*, *F1*, and *C* account for the buttons that Alice can press at either floor or at Cabin.

Last but not least, the ANt schema Elevator has two propositional symbols, C_at_P0 and C_at_P1. These are used to capture knowledge of where Cabin is.

2.2 States

A structure for an ANt schema $\mathcal{A}$ consists of a subgraph of $\mathcal{G}_\mathcal{A}$ together with a forest (or *placement graph*, as in [11]) over its nodes that captures 'location'.

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 has either none or a unique parent, denoted $\mathcal{F}(n)$. Nodes without a parent are called *roots*.

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 $n \in \mathcal{N}_{\mathcal{H}_1}$ such that $\mathcal{F}_1(n)$ is defined, $\mathcal{F}_2(n)$ is also defined and equal to $\mathcal{F}_1(n)$—that is, the hierarchy is strictly preserved.

The notion of substructure defines a partial order, which we denote by $\preceq$.

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

The graph, on the left, indicates the channels that are available: for example, the channel that corresponds to the button that Alice can press to call the elevator, and the one that connects *F0* to *E* and allows the floor to transmit requests to Elevator. Notice that, for readability, we always include the channel types in figures, even though they are not formally part of ANt structures.

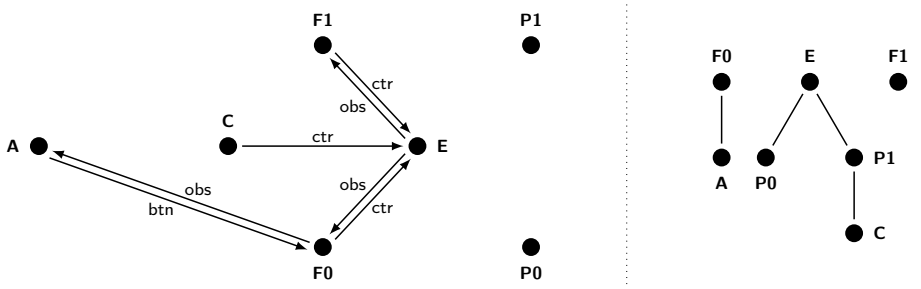

Fig. 2. An ANt structure for Elevator: the graph on the left and the forest on the right

To better visualize ANt structures, we combine the graph and the forest components through the nesting of nodes in the graph. This can be seen in Fig. 3, where three ANt structures are presented.

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. We work with a three-valued Łukasiewicz logic, i.e., propositions may have values $\oplus$ (true), $\ominus$ (false), or $\pm$ (undefined).

Definition 5 (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 $\mathcal{V}_n \colon \mathcal{P}_\mathcal{A} \to \{\oplus, \ominus, \pm\}$.

We denote the set of states of an ANt schema $\mathcal{A}$ by $\mathbb{S}_\mathcal{A}$ and, following our notational convention, the structure underlying a state σ by $\mathcal{S}_\sigma$.

Example 6. As an example, we define the state $\mathsf{elevator}_0$ whose underlying structure is shown in the top-left part of Fig. 4 and whose valuation is given by:

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

That is, the actors/nodes *E*, *F0*, *F1* and *A* all know that Cabin is at the platform *P1* (and that it is not at *P0*); no other node knows where Cabin is.

2.3 Interactions

Channels provide the means for actors to interact with each other. The interactions through which actors change protocol states can be more complex (in the sense that they can involve many actors and channels) and are therefore defined as ANt structures: given an interaction, its nodes are the actors of the ANt schema that are involved in the interaction, and its channels are those through which those actors interact with each other.

Definition 7 (Interaction). An interaction in the context of an ANt schema $\mathcal{A}$ is a structure for $\mathcal{A}$. We denote by $\mathbb{I}_{\mathcal{A}}$ the set of all interactions of $\mathcal{A}$.

Example 8. Figure 3 depicts three interactions for the ANt schema Elevator:

(a) callElevator0: Alice is at the ground floor and presses the button to call the elevator; the request is transmitted to Elevator through a *ctr* channel.
(b) moveCabin0: Cabin is at the first floor and the channel through which it can move to the ground floor is available.
(c) enterCabin0: Alice is at the ground floor and observes the position of the cabin through the two channels of type *obs*; the channel of type *door* that connects *F0* to *C* is available for Alice to enter the cabin.

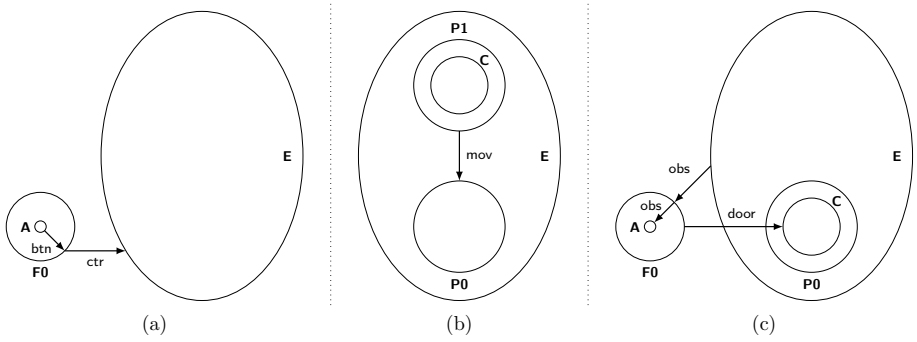

Fig. 3. Interactions of Elevator: (a) callElevator0, (b) moveCabin0, and (c) enterCabin0

