Logical Support for Bike-Sharing System Design

uu, Ionu; Chiri, Claudia Elena; Lopes, Antónia; Fiadeiro, José Luiz

Published in:
From Software Engineering to Formal Methods and Tools, and Back

DOI:
10.1007/978-3-030-30985-5_10

Publication date:
2019

Document Version
Peer reviewed version

Logical Support for Bike-Sharing System Design

Ionuţ Ţuţu[1,2], Claudia Elena Chiriţă[2], Antónia Lopes[3], and José Luiz Fiadeiro[2]

[1] Simion Stoilow Institute of Mathematics of the Romanian Academy, Romania
ittutu@gmail.com
[2] Department of Computer Science, Royal Holloway University of London, UK
claudia.elena.chirita@gmail.com, jose.fiadeiro@rhul.ac.uk
[3] LASIGE and Faculdade Ciências, Universidade de Lisboa, Portugal
malopes@ciencias.ulisboa.pt

Abstract. Automated bicycle-sharing systems (BSS) are a prominent example of reconfigurable cyber-physical systems for which the locality and connectivity of their elements are central to the way in which they operate. These features motivate us to study BSS from the perspective of Actor-Network Theory – a framework for modelling cyber-physical-system protocols in which systems are networks of actors that are no longer limited to programs but can also include humans and physical artefacts. In order to support logical reasoning about information-flow properties that occur in BSS, we use a logical framework that we have recently developed for actor networks, which results from a two-stage hybridization process. The first stage corresponds to a logic that captures the locality and connectivity of actors in a given configuration of the network; the second stage corresponds to a logic of possible interactions between actors, which captures the dynamics of the system in terms of network reconfigurations. To illustrate the properties that can be checked using this framework, we provide an actor-network specification of a particular BSS, and use a recently developed tool for hybridized logics to highlight and correct an information-flow vulnerability of the system.

1 Introduction

Bike-sharing systems (BSS) facilitate urban transport by allowing users to borrow bicycles from a location and return them at the destination of their journey. The journeys are usually short, and the loan of a bicycle is conditioned by a fee.

The BSS model appeared in the 1960s and has since evolved through several generations. The last decade in the history of BSS stands out for a rapid growth: each year, they are more widespread and their number, sheer size and complexity, keep increasing. This has led to reshaping cities, and has promoted bike-sharing systems as one of the main means of urban transportation. If the first generations of BSS were vulnerable to vandalism and incorrect usage, the recent third generation has tackled issues concerning accessibility, automated payment, and bike and station distribution across cities.

We are now witnessing a major step forward through which dockless and electric bicycles feature as solutions to a more balanced distribution, more

responsible parking and more equitable use of the public space. These features have contributed to the success of the fourth-generation BSS; from the few small schemes available in 2015, users can now benefit from city-scale systems with more than 17 million bikes across the globe [21]. In fact, one may even argue that BSS have become a victim of their own success: communities and owning organisations were not prepared for such a rapid spread. From the oversupply of bike-sharing systems in China that led to vast bicycle graveyards, to frequent attacks on the cyber-security of the systems and users' privacy breaches, fourth generation BSS are facing various new problems and threats.

In this paper, inspired by Stefania Gnesi's work on bike-sharing systems [2,3] (which is just one of her extensive contributions to formal methods in software engineering), we propose to model a BSS using tools and techniques that are specific to our field of work. We develop a formal specification of a BSS using an actor-network framework based on hybrid logic that we have recently presented in [12], and show how the logics advanced therein can be used in conjunction with off-the-shelf formal-specification tools and theorem provers in order to reason about the design of a BSS. Through this, we hope to demonstrate how the development and analysis of such models can contribute towards the implementation of a viable BSS transport network. This analysis process could be part of the feasibility studies and project design processes that precede the implementation of a BSS, or part of the monitoring and evaluation that needs to be conducted throughout its operation.

The BSS *model under consideration.* The case study that we consider in this paper is simple. We focus on a fourth generation bike-sharing system operating in a city that is divided into several geographic regions (e.g., by geo-fencing). The regions of the city are connected through infrastructure elements such as roads, pathways, and bridges. To borrow a bicycle, users must connect to the system and make a request from the region where they are located. Once they collect a bicycle, users must input the destination of their journey before travelling. If, at any time during their travel, the destination region becomes full, with no free parking spaces, the system automatically offers rewards at that region. Rewards are an alternative to the usual (external) redistribution of bicycles across the city; they could be free rides or credit, and are meant to encourage nearby users to borrow bicycles from that region in order to make room for incoming bicycles. Once users reach their destination, they must secure their bicycles to physical docks or in designated parking spaces in order to end their journeys.

The specification and verification process. In the following sections, we gradually formalize the BSS presented above using notions of actor-network theory and hybrid logic. Concepts and background information about the logics used are introduced on the fly as we progress towards the full specification of the system. In Section 2, we provide an informal overview of the main concepts involved in actor networks, which we illustrate with the BSS. This is continued with the presentation of the two hybrid-logic formalisms that we use in the paper, LNC (in Section 3) and LAN (in Section 4). Then, in Section 5, we discuss the hybrid-logic

specification of the BSS. Lastly, in Section 6, we analyse the design and show that any implementation of the BSS has the following properties:

- If a region has a free dock, then no rewards are offered at that region.
- If a reward is offered at a region, then a user is expected to arrive there.

Whilst the first property is desired to hold, the second indicates a vulnerability because it discloses private information about the movement of users. We show how to address this through small changes in the design.

2 Actor networks

Actor Networks (ANTs) are a framework for modelling cyber-physical-system protocols originally proposed in [23] in the context of physical security. They are based on Latour's Actor-Network Theory [17] 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. Beyond that, ANTs make location a primary concern, which is essential for physical security as well as other protocols. This brings ANTs in line with spatial logics and frameworks that deal with the physical distribution of systems [1,6]; in contrast with those studies, which offer extensive support for topological properties, in ANTs we only record the locality of actors in relation to other actors.

The structure of actor networks. There are three major steps to follow when modelling a cyber-physical protocol using ANTs. The first one requires the identification of the structural aspects of an actor network:

- The relevant *sorts* (or *kinds*) of actors.
  For the BSS case study that we consider in this paper, we distinguish the human *users* of the system, the *bicycles* that they can use to travel, the *docks* where the bicycles can be locked, as well as the various *regions* where docks are available for users to borrow or return bicycles.
- The means through which actors can store state attributes, knowledge or data, which for simplicity we capture through *propositional symbols*.
  For instance, in our running example, we use a propositional symbol (which may be regarded as a Boolean flag) specific to regions to indicate whether users are *offered rewards* (free rides) for borrowing bicycles at a given region.
- *Channel types*, which account for the ability of actors to connect to other actors. Every channel type has a source and a target, both of which are (actor) sorts, and can capture transfer of knowledge, intent, or specific actions.
  For the BSS, we consider channels of type: *ask* through which users (the source of the channel) may request to borrow a bicycle at particular regions (the target of the channel); and *path* between regions (i.e. with the same source and target) to indicate the fact that users can travel directly between two given regions (if they choose to do so) – in other words, we use *path* to capture the physical topology of the BSS network.

Interactions in actor networks. In order to model cyber-physical-system protocols, we also need to consider the *network configurations* in which an ANt may find itself. Every such configuration gives a detailed account of:

- The exact channels that are available between every two actors – that is, the actor interconnectivity in a particular configuration.
- How each actor is *located* in relation to other actors. For instance, this can capture the fact that, at a given moment in the 'execution' of the BSS, a bicycle is locked *in* a specific dock *in* a region; or that a user is travelling *on* a specific bicycle, and that both the user and the bicycle are *in* a region.
- For every actor, the propositional symbols that are true for that actor.

Network configurations can change as a result of interactions taking place. Through an *interaction*, we identify specific conditions that should be met for a reconfiguration to occur, and we describe the effects of that reconfiguration. For the BSS, we consider four kinds of interactions, which correspond to:

- *borrowing/taking* a bicycle from a dock (in a region);
- *travelling* (on a bicycle) between two regions connected by a path;
- *returning* a bicycle (to a dock) once the destination is reached; and
- *offering rewards* (which is done automatically by the system) at full regions.

