Modelling and Verification of Large-Scale Sensor Network Infrastructures

Michele Sevegnani*, Milan Kabáč†, Muffy Calder* and Julie A. McCann†
*School of Computing Science, University of Glasgow, Glasgow, UK
Email: {michele.sevegnani, muffy.calder}@glasgow.ac.uk
†Department of Computing, Imperial College London, London, UK
Email: {m.kabac, j.mccann}@imperial.ac.uk

***Abstract*—Large-scale wireless sensor networks (WSN) are increasingly deployed and an open question is how they can support multiple applications. Networks and sensing devices are typically heterogeneous and evolving: topologies change, nodes drop in and out of the network, and devices are reconfigured. The key question we address is how to verify that application requirements are met, individually and collectively, and can continue to be met, in the context of large-scale, evolving network and device configurations.**

We define a modelling and verification framework based on Bigraphical Reactive Systems (BRS) for modelling, with bigraph patterns and temporal logic properties for specifying application requirements. The bigraph diagrammatic notation provides an intuitive representation of concepts such as hierarchies, communication, events and spatial relationships, which are fundamental to WSNs.

We demonstrate modelling and verification through a real-life urban environmental monitoring case-study. A novel contribution is automated online verification using BigraphER and replay of real-life sensed data streams and network events by the Cooja network simulator. Performance results for verification of two application properties running on a WSN with up to 200 nodes indicate our framework is capable of handling WSNs of that scale.

***Index Terms*—Wireless Sensor Networks; modelling; bigraphs; runtime verification; monitoring.**

I. Introduction

Large-scale wireless sensor networks (WSN) are increasingly deployed and an open question is how they can support multiple applications. These applications have to meet requirements, e.g. concerning functional behaviour, security and quality of service. But it is difficult to verify application requirements because there are large numbers of connected devices, and the network and devices are heterogeous and evolving, as topologies change due to node/network failures, dynamic application requirements, etc. The key question we address is how to assure requirements are met, individually and collectively, and can continue to be met, in the context of large-scale, evolving network and device configurations.

In contrast with small-scale environments (e.g. homes, offices), at a large-scale, it is crucial to make explicit how large numbers of devices are grouped into logical network partitions, to allow reasoning about application-specific objects of interest (e.g. building, bridge, park) rather than individual sensors or devices. This is because the requirements of applications, e.g. the sample rate of sensors (ensuring recognition accuracy), data delivery, delay tolerance, node density in an area, etc., can differ between network partitions. Furthermore, the network infrastructure evolves over time as a result of node/network failure, battery depletion of nodes, updates, the deployment of new applications and/or changing environmental conditions. For example, many large-scale WSNs are deployed in the open air and thus are subject to node failure caused by harsh environmental conditions [1]–[3].

We propose a framework based on modelling large-scale network infrastructures and expressing application requirements as logical properties, and then reasoning about those properties using automated reasoning techniques. The novelty of our work is our models capture not only the spatial (i.e. physical hierarchy), operational (i.e. hardware, network, application configuration) and application-specific aspects of these infrastructures, but also their temporal evolution, i.e. *dynamic* behaviour. System reconfiguration, i.e. dynamic behaviour, is event-driven; typical infrastructure events include sensor measurement, application deployment, node failure. Also, we consider the different stages of the application life-cycle in the form of offline (design-time and deployment), online (monitoring) and predictive analysis. By large-scale, we mean networks with a significant number of nodes, e.g. over 100; and while we refer mainly to WSNs this work applies to sensor networks in general.

The modelling formalism is *Bigraphical Reactive Systems (BRS)*, a universal computational model defined by Milner [4] for modelling interacting systems that evolve in time and space, originally introduced in the field of Cyber-Physical Systems. This is an ideal formalism for this domain because it allows us to express how the spatial arrangement of nodes may drive computational effects within the system. Dynamic behaviour is defined by rules for rewriting bigraph models, referred to as *reaction rules*. Reasoning is based on bigraph patterns, which were introduced in [5]. BRS have been applied in various domains, e.g. modelling and verification of Mixed-Reality systems [5], networking [6], [7], and Cyber-Physical systems [8].

The contributions of this paper are summarised as follows:

- models of WSN infrastructure and their dynamic behaviour using BRS and reaction rules,
- application properties expressed as bigraph patterns for state predicates and temporal logic properties over bi-

graph patterns for computation paths and transition systems,
- integration with BigraphER [9] for automated model evolution and reasoning about bigraph patterns,
- use of Cooja simulator [10] for simulating network events and generating data streams in a WSN used for experimentation,
- demonstration of the framework through a case-study based on data streams collected from a real-life deployment of urban environmental monitoring, involving two applications running on a WSN of up to 200 nodes and data collected from the deployment over a period of two months.

The paper is organised as follows. Section II presents the case-study that is used for illustration throughout the paper. Section III presents the basic model for large-scale sensor network infrastructures using BRS and in Section IV we extend this to include dynamic behaviour. Section V is dedicated to reasoning about application requirements (properties). In Section VI we demonstrate the framework through experimentation with a model of the case-study and an evaluation of the performance of automated reasoning, using BigraphER, actual data streams, and Cooja for generating events and data streams that replay actual network events and data sensing in a WSN.

We consider two (state) properties that can be verified at run-time and give results for reasoning over WSNs with up to 200 nodes. Related work is presented in Section VII and we conclude in Section VIII.

II. A Large-Scale Sensor Network Infrastructure with Applications

This section presents the case-study used throughout the paper to illustrate our approach. It is based on our experience in developing a number of cross-city sensing applications and deploying sensor network infrastructures in urban areas such as the microclimate WSN that was deployed in 2017 in the Queen Elizabeth Olympic Park (London, UK) [3].

Assume a large-scale wireless sensor network infrastructure distributed over the city, composed of nodes offering processing capabilities and instrumented with heterogeneous sensors. This infrastructure is established by the city council for the purpose of monitoring various common environmental conditions including pollution, light, vibration, temperature, wind speed, etc.

Applications. Assume two applications, one is an event-driven service, the other based on periodic reporting.

Environmental monitoring application (Srv_1). To determine the environmental conditions in frequently visited urban areas and parks, this application requires data to be collected from pollution, temperature, humidity and light sensors connected to microcontroller class devices (nodes). The collected sensor data are used to compute the environmental status of urban areas and inform users.

Structural health application (Srv_2). This application monitors the material conditions of buildings and bridges in the city. Vibration sensors and wind speed sensors are needed to accurately monitor the structural health conditions of these objects.

Application Requirements. Application requirements can be defined at the level of individual network partitions or the entire sensor network infrastructure.

Data gathering (Req_1). Each application requires a minimal number of sensor nodes in the network to be available per application-specific partition (e.g. bridge, building, park) to ensure sufficient data is collected.