2.4 Networks

Protocols are formalized as actor networks. An actor network consists of (a) an ANt schema, which contains all the actors and the channels that connect them; (b) a set of possible worlds—each being associated with a so-called admissible state—including a subset of designated initial worlds; (c) a set of all possible interactions through which the actor network can evolve; and (d) for every such interaction, a transition relation on the set of worlds. Formally,

Definition 9 (Actor network). An actor network ν consists of:

- an ANt schema $\mathcal{A}$,
- a *domain* (set of *worlds*) $\mathcal{D}$ together with a labeling function $\varsigma\colon \mathcal{D} \to \mathbb{S}_{\mathcal{A}}$,
- a non-empty subset $\mathcal{D}_0 \subseteq \mathcal{D}$ of *initial worlds* (whose labels are *initial states*),
- a set $\mathcal{I} \subseteq \mathbb{I}_{\mathcal{A}}$ of interactions for $\mathcal{A}$,
- a *transition relation* $(\longrightarrow) \subseteq \mathcal{D} \times \mathcal{I} \times \mathcal{D}$ such that, for each interaction $\iota \in \mathcal{I}$, $w \xrightarrow{\iota} w'$ implies $\iota \preceq \mathcal{S}_{\varsigma(w)}$ (interactions are substructures of the source states).

We say that a state of $\mathcal{A}$ is *admissible* for ν if it corresponds to one of its worlds; we denote by $\mathbb{S}_\nu$ the image of $\mathcal{D}$ under ς (i.e., the set of admissible states of ν).

Therefore, an actor network can be regarded as a labeled transition system over a set of states of the schema, transitions being labeled with interactions.

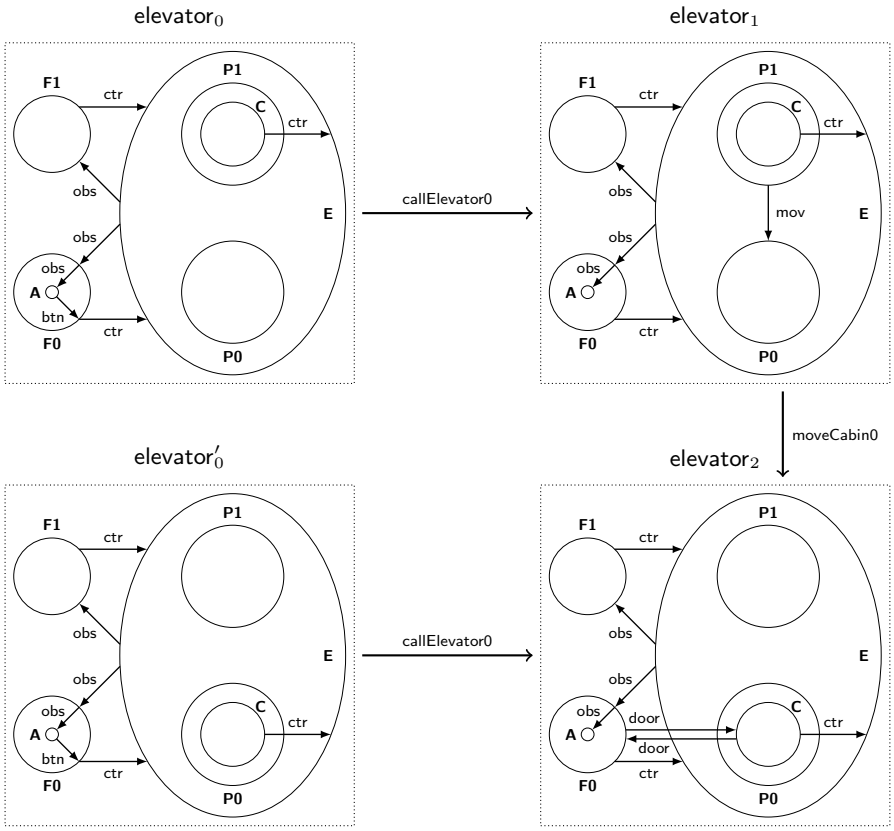

Fig. 4. Transitions performed by the interactions callElevator0 and moveCabin0

Example 10. An actor network ν_{Elevator} with Elevator as its schema could have, for example, $\mathsf{elevator}_0$ (labeled with the state defined in Example 4) as one of its initial worlds, and the worlds and transitions presented in Fig. 4, among others. Note that the valuations are not included in these diagrams; an axiomatic presentation of the valuations is discussed in Section 3.

The 'horizontal' transitions in Fig. 4 are performed by the interaction callElevator0 (cf. Fig. 3(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 channel of type *mov* that connects *P1* to *P0* through which Elevator can respond to the request (i.e., 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.

