Modeling and Verification of Evolving Cyber-Physical Spaces

Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy

ABSTRACT

We increasingly live in cyber-physical spaces – spaces that are both physical and digital, and where the two aspects are intertwined. Such spaces are highly dynamic and typically undergo continuous change. Software engineering can have a profound impact in this domain, by defining suitable modeling and specification notations as well as supporting design-time formal verification. In this paper, we present a methodology and a technical framework which support modeling of evolving cyber-physical spaces and reasoning about their spatio-temporal properties. We utilize a discrete, graph-based formalism for modeling cyber-physical spaces as well as primitives of change, giving rise to a reactive system consisting of rewriting rules with both local and global application conditions. Formal reasoning facilities are implemented adopting logic-based specification of properties and according model checking procedures, in both spatial and temporal fragments. We evaluate our approach using a case study of a disaster scenario in a smart city.

CCS CONCEPTS

• **Software and its engineering** → *Software system models*; *Model-driven software engineering*; *Abstraction, modeling and modularity*; *Formal methods*; *Model checking*; *Requirements analysis*;

KEYWORDS

Cyber-Physical Spaces; Dependable Systems; Safety and Reliability; Modelling and Specification; Formal Verification

ACM Reference Format:
Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. 2017. Modeling and Verification of Evolving Cyber-Physical Spaces. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 11 pages.
https://doi.org/10.1145/3106237.3106299

1 INTRODUCTION

Computing and communication capabilities are increasingly embedded into physical spaces thus blurring the boundary between computational and physical worlds; typically, this is the case in modern cyber-physical systems, like smart buildings or smart cities, hereafter called space-dependent systems. Conceptually, we consider such a composite environment as a cyber-physical space (CPSp), which consists of interrelated computational and physical entities.

Like any other software-intensive system, a CPSp is not a static construct. Dynamic actions (e.g. performed by agents) generate continuous change, leading to the notion of an *evolving cyber-physical space*. Thus an evolving CPSp must face the manifold challenges of dynamism – change may affect e.g. safety, security, or reliability requirements [50, 51] of the overall space-dependent system.

Formally modeling space and its change as well as reasoning about various properties of evolving space are crucial prerequisites for engineering dependable evolving CPSp. Our approach targets the critical system requirements phase, where a way to obtain formal assurances about the system is highly sought. This phase enables verifying requirements in the early stages of design, before implementing the actual system. Additionally, such high-level reasoning can aid in analysis of system behavior after deployment, or used for bootstrapping adaptation at runtime. Elementary properties of an evolving spatial environment of a software-intensive system can be roughly classified into three kinds:

(spatial; local): Most properties of space are locally bounded, i.e. they refer to direct, elementary relationships between entities or sets of entities forming a pre-defined structural pattern or anti-pattern.

(spatial; global): In more advanced cases, we have to predicate about spatial properties which are non-local in the sense that the entities of interest may be arbitrarily distributed in space. Proximity and reachability, for instance, are two kinds of spatial relationships which play an important role in an evolving CPSp.

(temporal): Concerning the temporal dimension, we typically quantify over system states along one or several execution paths; i.e. we are interested in behavioral characteristics of certain events or system properties.

A plethora of approaches are actively investigated by the research community to support reasoning about properties of *one* of these kinds. For example, graphs and graph pattern matching [16] provide suitable methods to deal with local spatial properties, while model checking based on various forms of spatial [46] and temporal logics [14] provides a rigorous approach for the verification of global spatial and temporal system properties, respectively. However, there is a considerable lack of approaches covering *all* of the above listed kinds of properties at the same time. This is a significant deficiency concerning the engineering of dependable space-and-time-dependent systems, since the properties of interest are often *complex spatio-temporal* properties. Informally speaking, a complex spatio-temporal property refers to behavioral characteristics (temporal) of spatial relationships (spatial; global) of complex structures (spatial; local).