Data delivery (Req_2). Srv_1 is an event-driven service that requires pollution, temperature and humidity data to be delivered to the application when the measured pollution level exceeds a safety threshold. Srv_2 is a periodic-based reporting service, requiring vibration and wind measurements to be delivered to the application at least once a minute.

Data processing (Req_3). Apart from collecting sensor measurements, Srv_2 requires application-specific computations to be carried out on the nodes for applying data compression and filtering. This requirement entails that a node has to have enough processing power to carry out all computations within a given time t.

Node resources (Req_4). To ensure the correct operation of the sensor network infrastructure, it is crucial to ensure that the energy levels of battery-powered nodes do not fall below a minimum threshold, as sensor devices are known to misread as the battery depletes.

Node failure (Req_5). Node failure may routinely occur in the infrastructure, it is crucial to ensure that the number of failed nodes does not exceed a level tolerated by Srv_1 and Srv_2.

Finally, we assume that the system is safe, secure and maintained over its lifetime, accounting for failure, exactly like data centre management but using highly limited computational resources and with the added complexity of cyber-physical interactions, i.e. where the positioning and environment of the devices are heavily impacted by the physics of their environments.

III. Modelling Sensor Network Infrastructures

Modelling large-scale sensor network infrastructures requires formalisms capable of expressing complex hierarchies of physical locations as well as connectivity, i.e. the links among the entities of the system. Moreover, they should offer support for the temporal evolution of a system and automated formal reasoning. Finally, *compositionality* is an essential feature for enabling modelling at scale. While various formalisms might fit these criteria to a greater or lesser extent, we propose that *bigraphs*, a universal process algebra that encapsulates both dynamic and spatial behaviour, fit all these criteria particularly well. Another distinctive feature of bigraphs is that they bridge formal mathematical modelling and systems design by supporting equivalent diagrammatic and algebraic representations. This allows systems designers to express graphically the spatial arrangement of the systems under consideration and to use these graphical forms as the principal modelling representation. In addition, the graphical form provides a convenient way for reporting feedback and analysis results to non-expert users.

We now present details of the bigraph model, we begin with some technical background on bigraphs and then gradually introduce the model through a series of examples, each one capturing a different aspect of the system.

A. Modelling with bigraphs

Bigraphs [4] were introduced by Milner as a universal mathematical model for representing the spatial configuration of physical or virtual objects and their interaction capabilities. A bigraph consists of a pair of relations over the same set of entities: a directed forest, called *place graph*, representing topological space in terms of containment and a hyper-graph, called *link graph*, expressing the interactions and (non-spatial) relationships among entities. Each entity is assigned a *type*[1] which determines its *arity*, i.e. number of links, and whether it is *atomic* i.e. it is a leaf in the place graph. Types can also be *parameterised*. The bigraphs in our model are *abstract*, therefore entities do not have identifiers. We employ *singleton types* [11] (types with exactly one value) to uniquely identify entities. For the purpose of presenting our approach, we provide only an informal overview of bigraphs. A concise semantics can be found elsewhere [4].

$P.Q$	Nesting i.e. P contains Q	(1a)
$P \mid Q$	Merge product	(1b)
$P \parallel Q$	Parallel product	(1c)
id	Identity	(1d)
$\mathsf{K}_{x,y}$	Entity of type K with names x and y	(1e)
$/x\ P$	Closure of name x	(1f)

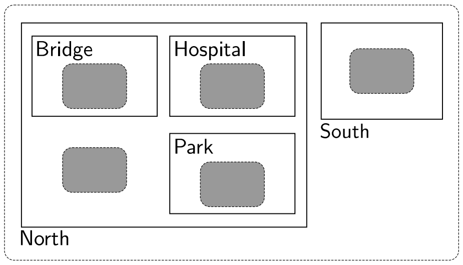
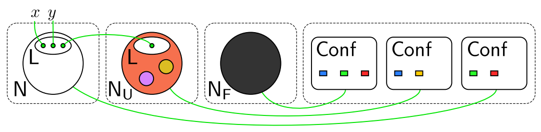


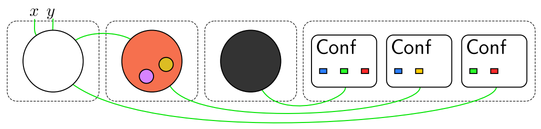
Fig. 1. Bigraph model of an example hierarchy of physical locations.

Bigraphs can be described in algebraic terms (Formulae 1a-1f) or with an equivalent rigorous graphical representation. Fig. 1 shows a bigraph with five entities modelling an example hierarchy of physical locations. Types are indicated here by North, Park, Hospital, etc. In general, bigraphs permit any kind of shape (sometimes coloured) for typed entities. In our model, we use boxes to denote physical locations. Spatial

[1]Types are sometimes called *controls* in bigraphs literature.



(a)



(b)

Fig. 2. Bigraph model of sensor network nodes (a) and corresponding simplified representation (b).

placement of entities is described by Formulae 1a and 1b. *Nesting* is the operation defining the containment relation on entities; *merge product* is the operation allowing to place two entities side-by-side at the same hierarchical level. Hence, in our example, North contains Bridge and North and South are at the same level in the spatial hierarchy. Grey rectangles are called *sites* and are specified by Formula 1d. They indicate parts of the model that have been abstracted away. In other words, an entity containing a site can contain zero or more entities of any kind. For example, Hospital may contain other locations specifying the floors and rooms within the building. Note that entities of atomic types cannot contain sites. Finally, a dashed rectangle denotes a *region* of adjacent parts of the system. Using the algebraic notation, the example bigraph in Fig. 1 can be expressed as follows:

$$\mathsf{North}.(\mathsf{Bridge} \mid \mathsf{Hospital} \mid \mathsf{Park} \mid \mathsf{id}) \mid \mathsf{South}$$

Connectivity between entities is specified by Formulae 1e and 1f, and is represented in the graphical notation by green edges called *links* as shown by the example bigraph in Fig. 2a. Links may be only partially specified, in which case they connect a *name*. Names are links (or potential links) to other bigraphs representing the external environment or context. By convention, names are drawn above the bigraph. In the example, names x and y are used to indicate incoming and outgoing links to remote resources. Sensor network nodes are represented in our model by three kinds of circles: uncoloured, amber, and dark grey for idle nodes (N), nodes in use ($\mathsf{N_U}$), and failed nodes ($\mathsf{N_F}$), respectively. Note the three nodes in the example may be in different physical locations as they are contained by three distinct regions. This is specified in the algebraic notation by Formula 1c. Network connections between nodes are modelled by binary links between *link-ends*. These are entities of type E represented in Fig. 2a as small green circles. All link-ends of a node are grouped within an entity of type L. This modelling strategy allows nodes to be connected to an arbitrary number of links while keeping the same type, thus the same arity. For purposes of aesthetic clarity, in the rest of this paper we adopt a simplified graphical notation in which E and L entities are omitted, as shown in Fig. 2b. Each node is also linked to its *configuration*, modelled

Description	Type(s)	Arity	Atomic	Notation
Node idle	N	1		circle
Node in use	$\mathsf{N_U}$	1		amber circle
Node failed	$\mathsf{N_F}$	1	✓	dark grey circle
Links	L	0		—
Link ends	E	1	✓	—
Configuration	Conf	1		rounded box
Data/Setting	P(x), W(x), ...	0	✓	small coloured box
App	App(x)	0		oval
App token	A(x)	0	✓	small coloured circle

TABLE I
ENTITY TYPES USED IN BIGRAPH MODEL.