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

The 'vertical' transition from $\mathsf{elevator}_1$ to $\mathsf{elevator}_2$ is performed by the interaction moveCabin0 (cf. Fig. 3(b)). As indicated by the interaction, this computation is local to *P0*, *P1*, *F0*, *F1*, E, and C. 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 exit the cabin.

3 Logics for ANts

3.1 The base logic

The logics through which we can specify and reason about actor networks are obtained through an iterated process of constrained hybridization. At the base of this construction is the three-valued propositional Łukasiewicz logic, which we recall below. A signature for the Łukasiewicz logic is given by a set $\mathcal{P}$ of *atomic propositions* (the propositional symbols defined by an ANt schema).

Definition 11 (Syntax). The set $\mathsf{Ł}(\mathcal{P})$ of sentences of the base logic is the least set that includes $\mathcal{P}$ and is closed under negation ($\overline{p}$) and implication ($p \supset q$).

Definition 12 (Semantics). Base-logic sentences are interpreted over functions $[\![\_]\!]\colon \mathcal{P} \to \{\oplus, \ominus, \oplusminus\}$ as follows: The negation maps $\oplus$ to $\ominus$ and vice versa, and leaves $\oplusminus$ unchanged. The implication $p \supset q$ evaluates to $\ominus$ if $p = \oplus$ and $q = \ominus$, to $\oplusminus$ if $p = \oplus$ and $q = \oplusminus$ or $p = \oplusminus$ and $q = \ominus$, and to $\oplus$ in all other cases. We say that a proposition p is valid if $[\![p]\!] = \oplus$ for all valuations $[\![\_]\!]$.

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

$\mathsf{M}\, p \triangleq (\overline{p} \supset p)$ $\quad$ p is possibly true—i.e., it has value $\oplus$ or $\oplusminus$

$\mathsf{L}\, p \triangleq \overline{\mathsf{M}\, \overline{p}}$ p is necessarily true—i.e., it has value $\oplus$

$\mathsf{N}\, p \triangleq \overline{\mathsf{M}\, p} = \mathsf{L}\, \overline{p}$ p is necessarily false—i.e., it has value $\ominus$

$\mathsf{I}\, p \triangleq \overline{\mathsf{M}\, p \supset \mathsf{L}\, p}$ p is unknown—i.e., it has value $\pm$

3.2 The state logic

The logic through which we can specify and reason about the states of an ANt is a constrained hybridization of $\mathsf{Ł}(\mathcal{P})$. Therefore, there are two main ingredients to consider here. Firstly, an *ANt-schema signature*, denoted Σ in what follows, which consists of a set $\mathcal{P}$ of propositional symbols (i.e., a signature of the base Łukasiewicz logic), a countably infinite set $\mathcal{N}om$ of nominals that includes a set $\mathcal{N}$ of actor names, and a set $\mathcal{T}$ of channel types (regarded as modalities). Secondly, a partial order on $\mathcal{T}$ and an edge-labeled directed finite graph $\mathcal{G}$ with components $\mathcal{N}$, $\mathcal{C}$, δ, ρ, and τ as in Definition 1. These provide the constraints that we impose on the models of the hybrid logic.

Definition 13 (Syntax). The syntax of the state logic is given by the grammar

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

where $p \in \mathsf{Ł}(\mathcal{P})$, $a \in \mathcal{N}om$, $b \in \mathcal{N}om \setminus \mathcal{N}$, $\kappa \in \mathcal{T}$, and π is a distinguished and new *parent* modality. We denote this set of sentences by $\mathsf{State}(\Sigma)$.

In the most general setting, hybrid sentences are evaluated over unconstrained Kripke models, that is over triples $\langle W, R, V \rangle$, where W is a set of nodes or possible worlds, R is a family of accessibility relations $R_\lambda \subseteq W \times W$, indexed by modalities λ, and V is a family of interpretations of the symbols from $\mathcal{P}$ indexed by possible worlds. The semantics of hybrid logics often includes additional constraints; for example, in the S4 variant of hybrid propositional logic, the accessibility relations are reflexive and transitive, and in the S5 variant they are reflexive and Euclidean. The constraints that we consider for the state logic follow from the underlying graph structure of the ANt schema used:

- There is a one-to-one correspondence between actors and possible worlds. For notational convenience, we do not distinguish possible worlds from actors.
- The accessibility relations conform to the channels and the channel types of the schema: for each channel type κ, R_κ consists of those pairs of nodes (n, n') that are connected through a channel of type κ.
- The interpretation of the parent modality π is functional and acyclic.

In other words, the constrained models of the hybridization of $\mathsf{Ł}(\mathcal{P})$ that we consider here are states of the actor-network schema $\mathcal{A} = \langle \mathcal{G}, \mathcal{T}, \tau, \mathcal{P} \rangle$.