In this paper, we argue that software engineering (SE) can have a profound impact in engineering of space-and-time-dependent systems, by defining suitable modeling and specification notations as well as supporting design-time formal verification. The typical SE approach –provide a suitable model amenable for analysis and use it to validate a design– is applied to the domain of CPSp. Thus, we present a methodology and a technical framework which support *modeling* of evolving CPSp and *reasoning* about complex spatio-temporal properties of the overall systems. We contextify our approach within cyber-physical systems where the space –computational or physical– they operate in can be abstracted as a discrete and relational structure. This viewpoint is complementary to other approaches which target continuous aspects. Our modeling approach grounds on Bigraphs and Bigraphical Reactive Systems [37], a modeling formalism proposed by Robin Milner as a fundamental theory for structures in ubiquitous computing. The idea of bigraphs is based on two fundamental concepts of discrete spaces: *locality* and *connectivity*. Locality is expressed in terms of a tree-based containment hierarchy, while connectivity is expressed by a hypergraph orthogonal to the containment structure. Possible local reconfigurations are expressed as rewriting rules called reaction rules, yielding a Bigraphical Reactive System (BRS). As a first contribution, we extend bigraphical rewriting such that reaction rules can be equipped with non-local application conditions.

The main contribution of this paper is a technical framework for integrating several fundamental techniques to support reasoning about complex spatio-temporal properties of a BRS-based model of evolving space. Reasoning facilities are implemented adopting logic-based specification of properties and according model checking procedures. Locally bounded spatial properties are expressed as bigraphical patterns which may be composed through logical formulae, and bigraphical matching is used as a fundamental technique to locate the points in space where such formulae hold. Concerning checking of global spatial properties, we interpret a bigraphical model as a so-called closure space [26], paving the way for adopting a spatial logic for closure spaces along with the functionality of a corresponding model checker [11]. Similarly, concerning checking of temporal properties, state transition models can be obtained from a BRS, serving as the underlying evaluation model for model checking based on a temporal logic. We restrict the combination of the above components to be suited for engineering dependable systems such that complexity of spatio-temporal reasoning in evolving spaces is manageable and expressiveness is not compromised. We demonstrate the applicability of our approach using a disaster scenario in a smart city environment as a case study, and evaluate the scalability of verification procedures for typical spatio-temporal properties in experiments using city environments of varying model sizes.

The rest of the paper is structured as follows. Section 2 introduces a smart office environment as a motivating scenario serving as a running example throughout the paper. Section 3 presents our approach to modeling space using bigraphs as the underlying modeling formalism. Section 4 is dedicated to the adaptation of spatial logics for closure spaces to bigraphs. In Sec. 5, modeling of evolutionary dynamics of a space is illustrated, while in Sec. 6, we integrate the spatial and temporal dimension, introducing our framework for reasoning about complex spatio-temporal properties. Section 7 evaluates our approach using a case study; related work is considered in Sec. 8, and Sec. 9 concludes the paper.

2 MOTIVATING EXAMPLE

In this section, we introduce a smart office environment as a motivating scenario and serving as running example of an evolving CPSp throughout the paper. It is inspired by the context-aware printing system originally described in [21] and later used in different variations in [31]. We begin with a brief description of the static structure of the cyber-physical space and then consider its dynamics, i.e. possible ways in which the system may change over time. Finally, we introduce a spatio-temporal property which represents a possible requirement that should be verified on the design. The example is presented in an informal manner; a concrete instance as well as its formalization will be presented in the following sections.

The cyber-physical space consists of an office environment which contains rooms; rooms may be connected through doors, which are either locked or unlocked. Rooms may contain computers, printers, and users. A printer may print only one job at a time; all other jobs submitted to a printer are queued in the printer's spool. The following possible changes in the cyber-physical space constitute the dynamics of our example:

- *(jobToSpool):* A user may submit a print job to a print spool through a computer which is connected to the same network as the respective printer; this may occur through arbitrary hops over networked computers.
- *(jobToPrinter):* Jobs may be transferred from the print spool to its associated printer if there is currently no other job being processed.

Please note that other possible changes such as locking/unlocking doors or users moving in the physical space may be part of the example domain, but not part of our system model. Besides the static structure and dynamics presented, the following requirement needs to be fulfilled by the design of a concrete office environment:

> Users must always be able to print jobs, and the room where the respective printer is located must be reachable through unlocked doors to collect printouts.