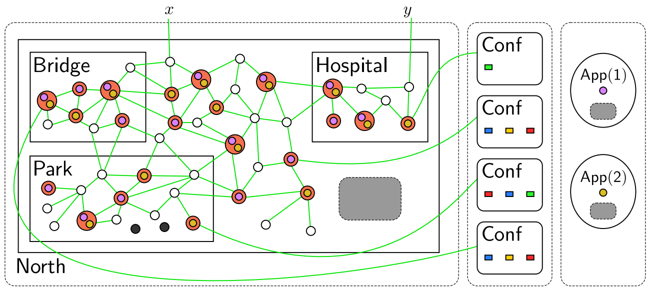


Fig. 4. Bigraph model of example city-wide sensor network infrastructure. The three regions in the bigraph correspond, from left to right, to the Physical, Data, and Service perspectives, respectively.

by the Conf rounded boxes in the region on the right. We adopt a code of small coloured boxes within Conf to represent the sensors installed on each node: red for atmospheric pressure (P(x)), green for temperature (T(x)), blue for wind speed (W(x)), and yellow for vibration (V(x)), etc. Typically, the types of these entities are parameterised and are used to carry data. For example, a node configuration may contain an entity of type P(987.54) to indicate a specific value of sensed atmospheric pressure. Other types of entity we employ in our model include MAC(x) and IP(x) to represent MAC and IP addresses, respectively. Each type in the form MAC(x) is a singleton type and is used to uniquely identify a node. In algebraic terms, the bigraph in Fig. 2a can be expressed as follows:

$$/l\ /a\ /b\ /c\ (\mathsf{N}_a.\mathsf{L}.(\mathsf{E}_x \,|\, \mathsf{E}_y \,|\, \mathsf{E}_l) \parallel \mathsf{N_U}_b.(\mathsf{L}.\mathsf{E}_l \,|\, \mathsf{A}(1) \,|\, \mathsf{A}(2)) \parallel \mathsf{N_F}_c \\ \parallel (\mathsf{Conf}_c.(\mathsf{W} \,|\, \mathsf{T} \,|\, \mathsf{P}) \,|\, \mathsf{Conf}_b.(\mathsf{W} \,|\, \mathsf{V}) \,|\, \mathsf{Conf}_a.(\mathsf{T} \,|\, \mathsf{P})))$$

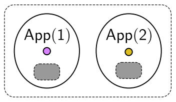


Fig. 3. Bigraph model of WSN applications.

Various applications may use different sets of nodes of the network: this is expressed in the model by placing *app tokens* (A(x)) inside $\mathsf{N_U}$ nodes. They are represented as small coloured circles, where a given colour indicates a specific application. In Fig. 2 for example, applications Srv_1 and Srv_2 (defined in the case-study in Sec. II) are assigned the mauve and yellow colours and are deployed on the amber node. Application-specific properties and settings are indicated as in Fig. 3 by the two ovals of type App(1) and App(2). The mapping between applications and app tokens is defined by placing a token A(x) in App(x). Note that additional settings can be stored within the two ovals as they both contain a site. This example is expressed algebraically as

$$\mathsf{App}(1).(\mathsf{A}(1) \mid \mathsf{id}) \mid \mathsf{App}(2).(\mathsf{A}(2) \mid \mathsf{id})$$

A complete summary of the types defined by our model is reported in Tab. I. At this stage, the full model of an example city-wide sensor network infrastructure can be defined by composing three bigraphs defined as in the three examples described above (Figs 1, 2, and 3). The result is shown in Fig. 4. Following the work of Benford et al. [5], we adopt a modelling approach based on three design perspectives each of which addresses a different facet of the overall system in depth: *Physical perspective* in which we model physical locations, nodes and their connectivity; *Data perspective* in which we model sensed data and node settings; *Service perspective* in which we model key aspects of the application deployed on the network. Each perspective corresponds to a region in the bigraph.

IV. Modelling Evolving Sensor Network Infrastructures

The model introduced in the previous section only supports static configurations: it does not describe the evolution of a system over time. In our application scenario, temporal changes are triggered by the occurrence of *events*, such as nodes failing, new nodes being installed, new applications being deployed, etc. We extend the model to encompass dynamic aspects by representing events arising in WSNs by means of *reaction rules*, a form of rewrite rules for bigraphs. We employ reaction rules in two orthogonal ways: as deterministic sequences of operations to update a bigraph; as components of a Bigraphical Reactive System (BRS) that can iteratively be applied in any order to compute the set of reachable configurations.

We start by providing a formal definition of reaction rules and BRS and then illustrate our model of events through four examples.

A. Events as reaction rules

A *reaction rule* consists of a pair of bigraphs: the left-hand side specifies the portions of a bigraph to be changed, while the right-hand side specifies how those are changed. We use $\longrightarrow$ to indicate the definition of reaction rules. Like in any rule-based system, a reaction rule $R \longrightarrow R'$ is applicable to a bigraph B when R is an occurrence in B (this is also called *bigraph matching*). The result of the application is bigraph B' which is obtained by substituting (in B) an occurrence of R with R'. Such a *reaction* is indicated with $B \rhd B'$. Finally, reaction rules can be *parameterised*.

A *BRS* consists of a set of reaction rules together with an initial bigraph on which the rules operate. Its *transition system* is a (possibly infinite) graph whose vertices are bigraphs representing the reachable states and whose edges represent state transitions, i.e. reactions over bigraphs. Within BRS, each reaction rule can be assigned a stochastic rate to model stochastic events (e.g. node failures). The corresponding transition system can then be treated as a Continuous Time Markov Chain (CTMC) [12]. Rule priorities can be introduced by defining a partial ordering on the reaction rules of a BRS, as implemented in [9].

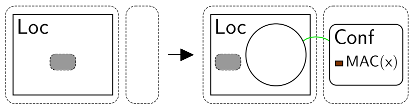


Fig. 5. Reaction rule $\texttt{new}(x)$: install a new node.