Definition 14 (Semantics). Given a state $\sigma = \langle \mathcal{S}, \mathcal{V} \rangle$ of $\mathcal{A}$, an *assignment* is a map $\alpha \colon \mathcal{N}om \to \mathcal{N}_\mathcal{S}$ whose restriction to the set $\mathcal{N}$ of actor names is the identity.[5] The satisfaction relation between ANt states and state-logic sentences is parameterized by assignments α and by actors n (i.e., by nodes of $\mathcal{S}$):

[5] Recall that, by Definition 5, $\mathcal{N}_\mathcal{S} = \mathcal{N}$.

– $\sigma, \alpha, n \vDash a$ iff $\alpha(a) = n$;
– $\sigma, \alpha, n \vDash p$ iff $[\![p]\!]_n = \oplus$ where $[\![\_]\!]_n$ 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 there is $c \in \mathcal{C}$ such that $\delta(c) = n$, $\kappa \in \tau(c)$, and $\sigma, \alpha, \rho(c) \vDash \phi$;
– $\sigma, \alpha, n \vDash \langle\pi\rangle\, \phi$ iff $\mathcal{F}(n)$ is defined and $\sigma, \alpha, \mathcal{F}(n) \vDash \phi$;
– $\sigma, \alpha, n \vDash @_a\, \phi$ iff $\sigma, \alpha, \alpha(a) \vDash \phi$;
– $\sigma, \alpha, n \vDash \exists b\, \phi$ iff $\sigma, \alpha', n \vDash \phi$ for some α' that agrees with α on $\mathcal{N}om \setminus \{b\}$.

We also define validity of a sentence at a state to mean that it is satisfied for every assignment at every node, validity of a sentence at an ANt structure to mean that it is satisfied at every state for it, and absolute validity of a sentence (at a schema) to mean that it is valid at every state of the schema:

– $\sigma \vDash \phi$ iff $\sigma, \alpha, n \vDash \phi$ for all assignments $\alpha \colon \mathcal{N}om \to \mathcal{N}$ and all $n \in \mathcal{N}$;
– $\mathcal{S} \vDash \phi$ iff $\sigma \vDash \phi$ for all states σ such that $\mathcal{S} \preceq \mathcal{S}_\sigma$;
– $\mathcal{A} \vDash \phi$, or simply $\vDash \phi$, iff $\sigma \vDash \phi$ for all $\sigma \in \mathbb{S}_\mathcal{A}$.

The validity relations extend to sets of sentences to mean that every sentence in the set is valid. Given a set Φ of sentences, we denote by $\mathbb{S}_\Phi$ the set of states over which all the sentences in Φ are valid.

We use the usual propositional connectives for conjunction ($\wedge$) and disjunction ($\vee$), as well as the dual modal operators ($[\_]$) and quantifier ($\forall$) given by:

– $[\kappa]\, \phi \triangleq \neg\, \langle\kappa\rangle\, \neg\, \phi$
That is, $\sigma, \alpha, n \vDash [\kappa]\, \phi$ iff $\sigma, \alpha, \rho(c) \vDash \phi$ for all $c \in \mathcal{C}$ with $\delta(c) = n$ & $\kappa \in \tau(c)$.
– $[\pi]\, \phi \triangleq \neg\, \langle\pi\rangle\, \neg\, \phi$
That is, $\sigma, \alpha, n \vDash [\pi]\, \phi$ iff $\sigma, \alpha, \mathcal{F}(n) \vDash \phi$ if $\mathcal{F}(n)$ is defined.
– $\forall b\, \phi \triangleq \neg\, \exists b\, \neg\, \phi$
That is, $\sigma, \alpha, n \vDash \forall b\, \phi$ iff $\sigma, \alpha', n \vDash \phi$ for all α' that agree with α on $\mathcal{N}om \setminus \{b\}$.

Notice that the symbols used for the negation ($\bar{}$) and implication ($\supset$) in the base logic are different from those of the state logic ($\neg$ and $\to$, respectively) to mark the difference between the two levels.

Example 15. The sentences presented below are properties of (valid at) the state $\mathsf{elevator}_0$ from Example 6, of its underlying structure $\mathcal{S}_{\mathsf{elevator}_0}$, or of the ANt schema itself. Naturally, any property of $\mathcal{S}_{\mathsf{elevator}_0}$ is also a property of $\mathsf{elevator}_0$, and any property of Elevator is also a property of $\mathcal{S}_{\mathsf{elevator}_0}$. Notice, however, that the converse does not hold for the following sentences:

s1 $\mathsf{elevator}_0 \vDash (E \vee F0 \vee F1 \vee A) \to \mathbf{L}\,(\mathsf{C\_at\_P1})$
The actors E, $F0$, $F1$, and A know that the cabin is at the first platform.

S2 $\mathcal{S}_{\mathsf{elevator}_0} \vDash @_A\,[btn]\,\langle ctr\rangle\,E$
Whenever Alice calls the elevator, the request is transmitted to the Elevator.

S3 $\mathsf{Elevator} \vDash \big((F0 \vee F1) \to [obs]\,A\big) \wedge \big(\langle obs\rangle\,A \to (F0 \vee F1)\big)$
The floors can only be observed by Alice, and that is all Alice can observe.