As an example of the conditions associated with interactions, we assume that a user can borrow a bicycle only if both the user and the bicycle are in the same region; moreover, the user must have requested to travel from that region, and the bicycle should be available in a dock in that region, where it is locked.

Specification for actor networks. Building on the above ingredients, the third major step in the development of an ANt-based design consists in describing the structure and the dynamics of the system by means of axioms written in a hybrid-logic language that is suitable for actor networks.

Through such axioms, we specify which configurations are well defined: for example, users can return bicycles at a region only if they are physically present at that region, on their bicycles, and if there are free docks available. Moreover, we also specify how networks evolve in time (as a result of reconfigurations) and the way their evolution is related to the interactions between actors.

We formalize and describe all these steps in detail, and give concrete examples of actors, channels, configurations, interactions, etc., in the following sections.

Logical support for actor networks. The mathematical structures that support the development and analysis of ANts are provided by a kind of hybrid logic (which we have recently proposed in [12]) whose models are hierarchical Kripke structures with two layers:

1. a *base layer*, specific to the states/configurations of a network; in this case, the possible worlds correspond to the actors of the network, which are the same for all configurations, while the accessibility relations capture their locality and connectivity, which may vary from one configuration to another;

2. an *upper layer*, specific to the dynamics of a network; in this case, the possible worlds correspond to network configurations (i.e., base-layer structures), and the accessibility relations capture the transitions that are possible between configurations (i.e., discrete reconfigurations of the network).

There are several defining features of actor networks that distinguish them from ordinary two-layered Kripke structures:

- Firstly, as mentioned above, all configurations of a given network share the same underlying set of actors; this means that the cyber-physical systems modelled using actor networks are closed. However, structures can be of arbitrary size, even infinite.
- Secondly, the base-layer accessibility relation that corresponds to the locality of actors is necessarily functional and acyclic; this means that, for every configuration, the locality of actors in relation to one another is given by a forest (typically of finite depth) whose nodes are the actors of the network.
- Thirdly, every network reconfiguration is determined by a specific *interaction* between the actors at the source of that reconfiguration, where by *interaction* we mean a pre-defined set of locality and connectivity constraints.

The logic of actor networks, hereafter denoted LAN, can be obtained by way of a double constrained-hybridization process, along the lines of the original construction presented in [12]. In a nutshell, a *hybridization* of a logic, regarded as a base logic, consists in an exogenous enrichment of that logic (in the sense of [20]) with features that are characteristic of hybrid logic (see [4,5]); this is done both at the syntactic level – by introducing nominals, modalities, and hybrid-logic operators – and at the semantic level, through Kripke structures whose possible worlds are labelled with models of the base logic.

In many cases, this process is *constrained*, in the sense that the Kripke structures of the resulting hybridized logic are subject to additional semantic constraints. For instance, the base-logic models that label the possible worlds of a Kripke structure may share certain information (through what are usually referred to as *rigidity constraints*); or some of the accessibility relations may be required to be reflexive, preorders or, moreover, equivalences, as in the T, S4, and S5 variants, respectively, of hybrid propositional logic.

As this short introduction to hybridization suggests, the process is applicable to a broad spectrum of logics, and its result is not a single, definite logical system, but a class of logical systems, where variations arise from tuning hybridization parameters such as the precise hybrid features added to the base logic, or the semantic constraints imposed on Kripke structures. This idea is explored in [9,19], where hybridization is formalized in the context of Goguen and Burstall's theory of institutions [15].[4] Actor-network logic(s) can be developed much in the same way: the main parameter is the base logical system, which by hybridization gives rise to a formalism for reasoning about the base layer of actor networks; and

[4] The papers [13] and [11] can be regarded as precursors of [19]; both deal with enriching abstract logics – one with temporal features, and the other with modal features.

by hybridizing that logic further, we obtain an even richer logical system that provides support for both levels of actor networks.

Despite the logic-independent nature of the construction, for the purpose of this contribution, and to make the paper more accessible to readers who may not be familiar with hybridization or institution theory, we choose to focus on a concrete logical system. We build this logic in two steps: first, we define a logic of network configurations, called LNC, as a hybridization of propositional logic; then, we introduce the logic of actor networks (LAN) as a hybridization of LNC.

A short comparison with the developments reported in [12] is in order. Our previous work also deals with a two-stage constrained hybridization aimed at developing logics for actor networks, but there are subtle and important differences to consider. One is that the base logical system corresponds in that case to the three-valued Łukasiewicz logic. Another is that the presentation relies heavily on the graph-theoretic notion of an *actor-network schema*, which determines an upper bound for the size of the models considered – in [12], all models are finite. Last but not least, the use of quantifiers over state variables is no longer restricted to the base layer of the logic of actor networks; instead, we can combine such quantifiers freely with any of the other sentence-building operators.

3 Network configurations

In what follows, we define the main building blocks of LNC: its *signatures* (structured collections of symbols), *models* (providing interpretations for the symbols declared in signatures), *sentences* (built from symbols declared in signatures), and *satisfaction relations* (establishing whether a property, formalized as a sentence, holds at a given model). All are interspersed with relevant fragments of the BSS specification that we are progressively building.

Definition 1. *An* LNC-signature *is a tuple* $\Sigma = \langle S, P, N, K \rangle$*, where:*

- S *is a set whose elements we call* sorts *or* kinds *of actors,*
- $P = \{P_s\}_{s \in S}$ *is an* S*-indexed family of sets of* propositional symbols*,*
- $N = \{N_s\}_{s \in S}$ *is an* S*-indexed family of sets of* actor names*, and*
- $K = \{K_{s \nrightarrow t}\}_{s,t \in S}$ *is an* $S \times S$*-indexed family of sets of* channel types*.*

BSS sorts In line with the informal description of the actor-network model[5] of the BSS from Section 2, the LNC-signature that we consider here contains the following four sorts: User, Bike, Dock, and Region.

BSS propositional symbols We also declare four propositional symbols:

travelling : User[6] to indicate if a user is travelling or not towards some region;
freeDock : Dock to indicate that there is no bicycle locked in a particular dock;

[5] Not to be confused with the formal concepts of *model* from Definitions 2 and 6.
[6] We use this colon notation to separate a propositional symbol from its sort, and also to separate a channel type from the two sorts on which it is defined.

fullRegion : Region to indicate that all docks in a region have bicycles in place;
rewardOffered : Region to capture the fact that users are offered free rides when borrowing bicycles from docks located in a particular region.

BSS actor names At this stage, we use no actor names. However, actor names may be introduced on the fly when dealing with quantified sentences – and in that case they are sourced from variables. In LNC, a *state variable* (or *variable*, for short) for a signature $\Sigma = \langle S, P, N, K\rangle$ is a triple (x, s, N_s), usually denoted simply by $x : s$, where x is the name of the variable, and s is its sort.[7] Variables determine extensions of signatures as follows: for every S-sorted set X of Σ-variables, $\Sigma[X] = \langle S, P, N \cup X, K\rangle$ is an LNC-signature that includes Σ.

BSS channel types We consider five channel types:

ask : User → Region to signal a user's intent to borrow a bicycle at a given region;
choose : User → Bike to capture the selection of a bicycle (to borrow) by a user;
choose : User → Dock for the selection of a dock (to return a bicycle) by a user;
travelTo : User → Region to specify the region towards which a user is travelling;
path : Region → Region to indicate that two regions are connected by a path.

The models of an LNC-signature Σ are Kripke structures that interpret the actor names in Σ as possible worlds, and the channel types as relations on worlds.