A simple example reaction rule is shown in Fig. 5. In our model, it defines the installation of a new idle node at a given physical location. The two regions represent the Physical and Data perspectives. In the right-hand side, an entity of type N (circle) is introduced in Loc and linked to its Conf box. Note the configuration initially only consists of the node's MAC address which is represented by the entity of type $\mathsf{MAC(x)}$. When the rule is instantiated, concrete values for parameter x are used, e.g. $\mathsf{MAC(34{:}36{:}3b{:}6e{:}ce{:}38)}$. Each entity of type Loc could contain other locations or nodes as indicated by the presence of a site. The reaction rule can be defined in algebraic terms:

$$\begin{aligned}\texttt{new}(x) &\overset{\text{def}}{=} \mathsf{Loc} \parallel 1\\ &\longrightarrow /a\,(\mathsf{Loc}.(\mathsf{N}_a.\mathsf{L}.1 \mid \mathsf{id}) \parallel (\mathsf{Conf}_a.\mathsf{MAC(x)}))\end{aligned}$$

Here 1 is used to denote an empty region. Hence, $\mathsf{L}.1$ means the entity does not contain a site.

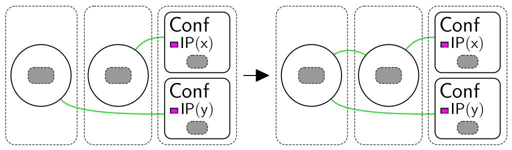


Fig. 6. Reaction rule $\texttt{link}(x, y)$: establish a new communication link between two idle nodes.

Network topology is modelled by binary connections between nodes. A new communication link between idle nodes can be established by applying the reaction rule given in Fig. 6. In the left-hand side, the two idle nodes (circles) are in two distinct regions, meaning they can be in different locations in the physical hierarchy. Each node is linked to an entity of type Conf in the region corresponding to the Data perspective. The only relevant settings in the reaction rule are the two IP addresses represented by the parameterised entities $\mathsf{IP(x)}$ and $\mathsf{IP(y)}$. Note that although ignored here, other settings/data may be present as both Conf boxes contain a site. In the right-hand side, a new link between the two nodes is established. The equivalent algebraic representation is

$$\begin{aligned}D(x,y) &\overset{\text{def}}{=} \mathsf{Conf}_a.(\mathsf{IP(x)} \mid \mathsf{id}) \mid \mathsf{Conf}_b.(\mathsf{IP(y)} \mid \mathsf{id})\\ L &\overset{\text{def}}{=} \mathsf{L}.(\mathsf{E}_l \mid \mathsf{id}) \mid \mathsf{id}\\ \texttt{link}(x,y) &\overset{\text{def}}{=} /a\,/b\,(\mathsf{N}_a.(\mathsf{L} \mid \mathsf{id}) \parallel \mathsf{N}_b.(\mathsf{L} \mid \mathsf{id}) \parallel D(x,y))\\ &\longrightarrow /l\,/a\,/b\,(\mathsf{N}_a.L \parallel \mathsf{N}_b.L \parallel D(x,y))\end{aligned}$$

This reaction rule is very general and can be used to represent (through iterative applications) any network topologies (e.g. star, mesh, etc.). Similar reaction rules can be defined to handle N_U nodes.

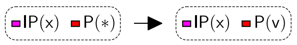


Fig. 7. Reaction rule $\texttt{read}(x, v)$: read sensor data from a node.

Sensing events are naturally modelled within the Data perspective by reaction rules in the form of the one described in Fig. 7. In this example, a sensor with IP address x is queried to get an updated value of atmospheric pressure, indicated by parameter v. In the left-hand side, type $\mathsf{P}(*)$ is used to match any value previously recorded by the sensor[2]. The rule is specified algebraically as follows

$$\texttt{read}(x,v) \overset{\text{def}}{=} \mathsf{IP(x)} \mid \mathsf{P}(*) \longrightarrow \mathsf{IP(x)} \mid \mathsf{P(v)}$$

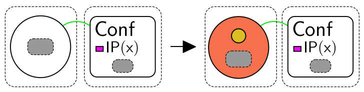


Fig. 8. Reaction rule $\texttt{sub}(x, y)$: subscribe an app to a node.

Finally, the last example shows the encoding of app subscription events. This is represented by the reaction rule in Fig. 8. The left-hand side is defined in a similar fashion to the one of reaction rule $\texttt{link}(x, y)$, but with only one idle node. In the right-hand side, the node is in use, thus its type is changed to N_U (amber circle). Moreover, an app token of type $\mathsf{A(y)}$ (yellow circle) is placed within it. The equivalent definition using algebraic notation is

$$\begin{aligned}\texttt{sub}(x,y) &\overset{\text{def}}{=} /a\,(\mathsf{N}_a.(\mathsf{L} \mid \mathsf{id}) \parallel \mathsf{Conf}_a.(\mathsf{IP(x)} \mid \mathsf{id}))\\ &\longrightarrow /a\,(\mathsf{N_U}_a.(\mathsf{L} \mid \mathsf{A(y)} \mid \mathsf{id}) \parallel \mathsf{Conf}_a.(\mathsf{IP(x)} \mid \mathsf{id}))\end{aligned}$$

The complete model includes more events such as: drop a communication link, move a node to a different location, join a network, battery depletion, deploy a new application on the system, etc. These can easily be expressed following the templates of the four reaction rules presented in this section.

[2]In our model, this is achieved by defining reaction rules equipped with *instantiation maps*. See source code at `https://github.com/mkabac/iceccs-2018/blob/master/model/model.big` for further details.

V. Verification of Sensor Network Infrastructures

We consider three different paradigms for automated property verification:

- checking a (static) property at deployment,
- monitoring properties as the system evolves, either iteratively checking (static) properties after every event, or checking (dynamic) properties in a linear temporal logic on computation paths,
- checking (dynamic) properties in a branching time logic on the transition system generated from all possible future application of reaction rules (i.e. all possible future events).

Static properties are specified by bigraph patterns, dynamic properties are specified by (possibly stochastic) temporal logics with bigraph patterns as atoms. The application requirements (i.e. $\mathsf{Req}_1, \ldots, \mathsf{Req}_5$) of the case-study presented in Sec. II serve as examples. The modelling and verification framework is depicted in Fig. 9; details are below.

A. Static properties with bigraph patterns

Bigraph patterns were introduced in [5] to express static properties over bigraphs in terms of bigraphs matching: a pattern φ is satisfied by a given bigraph B, if an instance of φ occurs in B. Patterns can also be combined with standard Boolean operators to form logical formulae; we write $\top$ to denote true, $\neg$ for negation and $\wedge$ to indicate conjunction. In our approach, bigraph patterns can be used both at deployment time and at runtime for monitoring. In the former, they act as predicates expressing requirements to be checked against a bigraph representation of the WSN; in the latter, they represent invariants that are checked at each rewrite step triggered by the occurrence of an event in the WSN.