Definition 16 (State specification). A *state specification* for an ANt schema $\mathcal{A}$ 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 $\mathsf{State}(\Sigma)$.

The sentences of a state specification of an ANt schema are used to restrict the set of admissible states of the actor networks defined over that schema.

Example 17. A state specification of Elevator could consist of the instances of the following sentence schemas, which we denote by Φ_{Elevator}:

E1 $@_C\,\langle\pi\rangle\,Pi \leftrightarrow @_E\,\mathsf{L}\,(\mathsf{C\_at\_P}i) \quad$ for $i \in \{0,1\}$
The cabin is at platform i if and only if the elevator knows it.

E2 $p \to [obs]\,p \quad$ for every $p \in \mathsf{Ł}(\mathcal{P})$
Knowledge is propagated through observation channels.

That is, this specification of Elevator determines what knowledge nodes have about the whereabouts of Cabin: the elevator proper always knows where Cabin is and the other nodes can acquire that information through observation channels.

The state logic is useful for deriving properties of states, ANt structures, and of ANt schemas and their specifications.

Definition 18 (Entailment). Given a finite set Φ of sentences and a sentence ϕ (defined over the same signature as Φ), we say that ϕ is a *semantic consequence* of Φ, or that Φ *entails* ϕ, and write $\Phi \vDash \phi$, if $\sigma \vDash \phi$ for all $\sigma \in \mathbb{S}_\Phi$.

The following two propositions allow us to redefine the three kinds validity of a state sentence in terms of entailment. They provide a syntactic characterization (as a set of sentences) $\Phi_\sigma/\Phi_\mathcal{S}$ for every state σ or structure $\mathcal{S}$. A sentence ϕ is valid at a state σ if and only if $\Phi_\sigma \vDash \phi$, and is valid at an ANt structure $\mathcal{S}$ if and only if $\Phi_\mathcal{S} \vDash \phi$; obviously, ϕ is absolutely valid if and only if $\emptyset \vDash \phi$.

Proposition 19. *Let $\Phi_\mathcal{S}$ be the (finite) set of all sentences of the form $@_a\,\langle\lambda\rangle\,b$ that are valid at 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 $\mathbb{S}_{\Phi_\mathcal{S}} = \{\sigma \in \mathbb{S}_\mathcal{A} \mid \mathcal{S} \preceq \mathcal{S}_\sigma\}$.*

Proposition 20. *For every state σ, let Φ_σ be the (finite) set that extends $\Phi_{\mathcal{S}_\sigma}$ with all state sentences of the form $\neg\,@_a\,\langle\kappa\rangle\,b$, $@_a\,[\pi]\,\mathsf{false}$, $@_a\,\mathbf{L}\,p$, $@_a\,\mathbf{N}p$, or $@_a\,\mathbf{I}p$ that are valid at σ, where a and b are actor names, κ is a type that labels one of the channels between a and b in the ANt schema, and p is a propositional symbol. Then σ is the only state that satisfies Φ_σ.*

Axioms

BLT	Axiom schemata for the base Łukasiewicz logic (see e.g. [12])
CT	Axiom schemata 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	$(a \wedge \phi) \rightarrow @_a\,\phi$
λ_E[6]	$([\lambda]\,\phi \wedge \langle\lambda\rangle\,a) \rightarrow @_a\,\phi$
$\forall_E$	$\forall b\,\phi \rightarrow \phi[a/b]$

Inference rules

$$\frac{\phi_1 \qquad \phi_1 \rightarrow \phi_2}{\phi_2}\ (MP) \qquad \frac{\phi}{@_a\,\phi}\ (@_I) \qquad \frac{@_a\,\phi}{\phi}\ (@_E)^*$$

$$\frac{(\phi_1 \wedge \langle\lambda\rangle\,a) \rightarrow @_a\,\phi_2}{\phi_1 \rightarrow [\lambda]\,\phi_2}\ (\lambda_I)^* \qquad \frac{\phi_1 \rightarrow \phi_2[a/b]}{\phi_1 \rightarrow \forall b\,\phi_2}\ (\forall_I)^*$$

$*$ a does not occur free in ϕ ($@_E$), in ϕ_1 or ϕ_2 (λ_I), or in ϕ_1 or $\forall b\,\phi_2$ ($\forall_I$).

Fig. 5. Hilbert-style axiom schemata and rules for basic hybrid logic

Entailment can be derived syntactically through the use of a proof system. The Hilbert-style axiomatization of the basic, unconstrained hybrid logic in Fig. 5 is both a simplification (because we do not consider the binder $\downarrow$) and an extension (due to the multi-modality setting and the different base logic) of the axiom system given in [3, Chapter 2].