Definition 2. *Let $\Sigma = \langle S, P, N, K\rangle$ be an* LNC*-signature. A* model*, or* Kripke structure*, of Σ is a triple $\langle A, \lhd, M\rangle$, where $\langle A, \lhd\rangle$ is a* Kripke frame *defining, for every two sorts $s, t \in S$, actor name $n \in N_s$, and channel type $\kappa \in K_{s\to t}$:*

- *a set A_s of* possible worlds*, or* actors*, of sort s,*
- *an element $A_{s,n}$ of A_s, i.e., an actor of sort s corresponding to the name n,*
- *an* accessibility relation*, or* channel*, $A_{s\to t,\kappa}$ between the sets A_s and A_t,*
- *a functional and acyclic relation $\lhd$ on $\biguplus\{A_s \mid s \in S\}$ that captures the locality of actors, and for which $a \lhd a'$ reads as a* is in/on/at *a',*

and M is a family of sets $M_{s,a} \subseteq P_s$ (the propositions of sort s that hold at a) indexed by sorts $s \in S$ and actors $a \in A_s$. When there is no risk of confusion, we may drop the sorts from the notations of $A_{s,n}$, $A_{s\to t,\kappa}$, and $M_{s,a}$.

Note the properties of the second component of an LNC-model: $\lhd$ is a special kind of accessibility relation that is used here to capture the placement (location) hierarchy on actors. Equivalently, it could be presented as a (rooted) forest structure over the set of all actors. We denote its inverse by $\rhd$.

A typical example of a Kripke structure for the LNC-signature of the BSS can be seen in Figure 1. We use graphical representations of LNC-models to make the correspondence between these semantic structures and the actual configurations of the BSS easier to perceive. The model is finite; it consists of six actors, depicted using circles, which are distributed and related as follows:

[7] We annotate the variables of sort s with the set of actor names of sort s in order to ensure that there are no accidental clashes between variables and actors names.

- $A_{\mathsf{User}} = \{\mathsf{U}\}$, $A_{\mathsf{Bike}} = \{\mathsf{B}\}$, $A_{\mathsf{Dock}} = \{\mathsf{D1}, \mathsf{D2}\}$, and $A_{\mathsf{Region}} = \{\mathsf{R1}, \mathsf{R2}\}$;
- the channels are represented using labelled arrows, and they are all singletons in this case; for instance, $A_{\mathsf{ask}} = \{(\mathsf{U}, \mathsf{R1})\}$ and $A_{\mathsf{path}} = \{(\mathsf{R1}, \mathsf{R2})\}$;
- the relation $\lhd$ is depicted through the nesting of nodes; we have, for example, $\mathsf{U} \lhd \mathsf{R1}$ to capture the fact that U is currently in the region R1, $\mathsf{D1} \lhd \mathsf{R1}$ to capture the fact that D1 is a dock in that region, and $\mathsf{B} \lhd \mathsf{D1}$ to capture the fact that the bicycle B is currently locked in D1;
- the propositions that hold at given actors are indicated in a more coded way, by symbolic decorations placed on their corresponding circles; we use the symbols $>$ for travelling, $-$ for freeDock, $+$ for fullRegion, and $\star$ for rewardOffered; for instance, $M_{\mathsf{R1}} = \{\mathsf{rewardOffered}\}$, and $M_{\mathsf{D2}} = \{\mathsf{freeDock}\}$.

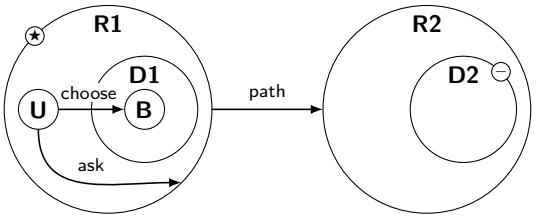

Fig. 1. An LNC-model for the BSS

The fact that the actors of a network are sorted has important consequences on the way we define the sentences of the logic of network configurations.

Definition 3. *Consider an* LNC*-signature* $\Sigma = \langle S, P, N, K \rangle$ *and let* s *be a sort in* S*. The* sentences of sort s *over* Σ *are defined by the following grammar:*

$$\varphi ::= p \mid n \mid \neg\, \varphi \mid \varphi \to \varphi \mid @_{n'}\, \varphi' \mid \langle \kappa \rangle\, \varphi^* \mid \langle \pi \rangle\, \varphi^\dagger \mid \exists X \cdot \varphi^\S$$

where $p \in P_s$ *and* $n \in N_s$ *are propositional symbols and actor names of sort* s*,* $n' \in N_{s'}$ *is an actor name (of any sort* s'*),* φ' *is a* Σ*-sentence of sort* s'*,* $\kappa \in K_{s \to s^*}$ *is a channel type with source sort* s *and target sort* s^**,* φ^* *is a* Σ*-sentence of sort* s^**,* π *is a distinguished and new* parent *modality,* $\varphi^\dagger$ *is a* Σ*-sentence (of any sort),* X *is a finite set of* Σ*-variables, and* $\varphi^\S$ *is a* $\Sigma[X]$*-sentence of sort* s*.*

Other propositional connectives such as 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, as well as the universal quantifier over state variables can also be defined in the conventional way:

$$[\kappa]\, \varphi = \neg \langle \kappa \rangle \neg\, \varphi \qquad [\pi]\, \varphi = \neg \langle \pi \rangle \neg\, \varphi \qquad \forall x \cdot \varphi = \neg\, \exists x \cdot \neg\, \varphi$$

The satisfaction relation between LNC-models and sentences is defined, as for many logical systems, by induction on the structure of sentences, and is parameterized by actors (i.e., possible worlds of the Kripke structures considered).

Definition 4. *Let $\langle A, \lhd, M\rangle$ be a Σ-model and a an actor of sort s in A. Then:*

- $\langle A, \lhd, M\rangle \vDash^a p$ *if* $p \in M_a$, *when p is a propositional symbol of sort s;*
- $\langle A, \lhd, M\rangle \vDash^a n$ *if* $a = A_n$, *when n is an actor name of sort s;*
- $\langle A, \lhd, M\rangle \vDash^a \neg\,\varphi$ *if* $\langle A, \lhd, M\rangle \nvDash^a \varphi$;
- $\langle A, \lhd, M\rangle \vDash^a \varphi_h \to \varphi_c$ *if* $\langle A, \lhd, M\rangle \vDash^a \varphi_h$ *implies* $\langle A, \lhd, M\rangle \vDash^a \varphi_c$;
- $\langle A, \lhd, M\rangle \vDash^a @_{n'}\,\varphi'$ *if* $\langle A, \lhd, M\rangle \vDash^{a'} \varphi'$, *where* $a' = A_{n'}$;
- $\langle A, \lhd, M\rangle \vDash^a \langle\kappa\rangle\,\varphi^*$ *if there exists* $(a, a^*) \in A_\kappa$ *such that* $\langle A, \lhd, M\rangle \vDash^{a^*} \varphi^*$;
- $\langle A, \lhd, M\rangle \vDash^a \langle\pi\rangle\,\varphi^\dagger$ *if there exists* $a^\dagger \rhd a$ *such that* $\langle A, \lhd, M\rangle \vDash^{a^\dagger} \varphi^\dagger$; *this implicitly means that the actor $a^\dagger$ is of the same sort as $\varphi^\dagger$;*
- $\langle A, \lhd, M\rangle \vDash^a \exists X \cdot \varphi^\S$ *if there is a $\Sigma[X]$-expansion $\langle A^\S, \lhd, M\rangle$ of $\langle A, \lhd, M\rangle$ such that $\langle A^\S, \lhd, M\rangle \vDash^a \varphi^\S$, where by* expansion *of $\langle A, \lhd, M\rangle$ we mean a $\Sigma[X]$-model that interprets all symbols in Σ in the same way as $\langle A, \lhd, M\rangle$.*

Notice that the satisfaction relation is also sorted: we evaluate LNC*-sentences φ of sort s only at actors whose sort is s. Given such a sentence φ of sort s, we write $\langle A, \lhd, M\rangle \vDash \varphi$ when $\langle A, \lhd, M\rangle \vDash^a \varphi$ for all actors $a \in A_s$.*

As an example, consider the following sentence, which is satisfied by the BSS model in Figure 1. Concerning the parsing of sentences, we rely on the usual precedence rules (e.g. unary sentence-building operators have a higher precedence than binary operators) and on parentheses to make sure no ambiguities arise.

$$\exists\{u\colon \mathsf{User}; b\colon \mathsf{Bike}; d\colon \mathsf{Dock}; r\colon \mathsf{Region}\}\cdot$$
$$@_u\,\big(\underbrace{\langle\pi\rangle\,(r \wedge \mathsf{rewardOffered}))}_{(1)} \wedge \underbrace{\langle\mathsf{ask}\rangle\, r}_{(2)} \wedge \underbrace{\langle\mathsf{choose}\rangle\,(b \wedge \langle\pi\rangle\,(d \wedge \langle\pi\rangle\, r)))}_{(3)}\big)$$

Intuitively, this sentence captures situations where a user (indicated by the operator $@_u$) is in a region r (in the sense that u and r are connected through the parent modality) where free rides are offered (as per part *1* of the sentence); the user u has requested to travel from that region (part *2* of the sentence), and has selected a bicycle available there (part *3* of the sentence).