An example bigraph pattern is given in Fig. 10. Informally, $\varphi_1(\mathsf{Loc})$ is satisfied by any bigraphs in which location Loc contains at least three idle nodes. In algebraic terms, the pattern is given by

$$\varphi_1(\mathsf{Loc}) \overset{\text{def}}{=} \mathsf{Loc}.(\mathsf{N}_a \mid \mathsf{N}_b \mid \mathsf{N}_c \mid \mathsf{id})$$

Similar patterns can be defined by specifying different locations or numbers of nodes. These patterns can be combined to define properties formalising Req_1: there are sufficient nodes available in every app-specific network partition. An example is given by the following formula

$$\varphi_1(\mathsf{Bridge}) \wedge \varphi_1(\mathsf{Park}) \wedge \varphi_1(\mathsf{Hospital})$$

We formalise Req_4 with predicates that are true if and only if a given resource is above threshold t in every node. For instance, when considering battery level (represented by node $\mathsf{B(x)}$), this is defined by

$$\varphi_2(x,t) \overset{\text{def}}{=} \neg\mathsf{B(x)} \qquad \text{with } x \leq t$$

The next following three patterns will be used in conjunction with temporal and stochastic logics.

A bigraph pattern to check whether there exists a node with IP address x and timestamp $\mathsf{UTC(t)}$ indicating the last time t a read event was recorded by the node is given by

$$\varphi_3(x,t) \overset{\text{def}}{=} \mathsf{IP(x)} \mid \mathsf{UTC(t)}$$

Finally, the two predicates introduced in Fig. 11 are satisfied when a node with IP address x is serving app y ($\varphi_4(x,y)$) and when it has failed ($\psi_4(x)$). The corresponding algebraic formulation is

$$\varphi_4(x,y) \overset{\text{def}}{=} /a\,(\mathsf{N}_{\mathsf{U}a}.(\mathsf{A(y)} \mid \mathsf{id}) \parallel \mathsf{Conf}_a.(\mathsf{IP(x)} \mid \mathsf{id}))$$
$$\psi_4(x) \overset{\text{def}}{=} /a\,(\mathsf{N}_{\mathsf{F}a} \parallel \mathsf{Conf}_a.(\mathsf{IP(x)} \mid \mathsf{id}))$$

B. Monitoring properties in evolving configurations

We define a computation *path* as a sequence of bigraphs $S_0\, S_1 \ldots S_t$ in which $S_{i-1} \longrightarrow\!\!\!\!\!\triangleright\, S_i$, with $i \leq t$. We indicate S_t as the current configuration (state) of the WSN and label each S_i with the propositions (i.e. bigraph patterns) that hold locally. Such sequences emerge at runtime when the current state is updated iteratively by reaction rules encoding events and then checked against (static) properties.

The specification of more complex and temporal properties for monitoring is possible, for instance, by reasoning over paths in Past-Time Linear-time Temporal Logic (**ptLTL**) [13]. In this logic the $\mathbf{Y}$ (for "Yesterday") operator, which is the temporal dual of $\mathbf{X}$ (for "Next") in **LTL**, refers to the previous time instant, i.e. $S_t \models \mathbf{Y}\phi$ iff $S_{t-1} \models \phi$. Requirements about the the freshness of data as required in Req_2 can easily be expressed by combining this operator with the bigraph pattern defined above:

$$\varphi_3(x,t) \wedge \mathbf{Y}\varphi_3(x,t') \qquad \text{with } t - t' \leq 60\text{s}$$

where x indicates the node's IP address.

C. Transition systems

For the purposes of illustration, assume we wish to focus on quantitative aspects of WSNs; for example, rules may encode the stochastic rates of events such as hardware failure, battery depletion, and node repair. The generated transition systems are CTMCs. When defining reaction rules, same care needs to be taken to generate a finite state space. For instance, rule priorities may be necessary to avoid adding duplicate communication links through repeated applications of reaction rule $\mathtt{link}(x,y)$. Similar to labels in paths, the CTMCs are augmented with labels indicating which bigraph patterns hold in each state. Quantitative dynamic properties are expressed by combining bigraph patterns with Continuous Stochastic Logic (**CSL**) [14]. The process of generating a labelled CTMC and verifying dynamic properties against it can be computationally expensive, so this can be carried out separately from the monitoring process. Also, when the state space is too large, it may be necessary to bound the number of transitions within a given path.

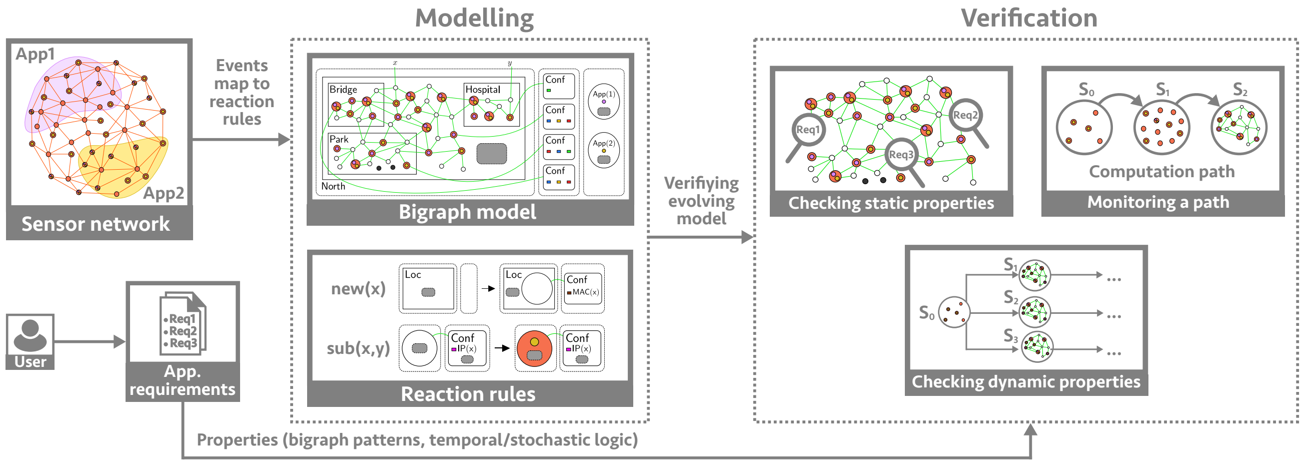


Fig. 9. Modelling and verification framework for large-scale sensor network infrastructures.

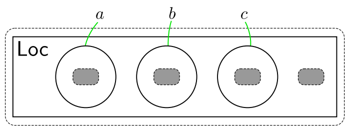


Fig. 10. Bigraph pattern $\varphi_1(\mathsf{Loc})$: a location contains at least three idle nodes.