Proposition 21 ([3, Chapter 2]). *The axiom schemata and inference rules presented in Fig. 5 are sound and complete with respect to the unconstrained Kripke-frame semantics of basic hybrid logic.*

Unlike the models of the basic hybrid logic, the models of the state logic are subject to the constraints defined by the ANt schema. Because of this, the axiom schemata and inference rules from Fig. 5 are no longer complete—though, obviously, they remain sound—with respect to the constrained Kripke semantics of the state logic. 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 contains, for instance, when considering the ANt schema Elevator from Example 2, sentences like $\neg\, @_A\,\langle ctr\rangle\,E$ (Alice cannot control the elevator directly.), while the latter contains sentences like $\neg\, @_a\,b$, where a and b are distinct actor names, or $\langle\pi\rangle\,\phi \rightarrow [\pi]\,\phi$.

In order to regain completeness, we introduce new axioms that reflect the semantic constraints of the state logic. The axiom schemata *N1* and *N2* from

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

Fig. 6 ensure that all possible worlds correspond to actor names, and that no two distinct names are interpreted in the same way. The axiom schema *C1* rules out those channels that are not defined in the ANt schema, while *C2* captures the channel subtyping relation. Lastly, $\pi 1$ and $\pi 2$ specify that the interpretations of the distinguished parent modality are functional and acyclic, respectively.

N1	$\bigvee\{a \mid a \in \mathcal{N}\}$
N2	$\neg @_a\, b \quad \text{for } a \neq b \in \mathcal{N}$
C1	$\neg @_a \langle\kappa\rangle b \quad \text{if there is no } c \in \mathcal{C} \text{ with } \delta(c) = a,\ \rho(c) = b, \text{ and } \kappa \in \tau(c)$
C2	$\langle\kappa\rangle a \rightarrow \langle\kappa'\rangle a \quad \text{for } \kappa \leq \kappa' \text{ and } a \in \mathcal{N}$
$\pi 1$	$\langle\pi\rangle a \rightarrow [\pi] a \quad \text{for } a \in \mathcal{N}$
$\pi 2$	$\neg @_a \langle\pi\rangle^n a \quad \text{for } 1 \leq n \leq \lvert\mathcal{N}\rvert \text{ and } a \in \mathcal{N}$

Fig. 6. Additional axiom schemata for the state logic

Definition 22. Under the notations and assumptions of Definition 18, we say that ϕ is provable from Φ, and write $\Phi \vdash \phi$, if and only if ϕ can be derived from Φ using the axiom schemata and inference rules from Fig. 5 and 6.

The soundness and completeness of the proof system for the state logic follow from Proposition 21 and the lemma below.

Lemma 23. *For any ANt schema $\mathcal{A}$, the states in $\mathbb{S}_{\mathcal{A}}$ are given precisely by those Kripke structures that satisfy the axioms in Fig. 6.*

Proposition 24. *The extension of the proof system for hybrid logic with the axiom schemata in Fig. 6 yields a sound and complete axiomatization of the state logic.*

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

Example 25. The properties $@_{F0}\, \mathsf{L}\,(\mathsf{C\_at\_P1})$ and $@_A\, \mathsf{L}\,(\mathsf{C\_at\_P1})$ can be derived for $\mathcal{S}_{\mathsf{elevator}_0}$ under the specification Φ_{Elevator} from Example 17. In symbols,

$$\Phi_{\mathcal{S}_{\mathsf{elevator}_0}} \cup \Phi_{\mathsf{Elevator}} \vdash @_{F0}\, \mathsf{L}\,(\mathsf{C\_at\_P1}), @_A\, \mathsf{L}\,(\mathsf{C\_at\_P1})$$

This example shows that valuations can sometimes be fully determined by the ANt structure and the axioms associated with the ANt schema. In this particular case, the valuation of the atomic proposition C_at_P1 at the nodes *F0* and *A* can be derived from the structural properties of the ANt structure and from the axiomatization of the way in which knowledge is propagated (see Example 17).

There are other general properties of Elevator that we might want to prove. For example, $@_C \langle\pi\rangle (P0 \vee P1)$—the cabin is 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. The corresponding logic for this kind of proofs is defined in the next sub-section.

3.3 The ANt logic

The logic through which we can reason about the actor networks of an ANt schema $\mathcal{A}$ requires a further level of hybridization. In this case, a higher-level *actor-network signature* Ω consists of a signature Σ of the state logic (now playing the role of the base logic), a countably infinite set $\mathcal{N}om$ of nominals together with a non-empty subset $\mathit{Init} \subseteq \mathcal{N}om$ of names of initial states, and a set $\mathcal{I}$ of interactions for $\mathcal{A}$ (regarded as modalities).

Definition 26 (Syntax). The syntax of the ANt logic is given by the grammar

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

where $\phi \in \mathsf{State}(\Sigma)$, $i \in \mathcal{N}om$, $j \in \mathcal{N}om \setminus \mathit{Init}$, and $\iota \in \mathcal{I}$. We denote by $\mathsf{ANt}(\Omega)$ the set of ANt-logic sentences defined over the signature Ω.

Notice that we use double symbols for the connectives of the ANt logic, and that the satisfaction operators are denoted using a colon. We extend the use of the double-symbol notation to the dual modal operators ($[\![\_]\!]$) and to the universal quantifier ($\forall\!\!\forall$), which are defined as in Section 3.2.