4 Network dynamics

The signatures, models, and sentences of the logic of actor networks (LAN) are obtained through a hybridization of LNC. In this case, we add nominals that identify initial configurations of networks, and *interactions* that act as modalities and are interpreted as transition relations between configurations.

By *interaction* for an LNC-signature Σ we mean a pair consisting of a *name* ι and an existentially quantified sentence $\exists X \cdot \varphi$ over Σ such that φ is quantifier-free. Intuitively, the sentence φ describes specific locality and connectivity relationships between the actors named in X (such as the fact that various regions of the BSS may be connected by paths). We usually denote interactions by $\iota\colon \exists X \cdot \varphi$.

Definition 5. *A* LAN-signature *is a tuple* $\Omega = \langle S, P, N, K, I, \Lambda \rangle$*, where:*

- $\langle S, P, N, K \rangle$ *is an* LNC*-signature,*
- I *is a set of* (names of) initial configurations*, and*
- Λ *is a set of* interactions *(with distinct names) for* $\langle S, P, N, K \rangle$*.*

In the context of the BSS, we extend the signature presented in Section 3 by adding a single name init for initial configurations, and four interactions. For each interaction, we also present a graphical representation (LNC-model) in Figure 2.

Take: $\exists\{u : \mathsf{User}; b : \mathsf{Bike}; d : \mathsf{Dock}; r : \mathsf{Region}\} \cdot \varphi$
where $\varphi = @_u \left(\langle \pi \rangle\, r \wedge \langle \mathsf{ask} \rangle\, r \wedge \langle \mathsf{choose} \rangle\, b \right) \wedge @_b \left\langle \pi \right\rangle \left(d \wedge \langle \pi \rangle\, r \right)$

This means that users can begin their journeys at a region only if they have requested to travel from that region (through an ask channel) and there are bicycles available there that they could borrow (through a choose channel).

Travel: $\exists\{u : \mathsf{User}; b : \mathsf{Bike}; r_1, r_2 : \mathsf{Region}\} \cdot \varphi$
where $\varphi = @_u \left(\mathsf{travelling} \wedge \langle \pi \rangle \left(b \wedge \langle \pi \rangle\, r_1 \right) \right) \wedge @_{r_1} \langle \mathsf{path} \rangle\, r_2$

That is, travellers can move only between regions that are connected by paths, and in order to do so, they must use bicycles.

Return: $\exists\{u : \mathsf{User}; b : \mathsf{Bike}; d : \mathsf{Dock}; r : \mathsf{Region}\} \cdot \varphi$
where $\varphi = @_u \left(\langle \pi \rangle \left(b \wedge \langle \pi \rangle\, r \right) \wedge \langle \mathsf{travelTo} \rangle\, r \wedge \langle \mathsf{choose} \rangle \left(d \wedge \mathsf{freeDock} \wedge \langle \pi \rangle\, r \right) \right)$

This means that, in order to return a bicycle, a traveller should be in control of that bicycle and should have reached already the destination; moreover, there should be a free dock available at that region (to lock the bicycle).

Reward: $\exists\{u : \mathsf{User}; r : \mathsf{Region}\} \cdot @_u \langle \mathsf{travelTo} \rangle \left(r \wedge \mathsf{fullRegion} \right)$

That is, in order to pre-emptively free some of the docks in a region, that region should be full, and there should be a user travelling towards it.

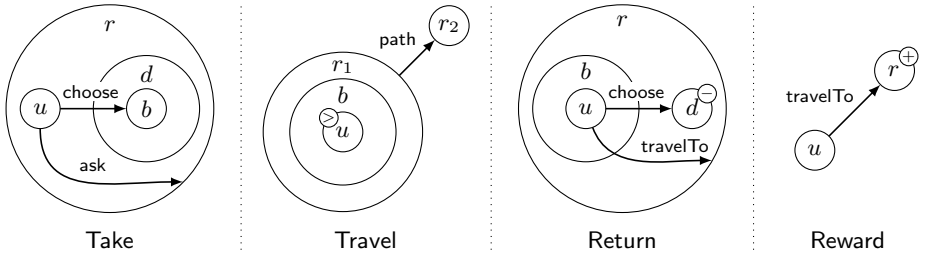

Fig. 2. Graphical representations of the four BSS interactions

Similarly to LNC, the models of a LAN-signature Ω are also Kripke structures, but the nominals are interpreted as configurations (i.e., LNC-models) and the interactions as transition relations between configurations.

Definition 6. *Consider a* LAN*-signature* $\Omega = \langle S, P, N, K, I, \Lambda \rangle$*. A* Kripke model $\langle D, C \rangle$ *of* Ω *consists of a* domain $|D|$*, i.e., a plain set of* worlds*, together with*

- *a possible world* $D_i \in |D|$ *for each* configuration name $i \in I$*,*
- *a transition relation* $D_\iota \subseteq |D| \times |D|$ *for each* interaction name ι *in* Λ*,*

and a family of $\langle S, P, N, K \rangle$*-models* $C_w = \langle A_w, \lhd_w, M_w \rangle$*, for* $w \in |D|$*, such that*

- *for all possible worlds* $w, w' \in |D|$ *and sorts* $s \in S$ *we have* $(A_w)_s = (A_{w'})_s$*;*
- *for every possible world* $w \in |D|$ *and interaction* $\iota \colon \exists X \cdot \varphi$ *in* Λ*, there exists a transition* $(w, w') \in D_\iota$ *if and only if* $C_w \vDash \exists X \cdot \varphi$*.*

When there is no risk of confusion, we may also denote $(w, w') \in D_\iota$ *by* $w \xrightarrow{\iota} w'$*.*

Figure 3 depicts a model for the BSS that corresponds to the journey of a user from one region (R1) to another (R2). This model has four possible worlds, C0 – C3, of which C0 is the interpretation of init; and it has three transitions, one for each of the following interactions: Take, Travel, and Return.

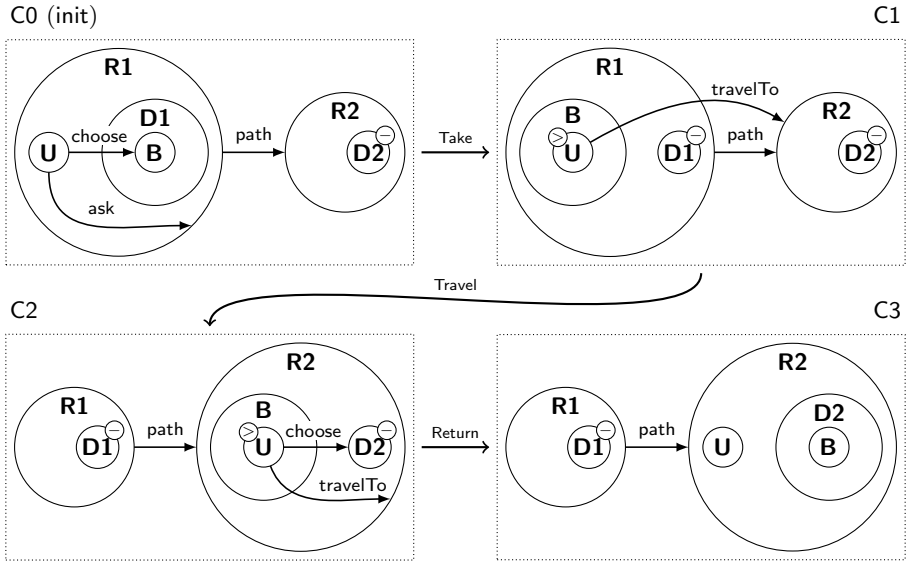

Fig. 3. A possible LAN-model for the BSS

In order to define the syntax of the logic of actor networks, we consider a different kind of extension of a signature: not only with state variables, but with network variables as well. In LAN, a *network variable* for a signature $\Omega = \langle S, P, N, K, I, \Lambda \rangle$ is a pair (y, I), where y is the name of the variable.[8] For any S-sorted set X of state variables for $\langle S, P, N, K \rangle$ and any set Y of network variables for Ω, we obtain the extended signature $\Omega[X; Y] = \langle S, P, N \cup X, K, I \cup Y, \Lambda \rangle$.