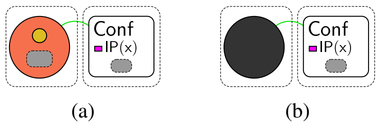


Fig. 11. Bigraph patterns $\varphi_4(x, y)$ and $\psi_4(x)$: a node is serving an app (a); a node has failed (b).

Two simple example properties are:

$$\mathbf{S}_{<0.1}\,[\,\neg\varphi_2(x,t)\,] \tag{2}$$

$$\mathbf{P}_{<0.1}\,[\,\varphi_4(x,y)\,\mathbf{U}_{\leq t}\,\psi_4(x)\,] \tag{3}$$

Formula 2 is another formalisation of Req_4: it specifies that the long-run probability of any node to have battery level x below threshold t is less than 0.1. **S** is the **CSL** steady state operator. Formula 3 formalises Req_5, which holds if the probability within t time units of failure of node with IP address x while serving app y is less than 0.1. This is a transient property with "Until" operator **U**. Propositions $\varphi_4(x,y)$ and $\psi_4(x)$ were introduced in Fig. 11.

Labelled CTMCs can be further augmented with *costs (rewards)* [15] that are associated with states or reactions. In our approach, this extension can be defined by associating numerical values to (a subset) of the reaction rules of a BRS.

The two example properties below are defined by associating costs to the reaction rules modelling the installation of new nodes in the WSN (see $\mathtt{new}(x)$ in Fig. 5). Operator **R** works in a similar fashion to the **P** and **S** operators.

$$\mathbf{R}\{\mathtt{new}\}_{=?}\,[\,\mathbf{C} \leq t\,] \tag{4}$$

$$\mathbf{R}\{\mathtt{new}\}_{=?}\,[\,\mathbf{S}\,] \tag{5}$$

Formula 4 (cumulative) computes the expected operational cost of installing new nodes within t time units, while Formula 5 (steady-state) computes the expected long-run cost rate per unit of time.

VI. Experimentation and Evaluation

In this section we give some results concerning our case-study and evaluate the performance of automated reasoning for monitoring two example properties. Our aim in this work is threefold.

First, we demonstrate *applicability* of our approach by modelling an example sensor network infrastructure and expressing application requirements as properties.

Second, we have developed an experimental set up to reason about actual or simulated network evolutions. This requires, in addition to the bigraph model, reaction rules, and application properties to be proved (or disproved), both a stream of sensed values (data) and a stream of network events. The data and network events correspond to reaction rules in the model.

Third, we evaluate automated reasoning *performance* by measuring the time taken to update configurations of the infrastructure model and verify static properties. We focus on updates that correspond directly to the different events of interest in the infrastructure: each update involves many rewrite steps as well as the verification of predicates presented in Section IV.

A. Modelling a city-wide environmental monitoring infrastructure

Full details of the bigraph model can be found on a public repository hosted on GitHub[3]. The entities are defined using the types presented in Tab. I. The configuration Conf of

[3]https://github.com/mkabac/iceccs-2018

sensor nodes defines the MAC address, IP address, a range of integrated sensors (temperature, humidity, light, etc.), the current battery state and the timestamp of the latest sensor reading. The WSN infrastructure is partitioned into physical locations presented in Fig. 1, Sec. III-A. The initial state of the infrastructure is a bigraph that consists of three perspectives: Physical, Data and Service as shown below.

$$S_0 \stackrel{\text{def}}{=} (\mathsf{North}.(\mathsf{Hospital} \mid \mathsf{Bridge} \mid \mathsf{Park}) \mid \mathsf{South} \mid \mathsf{c}) \parallel 1 \parallel 1$$

We denote a link to external resources with name c. Events are defined as rules, namely, `deploy`, `new`, `join`, `link`, `read` and `sub` as presented in Sec. IV-A. Application requirements (properties) Req_1 and Req_4 are expressed by bigraph patterns as predicates $\varphi_1(\mathsf{Loc})$ and $\varphi_2(x, t)$, as defined in Sec. V-A. Both requirements can be verified on states at runtime.

B. Experimental setup

Fig. 12 depicts the experimental set up, which we describe in more detail as follows.

BigraphER. For bigraph reasoning, we used a suite of open-source tools that provide support for rewriting and visualisation of bigraphs and reaction rules called BigraphER [9]. We encode both the model and the bigraph patterns in BigraphER.

Event generation. Events can be real-life, or simulated. In the case of simulation, the stream can be a replay of real (historical) events, which is the approach we take here.

To generate the simulated events we integrated BigraphER with Cooja [10], a WSN simulator for the Contiki operating system [16], which combines low-level emulation of sensor node hardware with firmware from different vendors and simulates network and radio model behaviors. The integration with Cooja allows us to evaluate the performance of our approach against simulated WSNs at different scales. Here, we assume a star topology and implement the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), the closest to a reference standard provided by this community. The sensor nodes are emulated using firmware for the Tmote Sky platform. The size of the simulated WSNs range from from 10 to 200 nodes.

Data values. We use data drawn from a sensor dataset[4] collected from the microclimate sensing system deployed in the Queen Elizabeth Olympic Park in London (see Section II) over a two-month period in 2017. Observe that this dataset was collected aperiodially, because the nodes were energy harvesting, operating in an energy neutral regime. In the experimental setup, the timestamps are not considered. Instead, data is delivered periodically, at least once every 20 seconds, and associated with reaction rule $\texttt{read}(x, v)$ (value v at node x).

Workflow. At all times, the bigraph model is a model of the current (simulated) WSN, so the model evolves, i.e. is updated by the (reaction rules that correspond to) generated events. The

[4]`https://github.com/mkabac/iceccs-2018/blob/master/data/input_dataset/ENO_Box_3B06.csv`

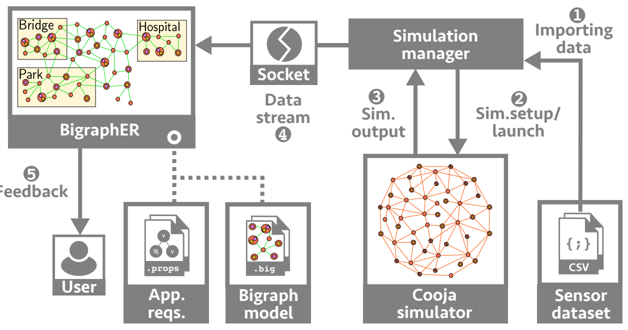


Fig. 12. The experimental setup for evaluating the framework with actual data and network events replayed by Cooja. Cooja can be replaced by an actual network.

predicates are repeatedly checked on the evolving model. In the following, numbers refer to arrows in Fig. 12.