Example 27. We can now write sentences about the dynamics of $\mathsf{Elevator}$ like

$$\bigwedge \Phi_{\mathsf{elevator}_0} \Rightarrow \langle\!\langle \mathsf{callElevator0} \rangle\!\rangle\, \bigwedge \Phi_{\mathsf{elevator}_1}$$

meaning that at the state $\mathsf{elevator}_0$ (which, by Proposition 20, is characterized by the sentences in $\Phi_{\mathsf{elevator}_0}$) there is a transition to the state $\mathsf{elevator}_1$ performed by the interaction $\mathsf{callElevator0}$. Note that, by Proposition 20, the sets of sentences $\Phi_{\mathsf{elevator}_0}$ and $\Phi_{\mathsf{elevator}_1}$ are finite, hence the conjunctions in the antecedent and consequent of the implication above are well formed.

The semantics of the ANt logic is defined once more by means of constrained Kripke models. This time, we restrict only the interpretations of the modalities: all interactions $\iota \in \mathcal{I}$ are substructures of the underlying structures of the states on which (the relational interpretations of) ι are defined (see Definition 9). That is, the models of the ANt logic are actor networks.

Definition 28 (Semantics). The satisfaction relation for the ANt logic is defined for an actor network ν with interactions according to Ω, an assignment $\alpha\colon \mathcal{N}om \to \mathcal{D}_\nu$ (which, in this case, is just a function), and a world $w \in \mathcal{D}_\nu$:

- $\nu, \alpha, w \Vdash i$ iff $\alpha(i) = w$;
- $\nu, \alpha, w \Vdash \phi$ iff $\varsigma_\nu(w) \vDash \phi$;
- $\nu, \alpha, w \Vdash \neg\, \psi$ iff $\nu, \alpha, w \nVdash \psi$;
- $\nu, \alpha, w \Vdash \psi_1 \Rightarrow \psi_2$ iff $\nu, \alpha, w \Vdash \psi_1$ implies $\nu, \alpha, w \Vdash \psi_2$;
- $\nu, \alpha, w \Vdash \langle\!\langle \iota \rangle\!\rangle\, \psi$ iff there is a transition $w \xrightarrow{\iota} w'$ in ν such that $\nu, \alpha, w' \Vdash \psi$;
- $\nu, \alpha, w \Vdash i : \psi$ iff $\nu, \alpha, \alpha(i) \Vdash \psi$;

– $\nu, \alpha, w \Vdash\!\models \exists j\, \psi$ iff $\nu, \alpha', w \Vdash\!\models \psi$ for some α' that agrees with α on $\mathcal{N}om \setminus \{j\}$.

Similarly to the first level of hybridization, we say that an ANt-logic sentence ψ defined over Ω is valid in an actor network 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:

– $\nu \Vdash\!\models \psi$ iff $\nu, \alpha, w \Vdash\!\models \psi$ for all assignments $\alpha\colon \mathcal{N}om \to \mathcal{D}_\nu$ and all $w \in \mathcal{D}_\nu$;

– $\Vdash\!\models \psi$ iff $\nu \Vdash\!\models \psi$ for all actor networks ν 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; and, given another sentence ψ, we say that Ψ entails ψ, which we denote $\Psi \Vdash\!\models \psi$, if $\nu \Vdash\!\models \psi$ for all $\nu \in \mathbb{N}_\Psi$.

The proof theory for the ANt logic builds once again on the proof theory for the basic hybrid logic. To that end, we use the same axiom schemata and inference rules from Fig. 5, only that in this case the tautologies of the Łukasiewicz logic are replaced by those of the state logic, and the Boolean and hybrid connectives of the state logic are replaced by those of the ANt logic. In addition, through the axiom schema *Inter* from Fig. 7, we introduce new axioms that reflect the semantic constraints of the models of the ANt logic: state properties of interactions hold in the states where the transitions occur.

Inter	$\langle \iota \rangle \,\mathsf{true} \Rightarrow \phi$	for all interactions ι and state-logic sentences $\phi \in \Phi_\iota$

Fig. 7. Additional axiom schema for the ANt logic

Definition 29. An ANt sentence ψ is provable from a set Ψ of sentences (of the same signature as ψ), denoted $\Psi \Vdash\!\vdash \psi$, if ψ can be derived from Ψ using the axiom system for hybrid logic and the additional axiom schema defined in Fig. 7.

Example 30. Consider the following axiomatization of the transitions of an actor network for the ANt schema Elevator. Most of the sentences below are of the form $\phi_1 \Rightarrow [\![\iota]\!]\, \phi_2$. They generalize Hoare triples and express properties of the transitions performed by interactions: intuitively, the sentence ϕ_1 is a precondition under which the interaction ι ensures the postcondition ϕ_2.

T1 $@_C \langle \pi \rangle\, P1 \Rightarrow [\![\mathsf{callElevator0}]\!]\, @_{P1} \langle mov \rangle\, P0$
When the elevator is called (at the ground floor) and the cabin is at the first platform, a request to move the cabin to the ground platform is issued.