[8] Similarly to the logic LNC, we annotate LAN-variables with the set of configuration names in order to avoid accidental name clashes when building the extension.

Definition 7. *The* LAN-sentences *over* $\Omega = \langle S, P, N, K, I, \Lambda \rangle$ *are given by:*

$$\psi ::= \varphi \mid i \mid \neg\psi \mid \psi \Rightarrow \psi \mid i : \psi \mid \langle\!|\iota|\!\rangle\, \psi \mid \exists X; Y \cdot \psi'$$

where φ *is an* LNC*-sentence over* $\langle S, P, N, K \rangle$*,* $i \in I$ *is a configuration name,* ι *is an interaction name in* Λ*,* X *and* Y *are finite sets of state and network variables, respectively, and* ψ' *is a* LAN*-sentence over the extended signature* $\Omega[X; Y]$*.*

For quantified sentences, when the set Y *is empty, we also write* $\exists X \cdot \psi'$ *in place of* $\exists X; Y \cdot \psi'$*. Similarly, when* X *is empty, we write* $\exists Y \cdot \psi'$*.*

Notice that, similarly to LNC-sentences, LAN-sentences are built using hybrid-logic operators, but in this case we use a distinct double-symbol notation; moreover, the local-satisfaction operators (represented as $@_n$ in LNC) are denoted here by means of a colon. We extend the use of this notation to other Boolean connectives ($\mathbb{\wedge}$, $\mathbb{\vee}$, $\mathbb{\Leftrightarrow}$), to the dual modal operators ($[\![\_]\!]$), and to the universal quantifier ($\mathbb{\forall}$), which are defined as in Section 3.

Definition 8. *Let* $\langle D, C \rangle$ *be a* LAN*-model for a signature* $\Omega = \langle S, P, N, K, I, \Lambda \rangle$*, and* $w \in D$ *a possible worlds. The* local satisfaction *of* Ω*-sentences by* $\langle D, C \rangle$ *at* w *is defined by structural induction, as follows:*

- $\langle D, C \rangle \Vdash^w \varphi$ *if* $C_w \vDash \varphi$*, when* φ *is an* LNC*-sentence over* $\langle S, P, N, K \rangle$*;*
- $\langle D, C \rangle \Vdash^w i$ *if* $w = D_i$*, when* i *is a configuration name;*
- $\langle D, C \rangle \Vdash^w \neg\psi$ *if* $\langle D, C \rangle \nVdash^w \psi$*;*
- $\langle D, C \rangle \Vdash^w \psi_h \Rightarrow \psi_c$ *if* $\langle D, C \rangle \Vdash^w \psi_h$ *implies* $\langle D, C \rangle \Vdash^w \psi_c$*;*
- $\langle D, C \rangle \Vdash^w i : \psi$ *if* $\langle D, C \rangle \Vdash^{w'} \psi$*, where* $w' = D_i$*;*
- $\langle D, C \rangle \Vdash^w \langle\!|\iota|\!\rangle\, \psi$ *if there exists a transition* $w \xrightarrow{\iota} w'$ *such that* $\langle D, C \rangle \Vdash^{w'} \psi$*;*
- $\langle D, C \rangle \Vdash^w \exists X; Y \cdot \psi'$ *if there exists a* $\Omega[X; Y]$*-expansion* $\langle D', C' \rangle$ *of* $\langle D, C \rangle$ *such that* $\langle D', C' \rangle \Vdash^w \psi'$*, where by* expansion *of* $\langle D, C \rangle$ *we mean a* $\Omega[X; Y]$*-model* $\langle D', C' \rangle$ *such that* (a) D' *has the same domain as* D *and gives the same interpretation as* D *for configuration names and interactions in* Ω*, and* (b) *for every world* $w \in |D'|$*,* C'_w *is an* $\langle S, P, N \cup X, K \rangle$*-expansion of* C_w*.*

We write $\langle D, C \rangle \Vdash \psi$ *when* $\langle D, C \rangle \Vdash^w \psi$ *for all possible worlds* $w \in |D|$*.*

The definition above justifies the use of different notations for the sentence-building operators of LAN. Formally, the relationship between the two kinds of operators is described in the list below, where $\langle D, C \rangle$ is an Ω-model with non-empty sets of actors for each sort, $w \in |D|$ is a possible world, φ, φ_h, φ_c, φ_1, φ_2 are $\langle S, P, N, K \rangle$-sentences, and φ' is an $\langle S, P, N \cup X, K \rangle$-sentence.

- $\langle D, C \rangle \Vdash^w \neg\varphi$ only if $\langle D, C \rangle \Vdash^w \neg\varphi$
- $\langle D, C \rangle \Vdash^w \varphi_h \to \varphi_c$ only if $\langle D, C \rangle \Vdash^w \varphi_h \Rightarrow \varphi_c$
- $\langle D, C \rangle \Vdash^w \varphi_1 \wedge \varphi_2$ if and only if $\langle D, C \rangle \Vdash^w \varphi_1 \mathbb{\wedge} \varphi_2$
- $\langle D, C \rangle \Vdash^w \varphi_1 \vee \varphi_2$ if $\langle D, C \rangle \Vdash^w \varphi_1 \mathbb{\vee} \varphi_2$

- $\langle D, C\rangle \Vdash^w \varphi_1 \leftrightarrow \varphi_2$ only if $\langle D, C\rangle \Vdash^w \varphi_1 \Leftrightarrow \varphi_2$
- $\langle D, C\rangle \Vdash^w \exists X \cdot \varphi'$ if $\langle D, C\rangle \Vdash^w \exists\!\!\!\exists X \cdot \varphi'$
- $\langle D, C\rangle \Vdash^w \forall X \cdot \varphi'$ if and only if $\langle D, C\rangle \Vdash^w \forall\!\!\!\forall X \cdot \varphi'$

Most of the above properties are one-way implications ('if', or 'only if'), because the global satisfaction of LNC-sentences relies on an implicit universal quantification (over actors). Equivalences are guaranteed to hold only for those sentence-building operators that commute with universal quantifiers.

5 On the design of a bike-sharing system

The two logical systems presented in this paper, LNC and LAN, enable us to give a detailed formal account of the way in which bike-sharing systems operate, hence opening the possibility for formal verification (using suitable proof techniques) at a later stage of development. To that end, in this section we discuss a formal specification of the BSS. By *formal specification* we mean a pair $\langle \Omega, \Gamma\rangle$, where Ω is a LAN-signature and Γ is a finite set of LAN-sentences over Ω. The actual bike-sharing networks defined by $\langle \Omega, \Gamma\rangle$ correspond to the Ω-models that satisfy all sentences in Γ. A specification of this kind is commonly known as a flat, or unstructured, specification. There are several ways to build structured specifications from these; for that purpose, various modularization techniques have been explored in the literature, especially around the notion of institution (see, e.g., the monograph [25]) – and they can be used in conjunction with LNC and LAN without special intervention. However, for the purpose of this work, and for simplicity, we describe the specification as a plain set of sentences over the LAN-signature for the BSS discussed in Section 4.