The requirement is an example of a spatio-temporal property of the system; it contains elementary predicates of different kinds. It has the temporal feature of "always", implying assurance of a predicate over time. It also includes spatial features with a both locally bounded and global scope. The predicate "the room the respective printer is located", is a locally bounded spatial predicate; it refers to a relation between a room and a printer. The requirement also contains a globally scoped predicate of "reachability" of a room by a user in specific circumstances, i.e. "through unlocked doors".

3 MODELING SPACE

In the metaphysical relational theory of space [1], space is composed of relations between entities, and space cannot exist in the absence of matter. This fundamentally gives rise to a graph structure that represents a topology; entities are represented by nodes, while relations between entities are represented by edges. In the following, a space will arise from two kinds of relations between entities: *containment*, signifying that an entity is located within another, and *linking*, signifying that two entities are connected in some way.

3.1 Modeling Space with Bigraphs

Bigraphs [37] are a formalism for structures in ubiquitous computing, which deals with both containment and linking among entities. A *bigraph* consists of two graphs. A *place graph* is a forest, a set of trees defined over a set of nodes. A *link graph* is a hypergraph over the same set of nodes and a set of edges, each linking an arbitrary number of nodes. Connections of an edge with its nodes are called *ports*. Place and link graphs are orthogonal, and edges between nodes can cross locality boundaries. Nodes are typed; the node types are called *controls* in bigraphical terminology. Bigraphs provide a concise and yet very expressive way to model relations in space and they are depicted graphically in an intuitive way. The bigraphs presented here are pure (non binding) and concrete; placing and linking are independent structures, and nodes and edges have discrete identifiers [36]. To simplify the notation, we ignore arities of controls and details related to bigraph composition, i.e. inner and outer interfaces of bigraphs. What follows is a rather informal presentation as used in the scope of this paper; the interested reader may refer to the work of Milner [37] for complete definitions and proofs of the bigraphical theory.

$$\mathsf{P.Q} \qquad \textit{Nesting}\ (\mathsf{P}\ \textit{contains}\ \mathsf{Q}) \tag{1a}$$
$$\mathsf{P} \mid \mathsf{Q} \qquad \textit{Juxtaposition of nodes} \tag{1b}$$
$$-_{\mathsf{i}} \qquad \textit{Site numbered}\ \mathsf{i} \tag{1c}$$
$$\mathsf{K_w} \qquad \textit{Node with control}\ \mathsf{K}\ \textit{having ports}\ \mathsf{w} \tag{1d}$$
$$\mathsf{W} \parallel \mathsf{R} \qquad \textit{Juxtaposition of bigraphs} \tag{1e}$$

Bigraphs can be described in algebraic terms (Formulae 1a-1e) or with an equivalent rigorous graphical representation. P, Q, and K are controls of bigraph nodes; controls are names that define a node's type. Nodes can be structured hierarchically through the containment relationship, expressed in Formula 1a. Two nodes may be placed at the same hierarchical structure level, as shown in Formula 1b. Additionally, bigraphs can contain sites, a special kind of node (Formula 1c) that denotes a placeholder; sites can be used to indicate presence of unspecified nodes. Each node can be associated with a number of named ports. If a single node of a given type exists in the bigraph, the control uniquely identifies that node. Otherwise, we use a port name as a way to uniquely identify it. In Formula 1d the node identified by control K has port names w. Equally named ports in a formula are connected forming a hyper-edge, called *link* in the sequel. Moreover, we will use the wildcard ? to denote *any* port name(s). Bigraphs form rooted hierarchies; in Formula 1e, W and R indicate different roots.

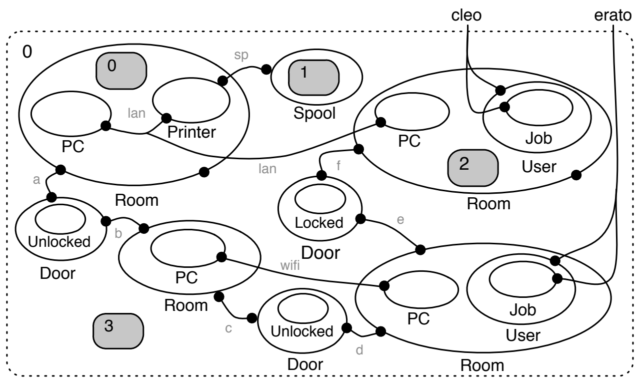

Figure 1: Bigraphical instance of a smart office environment.