We used OCaml for interfacing BigraphER with Cooja as well as for implementing the pre-processing of data streams. The data stream for the experiment is generated from a dataset imported from a CSV file using the simulation manager (1). The WSN is configured and launched in a script, which is provided to a simulation manager component in Cooja (2). The output of the simulator is collected and mapped to events (3); these include update events such as a new sensor node joining the network, a new link between nodes, sensor measurement, etc. The simulator can be replaced easily by an actual network, no modifications are required. The interaction between the (simulated) WSNs and BigraphER is ensured via TCP sockets (4). The evolving configurations of bigraph models are presented to users through diagrams and datasets in the JSON format (5).

Hardware. The evaluation has been carried out on a computer equipped with a 3.40GHz Intel Core i7-4770 processor and 16GB of RAM. Both Cooja and the BigraphER implementation ran simultaneously.

C. Experimental results

The results of the performance evaluation are presented in Tab. II. Each row in the table provides, respectively, (1) the size of the simulated WSN in terms of sensors, (2) the size of the sensor network model in terms of bigraph entities, (3) the number of rewriting steps needed for graph reconfiguration over the duration of the experiment and (4) - (9) the average time for completing reaction rules encoding WSN events into the model. These performance measurements include the time needed for completion of reaction rules to encode events from WSNs into the bigraph model as well as the time needed for the execution of predicates for the verification of application requirements Req_1 and Req_4 as defined in Sec. II. The reaction rules `new`, `join`, `sub`, `link` and `read`, have been triggered once per sensor node in each experiment. The `deploy` reaction rule has been triggered only twice to encode the deployment of services Srv_1 and Srv_2. We considered WSNs up to size 200 nodes. Cooja was the limiting factor: we were unable

WSN size (sensors)	Final model size (entities)	Rewriting steps	deploy (ms)	new (ms)	join (ms)	sub (ms)	link (ms)	read (ms)
10	223	108	0.352	1.337	1.633	2.152	9.713	10.260
50	987	440	0.366	5.936	8.141	11.120	54.063	102.050
100	2023	905	0.429	16.020	21.452	30.420	162.793	422.689
150	2969	1337	0.298	29.620	38.588	55.508	298.086	828.165
200	3810	1624	0.298	47.488	60.174	87.592	451.759	1158.441

TABLE II
PERFORMANCE MEASUREMENTS FOR THE MODELLING AND VERIFICATION OF SIMULATED WSNS AT DIFFERENT SCALES.

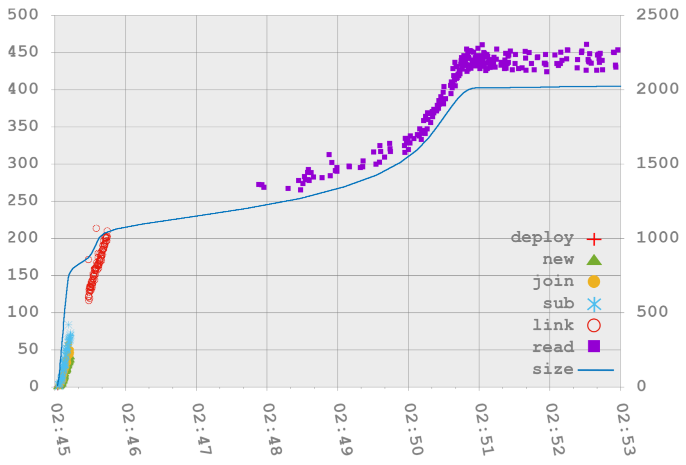


Fig. 13. Performance evaluation for WSN with 100 nodes. [x] execution time (hh:mm), [left y] event processing time (ms), [right y] model size (bigraph entities).

to simulate larger WSNs with Cooja as the simulation speed decreased to 1% of its normal operation.

The events of interest were generated in the same order as they are defined in Tab. II. Generally, we observe that the time it takes to complete reaction rules increases over the course of the experiment for each simulated WSN. This is because the execution of reaction rules is done over a model that is gradually increasing in size, causing later reaction rules (e.g. `link`, `read`) to require more time for bigraph matching compared to reaction rules applied to a smaller model at an early stage (e.g. `deploy`, `new`). Furthermore, scaling up the WSN naturally provides larger bigraph models and gradually increases execution time for the majority of reaction rules. The relation between the size of the bigraph model and the execution time of reaction rules is shown in Fig. 13 for the experiment carried out over a WSN with 100 nodes.

These preliminary results indicate that the prototype reasoning implementation can support modelling and online verification of sensor network infrastructures comprising of up to 200 nodes. We note that our case-study applications (environmental monitoring, structural health) are delay-tolerant applications without hard real-time requirements.

VII. Related Work

Modelling and verification in the context of sensor networks and cyber-physical systems has been addressed in various works, at different levels of abstraction. We divide the related work into approaches targeting the modelling of spaces, network-level and node-level aspects of these environments.

Spaces. Mottola et al. [17] propose logical neighbourhoods, a programming abstraction that defines the notion of proximity according to functionality related characteristics of sensor nodes, including both static and dynamic properties. This programming abstraction is supported by a routing protocol and a language allowing developers to define neighbourhoods declaratively. This work focuses on providing an abstraction for mapping nodes to neighbourhoods and capturing the set of nodes with functionally related characteristics. The abstraction is used to ensure efficient routing strategies. In contrast, the modelling in our approach allows us to express not only the spatial properties of the infrastructure but also the dynamic behaviour of sensor network infrastructures.

The works by Tsigkanos et al. [8] and Calder et al. [6] are the most similar to ours. In the work by Tsigkanos et al. [8], the authors presented a methodology and a BRS framework for the modelling of evolving Cyber-Physical Spaces (CPS) (e.g. offices, buildings, cities) and reasoning about spatio-temporal properties. Calder et al. [6] presented a BRS approach for modelling home wireless network infrastructures, focussing on modelling and reasoning about spatial and temporal behaviour of network interactions and network policies. Compared with these approaches, our work provides models that are tailored to large-scale sensor network infrastructures and define complex relationships between sensor nodes, their configurations, hierarchies of physical locations and deployed application services. These models are augmented with domain-specific requirements that can be expressed with a combination of spatial, temporal and stochastic logics.

Network. Modelling and verification at the network level is frequently used to analyse network protocols and algorithms. We mention two approaches dedicated to the modelling and verification of network-level supports in the context of WSN. The work by Olveczky et al. [18] presented an approach for the modelling, simulation, and model checking of advanced WSN algorithms. The authors presented a language and tool for the formal specification and analysis of real-time systems called Real-Time Maude, which uses a formalism based on an extension of rewriting logic that can be used for specifying distributed real-time systems in an object-oriented style.

Fehnker et al. [19] reported on the modelling and verification of a medium access control protocol for WSNs called LMAC. The authors analysed the ability of the LMAC protocol to detect and resolve collision using a timed automaton model and the UPPAAL model checker.