We consider three categories of sentences: *(a)* sentences that ensure that the base-layer Kripke structures (LNC-models) are actual, well-defined configurations of the BSS; *(b)* sentences that define the potential initial configurations of the BSS; *(c)* sentences that describe the effects that interactions have on the structure and properties of the configurations; and *(d)* sentences that deal with the frame problem[9] by specifying non-effects of the interactions in terms of attributes of the configurations that are preserved or reflected along transitions. For each category, we discuss below a few important examples. The full specification of the BSS is available in [27] through the repository engine Ontohub [8].

In order to make the following LNC and LAN-sentences easier to read, we make explicit all quantifiers over state variables; cf. Definition 8, where the satisfaction of a LNC-sentence at a world is implicitly quantified over actors of suitable sort.

Well-defined configurations The sentences in this category concern the locality and connectivity of the actors, as well as the way in which these structural aspects are related to propositional attributes such as travelling and rewardOffered.

[9] This problem is notorious for axiomatizing the way in which states change when an event occurs; various solutions have been proposed in connection to formalisms such as the situation calculus, event calculus, or default logic, among others; see, e.g. [26].

Take, for instance, the following two LNC-sentences, which ensure that at most one traveller can use a bicycle at a given time (1), and that, in any configuration, at most one bicycle can be locked in one of the docks (2).

$$\forall\{u_1, u_2 \colon \mathsf{User}; b \colon \mathsf{Bike}\} \cdot @_{u_1} \langle\pi\rangle\, b \wedge @_{u_2} \langle\pi\rangle\, b \to @_{u_1}\, u_2 \tag{1}$$

$$\forall\{b_1, b_2 \colon \mathsf{Bike}; d \colon \mathsf{Dock}\} \cdot @_{b_1} \langle\pi\rangle\, d \wedge @_{b_2} \langle\pi\rangle\, d \to @_{b_1}\, b_2 \tag{2}$$

In regard to connectivity, there is a special relationship between the channels of type $\mathsf{ask} \colon \mathsf{User} \to \mathsf{Region}$ and those of type $\mathsf{choose} \colon \mathsf{User} \to \mathsf{Bike}$. The aim is to ensure that whenever a user requests to travel from a region (through a channel of type ask), if there is a bicycle available in that region (i.e., a bicycle locked in a dock, and not claimed by another traveller), then the user can obtain a bicycle (perhaps even that bicycle) through a choose channel.

We specify this requirement in two steps: first, in (3) and (4) we provide preconditions for the existence of ask and choose channels; then, in (5) we describe choose as an injective partial mapping from users to bicycles (all in the same region), and in (6) we ensure that the interpretation of choose is maximal.

$$\forall\{u \colon \mathsf{User}; r \colon \mathsf{Region}\} \cdot @_u\, (\langle\mathsf{ask}\rangle\, r \to \langle\pi\rangle\, r) \tag{3}$$

$$\forall\{u \colon \mathsf{User}; b \colon \mathsf{Bike}\} \cdot @_u\, (\langle\mathsf{choose}\rangle\, b \to \exists\{d \colon \mathsf{Dock}; r \colon \mathsf{Region}\} \cdot (\langle\mathsf{ask}\rangle\, r \wedge @_b\, \langle\pi\rangle\, (d \wedge \langle\pi\rangle\, r))) \tag{4}$$

$$\forall\{u_1, u_2 \colon \mathsf{User}; b_1, b_2 \colon \mathsf{Bike}\} \cdot @_{u_1}\, \langle\mathsf{choose}\rangle\, b_1 \wedge @_{u_2}\, \langle\mathsf{choose}\rangle\, b_2 \to (@_{u_1}\, u_2 \leftrightarrow @_{b_1}\, b_2) \tag{5}$$

$$\forall\{u \colon \mathsf{User}; b \colon \mathsf{Bike}; d \colon \mathsf{Dock}; r \colon \mathsf{Region}\} \cdot (@_u\, \langle\mathsf{ask}\rangle\, r \to \exists\{b_1 \colon \mathsf{Bike}\} \cdot @_u\, \langle\mathsf{choose}\rangle\, b_1) \vee (@_b\, \langle\pi\rangle\, (d \wedge \langle\pi\rangle\, r) \to \exists\{u_1 \colon \mathsf{User}\} \cdot @_{u_1}\, \langle\mathsf{choose}\rangle\, b) \tag{6}$$

There is a similar relationship between the channels of type $\mathsf{travelTo} \colon \mathsf{User} \to \mathsf{Region}$ and the channels of type $\mathsf{choose} \colon \mathsf{User} \to \mathsf{Dock}$. For space considerations, and because the entire specification is available in [27], we do not present it explicitly here.

For propositional attributes, we consider the following characterizations:

$$\forall\{u \colon \mathsf{User}\} \cdot @_u\, \mathsf{travelling} \leftrightarrow \exists r \colon \mathsf{Region} \cdot @_u\, \langle\mathsf{travelTo}\rangle\, r \tag{7}$$

$$\forall\{d \colon \mathsf{Dock}\} \cdot @_d\, \mathsf{freeDock} \leftrightarrow \neg\, \exists b \colon \mathsf{Bike} \cdot @_b\, \langle\pi\rangle\, d \tag{8}$$

$$\forall\{r \colon \mathsf{Region}\} \cdot @_r\, \mathsf{fullRegion} \leftrightarrow \forall d \colon \mathsf{Dock} \cdot @_d\, (\langle\pi\rangle\, r \to \neg\, \mathsf{freeDock}) \tag{9}$$

Initiality constraints The only restriction that we impose on the interpretation of init is that no user is travelling at that configuration, and no reward is offered.

$$\mathsf{init} : (\neg\, \exists\{u \colon \mathsf{User}\} \cdot @_u\, \mathsf{travelling} \wedge \neg\, \exists\{r \colon \mathsf{Region}\} \cdot @_r\, \mathsf{rewardOffered}) \tag{10}$$

Interaction effects To axiomatize the effects of interactions, we make use of LAN-sentences of the form $\psi_h \Rightarrow \langle\!\langle \iota \rangle\!\rangle\, \psi_c$ or $\psi_h \Rightarrow [\![\iota]\!]\, \psi_c$, where ι is an interaction, ψ_h is a precondition for ι, and ψ_c is postcondition for ι – which holds either at one

of the configurations reached through a ι-transition, or at all such configurations, depending on whether we use the *possibility* or the *necessity* operator.

$$\begin{aligned}
&\forall\{u\colon \mathsf{User}; b\colon \mathsf{Bike}; d\colon \mathsf{Dock}; r\colon \mathsf{Region}\} \cdot \\
&\quad @_u\,(\langle\pi\rangle\, r \wedge \langle \mathsf{ask}\rangle\, r \wedge \langle \mathsf{choose}\rangle\, b) \wedge @_b\,\langle\pi\rangle\,(d \wedge \langle\pi\rangle\, r)^{10} \\
&\quad \Rightarrow [\![\mathsf{Take}]\!]\, @_u\,(\langle\pi\rangle\,(b \wedge \langle\pi\rangle\, r) \wedge \exists\{r_1\colon \mathsf{Region}\} \cdot \langle \mathsf{travelTo}\rangle\, r_1) \qquad (11)\\
&\forall\{u\colon \mathsf{User}; b\colon \mathsf{Bike}; r_1, r_2\colon \mathsf{Region}\} \cdot \\
&\quad @_u\,(\mathsf{travelling} \wedge \langle\pi\rangle\,(b \wedge \langle\pi\rangle\, r_1)) \wedge @_{r_1}\,\langle \mathsf{path}\rangle\, r_2 \\
&\quad \Rightarrow \langle\!|\mathsf{Travel}|\!\rangle\, @_u\,\langle\pi\rangle\,(b \wedge \langle\pi\rangle\, r_2) \qquad (12)\\
&\forall\{u\colon \mathsf{User}; b\colon \mathsf{Bike}; d\colon \mathsf{Dock}; r\colon \mathsf{Region}\} \cdot \\
&\quad @_u\,(\langle\pi\rangle\,(b \wedge \langle\pi\rangle\, r) \wedge \langle \mathsf{travelTo}\rangle\, r \wedge \langle \mathsf{choose}\rangle\,(d \wedge \mathsf{freeDock} \wedge \langle\pi\rangle\, r)) \\
&\quad \Rightarrow [\![\mathsf{Return}]\!]\,(@_b\,\langle\pi\rangle\,(d \wedge \langle\pi\rangle\, r) \wedge @_u\,(\langle\pi\rangle\, r \wedge \neg\, \mathsf{travelling})) \qquad (13)\\
&\forall\{r\colon \mathsf{Region}\} \cdot \exists\{u\colon \mathsf{User}\} \cdot @_u\,\langle \mathsf{travelTo}\rangle\,(r \wedge \mathsf{fullRegion}) \\
&\quad \Rightarrow [\![\mathsf{Reward}]\!]\, @_r\, \mathsf{rewardOffered} \qquad (14)
\end{aligned}$$