Figure 1 shows a graphical representation of a bigraph modeling the space of a concrete instance of a smart office environment as introduced in Sec. 2. Room and User are examples of node controls signifying a node type. Containment is represented graphically by nesting a node inside another; User contains a Job. Sites, graphically represented as shaded boxes, denote the presence of unspecified nodes, as e.g. in the case of Spool which may contain an arbitrary number of Job nodes. Ports, graphically represented as black bullets, indicate linking of a node to a name – forming the link graph. They may be used to identify nodes of a certain type, e.g., cleo as a User. Ports can also be linked together to form named edges; for example, edge wifi between two PC nodes represents wireless connectivity between them. Finally, the dotted outer box graphically represents a root. Using the algebraic notation, the same bigraph of Fig. 1 can be partially represented as in Formula 2.

$$\begin{aligned}&\mathsf{Room_a.(PC_{lan} \mid Printer_{lan,sp} \mid -_0) \mid Spool_{sp}.(-_1) \mid}\\ &\mathsf{Door_{a,b}.(Unlocked) \mid Room_f.(User_{cleo}.(Job_{cleo}) \mid -_2) \mid -_3}.\end{aligned} \tag{2}$$

More formally, a bigraph arises from two superimposed relations: nesting and linking. Let V_B be a set of nodes and K the set of types of nodes, called controls. Let $ctrl_B : V_B \to K$ be a typing function, called control map. A *place graph* is a tuple:

$$B^P = (V_B, ctrl_B, prnt_B, K)$$

where $prnt_B : V_B \to V_B$ is an acyclic parent mapping modeling nesting. A *link graph* is a tuple:

$$B^L = (V_B, E_B, ctrl_B, link_B, K)$$

where E_B is a finite set of edges and $link_B : E_B \to 2^{V_B}$ is a link mapping assigning each edge the set of nodes which are connected by that edge. Finally, a bigraph B consists of B^P and B^L: $B = (V_B, E_B, ctrl_B, prnt_B, link_B, K)$.

4 REASONING ON SPACE

In this section, we present our approach to reasoning about spatial properties of (finite) bigraphical models of space, covering both locally bounded and global spatial properties. The approach is based on SLCS [11], an extension of the topological semantics of modal logics to closure spaces, a closure space being a generalization of a standard *topological* space [26]. In Sec. 4.1, we provide a context for evaluating spatial predicates, i.e. an evaluation model whereupon truth values of relations in the space described by a bigraph can be verified. In Sec. 4.2, we introduce syntax and semantics of our form of SLCS serving as a spatial logic for bigraphical models of space.

4.1 Evaluation Model of Space

As a prerequisite to reasoning about spatial properties, we stepwise develop an evaluation model throughout the remainder of this section. Since we ground our approach on SLCS, a spatial logic for closure spaces, we first recall basic definitions of closure spaces before we show how to automatically derive *closure spaces over bigraphs*. Thereupon, we define our notion of a *bigraphical closure model*, which is a bigraphical closure space equipped with a valuation function associating each point in the closure space with a set of atomic propositions that hold for that point.

4.1.1 Closure Spaces over Bigraphs. A closure space is a notion originating from the field of mathematical topology, built upon what can be informally referred as the "least possible enlargement" of a set. Formally, a closure space is a pair $(X, \mathcal{C})$ where X is a set, and the closure operator $\mathcal{C} : 2^X \to 2^X$ assigns to each subset A of X its closure, such that for all $A, B \subseteq X$:

$$\mathcal{C}(\emptyset) = \emptyset;\ A \subseteq \mathcal{C}(A) \text{ and } \mathcal{C}(A \cup B) = \mathcal{C}(A) \cup \mathcal{C}(B).$$