Sensors. At the level of sensors, modelling and verification is needed to make guarantees on the code running on sensors. For example, in the work by Bucur et al. [20], authors presented the first tool dedicated to the pre-deployment verification of multithreaded, adaptive applications written in nesC for the TinyOS platform. To do so, the approach performs static verification of a TinyOS application running on a sensor node against a context-aware safety specification requiring the program to be in a safe state with respect to actuation and memory configuration. Kleenet [21] is an alternative tool for the pre-deployment testing of sensor network applications based on the KLEE symbolic virtual machine. Kleenet is designed to detect bugs resulting from complex interactions of multiple nodes, non-deterministic events in the network, and unpredictable data inputs.

The approaches listed at the level of the network and sensors are complementary to our work. Our contribution pertains to different levels of abstraction and defines models of infrastructure for verifying application requirements at run-time. Currently, our approach verifies application requirements pertaining to all three levels of abstraction, such as node density per location (i.e. space), frequency for delivering sensor data (i.e. network) and current battery state on nodes (i.e. sensors).

VIII. Conclusion

We have presented a modelling and verification framework for wireless sensor network infrastructures at a large scale to support reasoning about their spatial, operational and behavioural aspects. We have established a general approach based on BRS for modelling and formal verification that includes bigraph patterns for expressing and monitoring spatial properties on bigraphs, past time linear temporal logic for reasoning over computation paths, and stochastic logic for reasoning about future configurations.

The bigraph model encapsulates three modelling perspectives: the Physical perspective in which we model physical locations, nodes and their connectivity; the Data perspective in which we model sensed data and node settings; and the Service perspective in which we model key aspects of the applications. Each perspective corresponds to a region in the bigraph, which makes the diagrammatic form especially intuitive.

Throughout, we have used an environmental monitoring cases-study for illustration and we provided both algebraic and diagrammatic representations of the bigraphs.

For experimentation and evaluation of our approach for on-line verification, we implemented a prototype automated reasoning set-up based on the open-source BigraphER tool and the Cooja network simulator for generation of events. The events were a replay of real-life data. The results indicate our approach is capable of handling online verification of large-scale infrastructures comprising of up to 200 nodes.

Future work includes evaluation with events streamed from a live sensor network infrastructure such as the FIT IoT-Lab testbed [22]. Finally, we are working on a domain-specific language to improve the usability of our approach by enabling formal algebraic representations to be generated from domain specific, high-level specifications.

Acknowledgment

This work is supported by the Engineering and Physical Sciences Research Council, under grant EP/N007565/1 S4: Science of Sensor Systems Software, and by the Intel Collaborative Research Institute in Urban IoT. We also thank Greg Jackson for providing the dataset collected from the microclimate WSN deployment in the Queen Elizabeth Olympic Park.

References

[1] R. Hartung, U. Kulau, B. Gernert, S. Rottmann, and L. Wolf, "On the experiences with testbeds and applications in precision farming," in *International Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems (FAILSAFE'17)*. ACM, 2017.

[2] X. Fang and I. Bate, "Issues of using wireless sensor network to monitor urban air quality," in *International Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems (FAILSAFE'17)*. ACM, 2017.

[3] G. Jackson, S. Gallacher, D. Wilson, and J. A. McCann, "Tales from the wild: Lessons learned from creating a living lab," in *International Workshop on the Engineering of Reliable, Robust, and Secure Embedded Wireless Sensing Systems (FAILSAFE'17)*. ACM, 2017.

[4] R. Milner, *The space and motion of communicating agents*. Cambridge University Press, 2009.

[5] S. Benford, M. Calder, T. Rodden, and M. Sevegnani, "On lions, impala, and bigraphs: Modelling interactions in physical/virtual spaces," *ACM Trans. Comput.-Hum. Interact.*, vol. 23, no. 2, 2016.

[6] M. Calder, A. Koliousis, M. Sevegnani, and J. Sventek, "Real-time verification of wireless home networks using bigraphs with sharing," *Science of Computer Programming*, vol. 80, Part B, no. 0, 2014.

[7] M. Calder and M. Sevegnani, "Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing," *Formal Aspects of Computing*, vol. 26, no. 3, 2014.

[8] C. Tsigkanos, T. Kehrer, and C. Ghezzi, "Modeling and verification of evolving cyber-physical spaces," in *Foundations of Software Engineering*, ser. ESEC/FSE 2017, 2017.

[9] M. Sevegnani and M. Calder, "BigraphER: Rewriting and analysis engine for bigraphs," in *International Conference on Computer Aided Verification (CAV'16)*. Springer, 2016.

[10] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with Cooja," in *IEEE Conference on Local Computer Networks*, 2006.

[11] S. Hayashi, "Singleton, union and intersection types for program extraction," in *Theoretical Aspects of Computer Software*. Springer Berlin Heidelberg, 1991.

[12] J. Krivine, R. Milner, and A. Troina, "Stochastic bigraphs," *Electronic Notes in Theoretical Computer Science*, vol. 218, pp. 73–96, 2008.

[13] O. Lichtenstein, A. Pnueli, and L. Zuck, "The glory of the past," in *Logics of Programs*. Springer Berlin Heidelberg, 1985.

[14] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-Checking Algorithms for Continuous-Time Markov Chains," *IEEE Trans. Software Eng.*, vol. 29, no. 6, 2003.

[15] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *International Conference on Computer Aided Verification (CAV'11)*. Springer, 2011.

[16] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *IEEE International Conference on Local Computer Networks*, 2004.

[17] L. Mottola and G. P. Picco, "Logical neighborhoods: a programming abstraction for wireless sensor networks," in *IEEE international conference on Distributed Computing in Sensor Systems (DCOSS)*, 2006.

[18] P. C. Olveczky and S. Thorvaldsen, "Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in real-time maude," *Theoretical Computer Science*, vol. 410, no. 2, 2009.

[19] A. Fehnker, L. van Hoesel, and A. Mader, "Modelling and verification of the lmac protocol for wireless sensor networks," in *Integrated Formal Methods*. Springer Berlin Heidelberg, 2007.

[20] D. Bucur and M. Kwiatkowska, "Bug-free sensors: The automatic verification of context-aware tinyos applications," in *Ambient Intelligence*. Springer Berlin Heidelberg, 2009.

[21] R. Sasnauskas, O. Landsiedel, M. H. Alizai, C. Weise, S. Kowalewski, and K. Wehrle, "Kleenet: Discovering insidious interaction bugs in wireless sensor networks before deployment," in *ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN '10, 2010.

[22] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele *et al.*, "FIT IoT-Lab: A large scale open experimental IoT testbed," in *World Forum on Internet of Things (WF-IoT)*. IEEE, 2015.