Consider, for instance, the sentence 12 describing the effects of the interaction Travel. By the definition of Travel (on page 10), we know that the interaction can lead to a change of any configuration where a traveller u, currently in a region r_1, can follow a path in order to reach another region, r_2. The axiomatization of Travel ensures that there exists indeed a reconfiguration of the system (as per the semantics in LAN of the *possibility* operator) such that u reaches the region r_2.

The frame problem The sentences(11)–(14) above capture successfully the direct effects that the interactions may have, but convey no information about their non-effects. For example, Travel does not affect the relative locality of bicycles with respect to the docks. We specify this property in (15) and (16).

$$\forall\{b\colon \mathsf{Bike}; d\colon \mathsf{Dock}\} \cdot @_b\,\langle\pi\rangle\, d \Rightarrow [\![\mathsf{Travel}]\!]\, @_b\,\langle\pi\rangle\, d \qquad (15)$$

$$\forall\{b\colon \mathsf{Bike}; d\colon \mathsf{Dock}\} \cdot \langle\!|\mathsf{Travel}|\!\rangle\, @_b\,\langle\pi\rangle\, d \Rightarrow @_b\,\langle\pi\rangle\, d \qquad (16)$$

A considerable number of other trivial sentences of this kind need to be added to the BSS specification; see [27]. In addition, some of the sentences describing non-effects are necessarily conditional. For instance, the reward offered at a region is preserved by Take if no dock in that region can be freed by the interaction.

$$\begin{aligned}
&\forall\{r\colon \mathsf{Region}\} \cdot @_r\, \mathsf{rewardOffered} \\
&\quad \wedge \neg\, \exists\{u\colon \mathsf{User}; b\colon \mathsf{Bike}; d\colon \mathsf{Dock}\} \cdot @_u\,\langle \mathsf{choose}\rangle\,(b \wedge \langle\pi\rangle\,(d \wedge \langle\pi\rangle\, r)) \\
&\quad \Rightarrow [\![\mathsf{Take}]\!]\, @_r\, \mathsf{rewardOffered} \qquad (17)
\end{aligned}$$

A different situation arises when considering the 'reflection' of the rewardOffered properties. If, say, a reward is offered at a region r after a Reward-transition, then

[10] Note that $\langle \mathsf{ask}\rangle\, b$ entails $\langle\pi\rangle\, b$; moreover, under the hypothesis $@_u\,\langle \mathsf{choose}\rangle\, b$, and by (3), (4), and the functionality of π, $@_u\,\langle \mathsf{ask}\rangle\, r$ and $\exists\{d\colon \mathsf{Dock}\} \cdot @_b\,\langle\pi\rangle\,(d \wedge \langle\pi\rangle\, r)$ are semantically equivalent. Still, to give a better picture of the configurations affected by Take (or by any of the other interactions), we write the precondition in full.

either the reward is also offered (at the same region r) at the source configuration of the transition (in which case, the property is preserved by the interaction), or it is generated by the transition as an effect of Reward.

$$\begin{aligned}&\Downarrow\{r\colon \mathsf{Region}\} \cdot \langle\!|\mathsf{Reward}|\!\rangle\, @_r\, \mathsf{rewardOffered}\\ &\quad \Rightarrow @_r\, \mathsf{rewardOffered} \vee \exists\{u\colon \mathsf{User}\} \cdot @_u\, \langle \mathsf{travelTo}\rangle\, (r \wedge \mathsf{fullRegion})\end{aligned} \tag{18}$$

It is important to highlight the fact that, under the current specification, Reward is the only interaction through which rewards can be offered at given regions. That is, for any interaction ι different from Reward, we have:

$$\Downarrow\{r\colon \mathsf{Region}\} \cdot \langle \iota \rangle\, @_r\, \mathsf{rewardOffered} \Rightarrow @_r\, \mathsf{rewardOffered} \tag{19}$$

6 Information-flow properties

Generally, in the context of actor networks, by *information flow* we refer to the transfer of information from one configuration to another as a result of a reconfiguration process. A particular application of information-flow properties is to characterize invariants. For that purpose, suppose $\langle \Omega, \Gamma \rangle$ is the actor-network specification of the BSS described in Section 5. We say that a LAN-sentence ψ over some extension $\Omega[X;Y]$ of Ω is an *invariant* for $\langle \Omega, \Gamma \rangle$ when, for every interaction ι, $\Downarrow X; Y \cdot \psi \Rightarrow [\iota]\, \psi$ is a semantic consequence of Γ – that is, when $\langle D, C \rangle \Vdash \Downarrow X; Y \cdot \psi \Rightarrow [\iota]\, \psi$ for all models $\langle D, C \rangle$ that satisfy all sentences in Γ.[11]

We analyse the following two invariants of the BSS, denoted by FD and RO.

FD : If a region has a free dock, then no reward is offered at that region.

$$\forall\{d\colon \mathsf{Dock}\} \cdot @_d\, (\mathsf{freeDock} \rightarrow [\pi]\, \neg\, \mathsf{rewardOffered}) \tag{20}$$

RO : If a reward is offered at a region, then a traveller is expected to arrive there.

$$\forall\{r\colon \mathsf{Region}\} \cdot @_r\, \mathsf{rewardOffered} \rightarrow \exists\{u\colon \mathsf{User}\} \cdot @_u\, \langle \mathsf{travelTo}\rangle\, r \tag{21}$$

To verify that the sentences $\mathsf{FD} \Rightarrow [\![\iota]\!]\, \mathsf{FD}$ and $\mathsf{RO} \Rightarrow [\![\iota]\!]\, \mathsf{RO}$ are indeed consequences of $\langle \Omega, \Gamma \rangle$, where ι is any of the four interactions of the BSS, we use a recently developed extension of the Heterogeneous Tool Set (HETS) [22], called H [7], that provides support for specification and reasoning in hybridized logics.

In particular, for reasoning purposes, H implements a verification-by-translation method based on theoretical results presented in [10]. This involves three main steps: *(1)* translating the verification problem to first-order logic using a suitable encoding of the hybridized logic, *(2)* solving the problem there using automated-theorem-proving technologies that have already been developed for first-order logic (such as SPASS [28] or Vampire [24]), and *(3)* transferring the result of the verification process back to the hybridized logic.

[11] When both X and Y are empty, we can further prove that ψ holds at all reachable configurations of the model of $\langle \Omega, \Gamma \rangle$ by verifying that Γ entails $\mathsf{init} : \psi$.

Details of this process, including intermediate results that assist the theorem provers in establishing that FD and RO are invariants, can be found in [27].

Dealing with vulnerabilities As the two invariants discussed above show, the analysis of information-flow properties can be used to validate the design of the BSS by providing formal guarantees of desired properties such as FD; but the same process can also be used to prove the existence of vulnerabilities: by RO, it follows that any implementation of the current BSS design may inadvertently disclose information about the users' whereabouts.