The elements of X are called the *points* of the closure space $(X, \mathcal{C})$. For any subset $A \subseteq X$, we define the *complement* of A in X as $\overline{A} = X \setminus A$. For each $A \subseteq X$, the *interior* $\mathcal{I}(A)$ of A is the set $\overline{\mathcal{C}(\overline{A})}$. Moreover, the *boundary* of a set informally refers to the set of elements which can be approached both from inside it and from outside of it. Formally, the boundary of $A \subseteq X$ is defined as $\mathcal{B}(A) = \mathcal{C}(A) \setminus \mathcal{I}(A)$. Two more variants of boundary exist, the *interior boundary* $\mathcal{B}^-(A) = A \setminus \mathcal{I}(A)$, and the *closure boundary* $\mathcal{B}^+(A) = \mathcal{C}(A) \setminus A$.

As shown in [11, 26], every graph for which the set of edges forms a binary relation induces a closure space, called a *quasi-discrete closure space*, by interpreting closure as the adjacency of nodes. Thus, the idea of obtaining a closure space from a bigraph, in the following referred to as a *bigraphical closure space*, is to transform a bigraph into a simple graph. Basically, this transformation follows the approach presented in [31]. Bigraphical nodes (excluding sites) are mapped to simple graph nodes, and the hyperlink structure is flattened such that every bigraphical link is represented by a dedicated node of the simple graph, this node being adjacent to the simple graph nodes corresponding to the bigraphical nodes which are connected by the link. In contrast to [31], which uses graphs with containment to represent node hierarchies, bigraphical nesting relationships are also represented by nodes in a simple graph to treat containment and linking in a uniform way.

Figure 2 shows an example of a simple graph obtained from the bigraph of Fig. 1, where nodes representing bigraphical nesting relationships are unlabelled. This simple graph may be interpreted as a bigraphical closure space, its nodes representing the points of the closure space. Closure, interior and boundary of a set of points can be intuitively obtained based on adjacency relationships. For example, if $A = \{lan\}$, we have $\mathcal{I}(A) = \emptyset$, $\mathcal{B}^-(A) = \{lan\}$ $\mathcal{C}(A) = \{Printer, PC, PC, lan\}$ and $\mathcal{B}^+(A) = \{Printer, PC, PC\}$.

Formally, let $B = (V_B, E_B, ctrl_B, prnt_B, link_B, K)$ be a bigraph, and B' with $V'_B \subseteq V_B$, $E'_B \subseteq E_B$, $prnt'_B \subseteq prnt_B$ and $link'_B \subseteq link_B$ its corresponding bigraph excluding sites, expressed by a mapping $excl_B : B \to B'$. The transformation of B into a corresponding simple undirected graph $G = (V, E)$, where V is its set of nodes and $E \subseteq \big\{\{x, y\} \mid x, y \in V\big\}$ is its set of edges, is defined for B', yielding a bijective mapping:

$$\tau' : V'_B \cup E'_B \cup prnt'_B \to V \tag{4}$$

for which the following two conditions hold:

- The nesting of places in B' coincides with the adjacency structure in G: $\forall\ (v'_1, v'_2) \in prnt'_B$ there are $v, v_1, v_2 \in V$ with $v = \tau'\big((v'_1, v'_2)\big)$, $v_1 = \tau'(v'_1)$, $v_2 = \tau'(v'_2)$ and we have $\{v, v_1\}, \{v, v_2\} \in E$.
- The linking structure in B' coincides with the adjacency structure in G: $\forall\ e' \in E'_B$ and $(e', v'_1), (e', v'_2), ..., (e', v'_n) \in link'_B$, there are $v, v_1, v_2, ..., v_n \in V$ with $v = \tau'(e')$, $v_1 = \tau'\big((e', v'_1)\big)$, $v_2 = \tau'\big((e', v'_2)\big)$, ..., $v_n = \tau'\big((e', v'_n)\big)$ and we have $\{v, v_1\}, \{v, v_2\}, ..., \{v, v_n\} \in E$.