T2 $@_C \langle \pi \rangle\, P0 \Rightarrow [\![\mathsf{callElevator0}]\!]\, @_{F0} \langle door \rangle\, (C \wedge \langle door \rangle\, F0)$
If the cabin is already at the ground platform, then the doors are opened.

T3 $[\![\mathsf{moveCabin0}]\!]\, @_{F0} \langle door \rangle\, (C \wedge \langle door \rangle\, F0)$
The doors are opened whenever the cabin moves to the (ground) platform.

T4 $@_A \langle\pi\rangle (F0 \wedge \langle door\rangle C) \Rightarrow \langle\!\langle \mathsf{enterCabin0} \rangle\!\rangle \mathsf{true}$
If Alice is at *F0* and the doors are open, then she can enter the cabin.

T5 $@_a \langle\pi\rangle s \Rightarrow [\![\iota]\!] @_a \langle\pi\rangle t$ for interactions ι such that $\iota \vDash @_s \langle mov\rangle t$
Any interaction that involves a channel of type *mov* between actors s and t (regarded as locations) determines the movement to t of any actor in s.

T6 $@_a \langle\pi\rangle s \Rightarrow [\![\iota]\!] @_a \langle\pi\rangle s$ for interactions ι such that $\iota \nvDash @_s \langle mov\rangle \mathsf{true}$
But if the interaction does not involve a *mov* channel starting as s, then the actors in s maintain their location.

Then we can derive complex actor-network sentences such as:

$$@_C \langle\pi\rangle (P0 \vee P1) \Rightarrow [\![\mathsf{callElevator0}]\!] \big(@_A \mathbf{L} (\mathsf{C\_at\_P0}) \vee [\![\mathsf{moveCabin0}]\!] @_A \mathbf{L} (\mathsf{C\_at\_P0})\big)$$

That is, provided that we start at a world where Cabin is at one of the platforms, if the elevator is called, then Alice either knows immediately that the cabin is at the ground platform, or she discovers this as soon as the cabin is moved.

Proposition 31. *The extension of the hybrid-logic proof system with the axiom schema in Fig. 7 yields a sound and complete axiomatization of the* ANt *logic.*

$$\Psi \Vdash \psi \qquad \textit{iff} \qquad \Psi \Vdash \psi$$

4 Concluding Remarks

In this paper, we have shown how a suite of logics can be developed through a two-stage constrained-hybridization process, providing in this way support for the specification and verification of cyber-physical system protocols modeled as actor networks (ANts) in the sense of [9]. The first stage of the hybridization 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, that is the way their structure can evolve as a result of the interactions that occur within them.

One of the main novelties of our approach is that we rely on unconventional semantic constraints, derived from the structural characteristics of actor-network states, or from the general properties of the state transitions. This results in faithful representations, at a logical level, of the way computation is performed in actor networks. 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 axiomatized 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.

Two main research directions are ongoing. The first aims to use the expressive power of our formalism to reason about security protocols in cyber-physical systems, in particular the existence of covert channels. The second aims to extend the logic to support the modeling of the transition system defined by interactions through graph transformations.

Acknowledgment. The work of D. Pavlovic was partially supported by NSF. J. Fiadeiro and A. Lopes received support from AFOSR for research visits to the University of Hawaii.

References

1. Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. Logic Journal of the IGPL **8**(3) (2000) 339–365
2. Prior, A.: Past, Present and Future. Oxford University Press (1967)
3. Braüner, T.: Hybrid logic and its Proof-Theory. Volume 37 of Applied Logic Series. Springer (2011)
4. Neves, R., Madeira, A., Martins, M.A., Barbosa, L.S.: Proof theory for hybrid(ised) logics. Science of Computer Programming **126** (2016) 73–93
5. Diaconescu, R.: Quasi-varieties and initial semantics for hybridized institutions. Journal of Logic and Computation **26**(3) (2016) 855–891
6. Găină, D.: Birkhoff style calculi for hybrid logics. Formal Aspects of Computing (in press) 1–28
7. Madeira, A., Neves, R., Barbosa, L.S., Martins, M.A.: A method for rigorous design of reconfigurable systems. Science of Computer Programming **132** (2016) 50–76
8. Martins, M.A., Madeira, A., Diaconescu, R., Barbosa, L.S.: Hybridization of institutions. In Corradini, A., Klin, B., Cîrstea, C., eds.: CALCO 2011. Volume 6859 of LNCS., Springer (2011) 283–297
9. Pavlovic, D., Meadows, C.A.: Actor-network procedures – (extended abstract). In Ramanujam, R., Ramaswamy, S., eds.: Distributed Computing and Internet Technology. Volume 7154 of LNCS., Springer (2012) 7–26
10. Latour, B.: Reassembling the Social: An Introduction to Actor-Network Theory. Oxford University Press (2005)
11. Milner, R.: The Space and Motion of Communicating Agents. CUP (2009)
12. Malinowski, G.: Many-Valued Logics. Oxford Logic Guides. Clarendon Press (1993)