What we briefly demonstrate next is that we can emend the design presented in Section 5 to prevent the vulnerability described by RO while maintaining the desired property FD. For that purpose, it suffices to examine the proof of RO (as provided through the H extension of HETS), which reveals that the invariant hinges on the fact that rewards are deterministically offered by Reward-transitions, and only by Reward-transitions. This suggests that one way to correct the design is by dropping the sentences described in (19), or by replacing them with sentences that express the fact that rewards can also be offered by Take, Travel, or Return when the region is full. Intuitively, the effect of the change is that rewards may now be non-deterministically offered by any of the four interactions. As a final verification, we have checked that the change does not affect FD.

To prove that RO is no longer deducible from the specification, all we need is to find a model of the specification that does not satisfy RO. We obtain such a model from the Kripke structure depicted in Figure 3, by letting the Return-transition generate a reward at R2 in the configuration C3. This model satisfies all BSS axioms presented in Section 5 except (19), and does not satisfy RO.

7 Conclusions and further work

In this paper, we have presented an actor-network approach to the design and analysis of a bike-sharing system. We have shown how various aspects of the BSS can be formalized using a combination of two hybrid-logic formalisms: a logic of network configurations, which deals with static aspects of the BSS, and a logic of actor networks, which is defined on top of the logic of network configurations and deals with dynamic aspects of the system – i.e., the way reconfigurations occur as a result of interactions between actors.

The full specification of the BSS has been analysed using a recently developed extension of HETS that provides support for hybrid(ized) logics. This includes support for parsing, static analysis, as well as formal verification – which relies on a general encoding of hybridized logics into first-order logic. Using the toolset, we have confirmed two information-flow properties of the BSS: first, that the rewards at a region automatically cease to be offered when one of the docks in that region becomes free; and second, that all regions where rewards are offered have users travelling towards them. The latter shows that the system is vulnerable by design; we have used HETS again to identify (and then correct) the part of the specification that is responsible for the vulnerability.

Beyond the actual analysis of the BSS, one of the benefits of conducting a case study of this kind is that it confirms that the formal specification & verification tools that we have available today (thanks to a decades-long series of theoretical developments on the foundations of algebraic specification) are already well capable of dealing with complex, real-world reconfigurable systems. At the same time, it shows some of the limitations of the current technology – or, at least, of the one discussed in this paper. Currently, for the BSS, there is no way to ensure that all travellers can actually reach their destination (or that they eventually do). This is due to the limited expressive power of hybrid logics. A simple and elegant solution that we aim to pursue further is to consider dynamic-logic operators as in [16,18,14]. This raises a series of new and interesting challenges because some of the key results on hybrid logics, such as their encoding into first-order logic, cannot be generalized in a straighforward way to dynamic logics.

References

1. Bartocci, E., Bortolussi, L., Loreti, M., Nenzi, L.: Monitoring mobile and spatially distributed cyber-physical systems. In: Proceedings of the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE 2017, Vienna, Austria, September 29 – October 02, 2017. pp. 146–155. ACM (2017)
2. ter Beek, M.H., Fantechi, A., Gnesi, S.: Challenges in modelling and analyzing quantitative aspects of bike-sharing systems. In: Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change – 6th International Symposium. LNCS, vol. 8802, pp. 351–367. Springer (2014)
3. ter Beek, M.H., Gnesi, S., Latella, D., Massink, M.: Towards automatic decision support for bike-sharing system design. In: Software Engineering and Formal Methods. LNCS, vol. 9509, pp. 266–280. Springer (2015)
4. Blackburn, P.: Representation, reasoning, and relational structures: A hybrid logic manifesto. Logic Journal of the IGPL **8**(3), 339–365 (2000)
5. Braüner, T.: Hybrid Logic and its Proof-Theory, Applied Logic Series, vol. 37. Springer (2011)
6. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Model checking spatial logics for closure spaces. Logical Methods in Computer Science **12**(4) (2016)
7. Codescu, M.: Hybridisation of institutions in Hets. In: 8th Conference on Algebra and Coalgebra in Computer Science, CALCO 2019, June 3–6, 2019, London, UK (2019), Tool available at `http://imar.ro/~diacon/forver/forver.html`
8. Codescu, M., Kuksa, E., Kutz, O., Mossakowski, T., Neuhaus, F.: Ontohub: A semantic repository engine for heterogeneous ontologies. Applied Ontology **12**(3-4), 275–298 (2017)
9. Diaconescu, R.: Quasi-varieties and initial semantics for hybridized institutions. J. Log. Comput. **26**(3), 855–891 (2016)
10. Diaconescu, R., Madeira, A.: Encoding hybridized institutions into first-order logic. Mathematical Structures in Computer Science **26**(5), 745–788 (2016)
11. Diaconescu, R., Stefaneas, P.S.: Ultraproducts and possible worlds semantics in institutions. Theor. Comput. Sci. **379**(1-2), 210–230 (2007)
12. Fiadeiro, J.L., Ţuţu, I., Lopes, A., Pavlovic, D.: Logics for actor networks: A two-stage constrained-hybridisation approach. Journal of Logical and Algebraic Methods in Programming **106**, 141–166 (2019)

13. Finger, M., Gabbay, D.M.: Adding a temporal dimension to a logic system. Journal of Logic, Language and Information **1**(3), 203–233 (1992)
14. Găină, D., Ţuţu, I.: Birkhoff completeness for hybrid-dynamic first-order logic. In: Automated Reasoning with Analytic Tableaux and Related Methods – 28th International Conference, TABLEAUX 2019, London, UK, September 3–5, 2019, Proceedings. Springer (2019)
15. Goguen, J.A., Burstall, R.M.: Institutions: Abstract model theory for specification and programming. J. ACM **39**(1), 95–146 (1992)
16. Hennicker, R., Madeira, A., Knapp, A.: A hybrid dynamic logic for event/data-based systems. In: Fundamental Approaches to Software Engineering – 22nd International Conference. LNCS, vol. 11424, pp. 79–97. Springer (2019)
17. Latour, B.: Reassembling the Social: An Introduction to Actor-Network Theory. Oxford University Press (2005)
18. Madeira, A., Barbosa, L.S., Hennicker, R., Martins, M.A.: A logic for the stepwise development of reactive systems. Theor. Comput. Sci. **744**, 78–96 (2018)
19. Martins, M.A., Madeira, A., Diaconescu, R., Barbosa, L.S.: Hybridization of institutions. In: Algebra and Coalgebra in Computer Science. LNCS, vol. 6859, pp. 283–297. Springer (2011)
20. Mateus, P., Sernadas, A., Sernadas, C.: Exogenous semantics approach to enriching logics. In: Essays on the Foundations of Mathematics and Logic, Advanced Studies in Mathematics and Logic, vol. 1, pp. 165–194. Polimetrica (2005)
21. Moon-Miklaucic, C., Bray-Sharpin, A., de la Lanza, I., Khan, A., Re, L.L., Maassen, A.: The evolution of bike sharing: 10 questions on the emergence of new technologies, opportunities, and risks. Tech. rep., Washington, DC: World Resources Institute (2019), available online at `http://www.wri.org/publication/evolution-bike-sharing`
22. Mossakowski, T., Maeder, C., Lüttich, K.: The heterogeneous tool set (Hets). In: Proceedings of 4th International Verification Workshop in connection with CADE-21. vol. 259. CEUR-WS.org (2007)
23. Pavlovic, D., Meadows, C.A.: Actor-network procedures (extended abstract). In: Distributed Computing and Internet Technology. LNCS, vol. 7154, pp. 7–26. Springer (2012)
24. Riazanov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Commun. **15**(2-3), 91–110 (2002)
25. Sannella, D., Tarlecki, A.: Foundations of Algebraic Specification and Formal Software Development. Monographs in Theoretical Computer Science, an EATCS Series, Springer (2012)
26. Shanahan, M.: Solving the frame problem – a mathematical investigation of the common sense law of inertia. MIT Press (1997)
27. Ţuţu, I., Chiriţă, C., Lopes, A., Fiadeiro, J.: A hybrid-logic specification of a BSS. Ontohub (2019), `https://ontohub.org/forver/BSS.dol`
28. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Automated Deduction. LNCS, vol. 5663, pp. 140–145. Springer (2009)