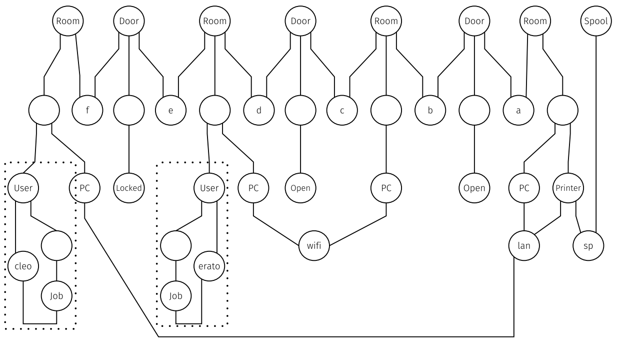

Figure 2: Bigraphical closure space obtained from the bigraph of Fig. 1, with occurrences of pattern User?.(Job?).

Given a simple graph $G = (V, E)$ obtained from B, i.e. $G = \tau(B)$ with $\tau \overset{def}{=} \tau' \circ excl_B$, a closure space $(V, \mathcal{C})$ may be derived from G where the closure operator $\mathcal{C}$ is obtained from E as in Formula 5.

$$\mathcal{C}(A) = A \cup \{x \in V \mid \exists\ a \in A : \{a, x\} \in E\} \tag{5}$$

4.1.2 Bigraphical Closure Model. Now that we have defined how to obtain closure spaces over bigraphs, a second prerequisite for defining a spatial logic is to associate each point $x \in V$ of a bigraphical closure space $(V, \mathcal{C})$ with a set of atomic propositions that hold for that point. A conventional approach, similar to the one taken in [11], would be to use labels of nodes in the simple graph $G = (V, E)$ as atomic propositions. Instead, we lift the specification of atomic propositions to the bigraphical level. The idea is that the developer specifies bigraphical patterns representing locally bounded spatial structures of interest, typically complex entities such as a *User* holding a print *Job*. Occurrences of such a pattern may be found in a bigraph B using bigraph matching [4, 28], and can be mapped to the bigraphical closure space induced by B. The set of atomic propositions that hold for a particular point $x \in V$ is finally defined as the set of pattern occurrences of which x is a part of. For example, for the bigraphical closure space of Fig. 2 obtained from the bigraph of Fig. 1, occurrences of the pattern p_1 = User?.(Job?) in the bigraphical closure space are identified by dotted rectangles. Thus, p_1 holds for every point being part of an occurrence of this pattern.

More formally, let P be the overall set of bigraph patterns specified by the developer. Moreover, given a pattern $p \in P$, let $O(B, p)$ be the set of all occurrences of this pattern in a bigraph B. For each occurrence $o \in O(B, p)$, there is a corresponding subgraph $\tau(o) \subseteq G$ in the simple graph $G = \tau(B)$ yielding the bigraphical closure space over B. We finally define a *bigraphical closure model*:

$$\mathcal{M} = \big((V,\ \mathcal{C}),\ v\big)$$

as a pair consisting of a bigraphical closure space $(V, \mathcal{C})$ induced by a graph $G = \tau(B)$ obtained from a bigraph B, and a valuation function $v : P \to 2^V$ assigning to each bigraphical pattern $p \in P$ the set of points in V which are part of an occurrence of p in B

mapped to G, i.e. for every $p \in P$:

$$v(p) = \big\{x \in V \mid x \in \bigcup_{o \in O(B,p)} \tau(o)\big\}.$$

4.2 Verification in Space

Building upon the previously defined fundamental operators of closure and boundary, we proceed to briefly outline the syntax and semantics of SLCS [11], a spatial logic for closure spaces. The logic will be evaluated upon bigraphical closure models $\mathcal{M}$, described previously. Essentially, a formula consists of propositions representing bigraphical patterns along with SLCS operators. The SLCS logic features boolean operators, a "one step" modality turning closure into a logical operator, and a surrounds operator. Given that p is drawn from a set of bigraph patterns P, the syntax of SLCS is defined by the following grammar:

$$\phi ::= p \mid \top \mid \neg\phi \mid \phi \wedge \psi \mid C\,\phi \mid \phi\,\mathcal{S}\,\psi. \tag{8}$$

In Formula 8, $\top$ denotes true, $\neg$ is negation, $\wedge$ is conjunction, C is the closure operator, and $\mathcal{S}$ is the spatial surrounds operator. Satisfaction $\mathcal{M}, x \models \phi$ of formula ϕ at point x in model $\mathcal{M} = ((V, C), v)$ is defined by induction on those terms:

$$\begin{aligned}
&\mathcal{M}, x \models p \Leftrightarrow x \in v(p);\\
&\mathcal{M}, x \models \top \Leftrightarrow \mathit{true};\\
&\mathcal{M}, x \models \neg\phi \Leftrightarrow \mathit{not}\ \mathcal{M}, x \models \phi;\\
&\mathcal{M}, x \models \phi \wedge \psi \Leftrightarrow \mathcal{M}, x \models \phi\ \mathit{and}\ \mathcal{M}, x \models \psi;\\
&\mathcal{M}, x \models C\,\phi \Leftrightarrow x \in C(y \in V \mid \mathcal{M}, y \models \phi);\\
&\mathcal{M}, x \models \phi\,\mathcal{S}\,\psi \Leftrightarrow \exists A \subseteq X : x \in A \wedge\\
&\qquad \forall y \in A : \mathcal{M}, y \models \phi \wedge \forall z \in B^{+}(A) : \mathcal{M}, z \models \psi.
\end{aligned}$$

More complex logical operators can be defined based on the fundamental operators of closure and surrounds. In the following, we illustrate those being useful in the context of this paper and later used for the evaluation of our approach in Sec. 7. First, we introduce a notion of *nearness*, i.e. two nodes in a bigraph which are either linked or in a nesting relationship. The corresponding points of such nodes in the bigraphical closure space can be found two steps away from another since linking and nesting relationships are represented by dedicated points (see Sec. 4.1). Since our aim is to specify spatial properties on the bigraph level, we define nearness by nesting applications of the closure operator:

$$\mathcal{N}\,\phi \overset{def}{=} C(C\phi).$$

Formula 11a is satisfied for the PC connected to the Printer in the upper left part of Fig. 1 since these nodes are linked to each other (due to the link graph). Similarly, Formula 11b is satisfied for the Printer since it is located in the printer Room (due to the place graph).

$$\mathsf{PC}_? \wedge\ \mathcal{N}\,\mathsf{Printer}_?. \tag{11a}$$

$$\mathsf{Printer}_? \wedge\ \mathcal{N}\,\mathsf{Room}_?. \tag{11b}$$

Operator $\mathcal{N}$ can be applied arbitrarily often to predicate about bigraph nodes being in a defined *proximity* from each other. Moreover, $\mathcal{N}$ can be also utilized to predicate about the nearness or proximity of complex entities specified as bigraphical patterns. We further consider operators being defined based on the spatial surrounds operator [11]; *weak reachability* ($\mathcal{R}$) and *strong reachability* ($\mathcal{T}$):

$$\phi\,\mathcal{R}\,\psi \overset{def}{=} \neg\big((\neg\psi)\,\mathcal{S}\,(\neg\phi)\big),$$

$$\phi\,\mathcal{T}\,\psi \overset{def}{=} \phi \wedge \big((\phi \vee \psi)\,\mathcal{R}\,\psi\big).$$

A point x satisfies $\phi\,\mathcal{R}\,\psi$ if either ψ is satisfied by x or there exists a sequence of points starting from x, all satisfying ϕ, leading to a point satisfying both ϕ and ψ. Utilizing the notion of weak reachability, we can define a strong reachability operator $\mathcal{T}$, where $\phi\,\mathcal{T}\psi$ is satisfied for a point x if it satisfies ϕ and we can reach a point satisfying ψ while passing only by points satisfying ϕ. Please note that ϕ and ψ are again specified on the bigraph level, i.e. using bigraph patterns. For example, consider Formula 14 which specifies a Spool being reachable from a Room in which a Printer is located. It is satisfied for the Room in the upper left corner of Fig. 1.

$$\mathsf{Room}_?(\mathsf{Printer}_? \mid -_0))\ \mathcal{R}\ \mathsf{Spool}_?. \tag{14}$$

Building upon these operators, we can further define a more complex "reach through" operator $\mathfrak{R}$ (Formula 15). It is satisfied for a point x if x satisfies ϕ and there is a sequence of points starting from x, all satisfying ψ, reaching a target point satisfying ζ.

$$\phi\,\mathfrak{R}(\psi)\,\zeta \overset{def}{=} \phi\,\mathcal{T}\,\big((\psi\,\mathcal{T}\,\zeta) \wedge (\psi\,\mathcal{T}\,\phi)\big). \tag{15}$$

For an example illustrating the use of the derived $\mathfrak{R}$ operator, consider Formula 16; it specifies that from the room in which User cleo is located, a room in which a Printer is located can be reached by traversing Rooms or unlocked Doors. Note that there is no point satisfying Formula 16 in the bigraphical closure model of Fig. 2, since the room in which User cleo is located has a locked door to another Room.

$$\begin{aligned}
&\mathsf{Room}_?.(\mathsf{User}_{\mathsf{cleo}}.\ -_0\ \mid -_1)\ \mathfrak{R}\\
&\big(\mathsf{Door}_?.(\mathsf{Unlocked}) \vee \mathsf{Room}_?.(-_2)\big)\ \mathsf{Room}_?.(\mathsf{Printer}_{\mathsf{sp}} \mid -_3).
\end{aligned} \tag{16}$$

So far, we have considered the satisfaction of an SLCS predicate at a particular point of a bigraphical closure model. In our model checking procedure, SLCS formulae are evaluated over the whole space represented by the bigraphical closure model obtained from a bigraph. Initially, following the model checking approach presented in [10], such an evaluation yields the set of points of the bigraphical closure space where the formula is true. We finally return a truth value which indicates the existence of points in the bigraphical closure model for which the evaluated formula is true, i.e. if the set obtained in the initial step is non-empty. Thus, model checking of Formulae 11a, 11b and 14 on our example evaluates to true, while Formula 16 evaluates to false.

5 MODELING DYNAMICS OF SPACE

Space is rarely static, thus a formalism for modeling evolving spatial systems should also capture system *dynamics* to enable reasoning about the effects of changes in space.

Bigraphical Reactive Systems (BRS) [37] extend bigraphs with well defined semantics of dynamic behavior expressed as a set of rules. BRS essentially allow describing possible ways in which the structure of the space can evolve through the application of transformation rules which selectively rewrite parts of a bigraph; they are called *reaction* rules. Reaction rules have the general form of $\mathsf{R} \longrightarrow \mathsf{R}'$, where R is called the *redex* and R′ is called the *reactum*; both the redex and reactum are bigraphs. If an occurrence of a redex can be found in the host bigraph, it may be replaced by the reactum, in a fashion similar to graph rewriting [17]. Redex and reactum can be considered as *patterns*, and the mechanism for finding occurrences of a redex is the same as the one outlined in the previous section for identifying parts of a bigraph that match a pattern. Intuitively, given an initial configuration of space specified by a bigraph and a set of reaction rules, new configurations may be generated by repeatedly applying reactions.

For example, the action *jobToPrinter* of our example smart office may be specified as in Formula 17. Utilizing the parameter matching facilities of the formalism through sites, the Job linked to name *n* is moved from the Spool to the respective Printer, while other entities found in the spool are not modified, as they are matched to the respective sites. By omitting specification of a site in the redex's Printer node, we would allow matching only when the printer does not contain other jobs. Moreover, by juxtaposing two bigraphs, we indicate that the Spool may reside in a different hierarchy; indeed, it is found outside a Room in Fig. 1.

$$\begin{aligned} &\mathsf{Printer_{sp}} \;\;\|\;\; \mathsf{Spool_{sp}.(Job_n \mid -_0)} \longrightarrow \\ &\qquad\qquad \mathsf{Printer_{sp}.(Job_n)} \;\;\|\;\; \mathsf{Spool_{sp}.(-_0)}. \end{aligned} \tag{17}$$

Observe that a reaction rule's redex specifies spatial, *local application conditions* only. However, we also need an extension of the bigraphical rewriting approach such that reaction rules can be equipped with additional non-local application conditions. To understand this requirement better, consider the change inherent in *jobToSpool* of our smart office example, where a user may submit a print job to a print spool through a computer which is connected to the same network as the respective printer. Here, connectivity of a computer to a printer may be established over multiple network hops. To specify such a redex in a conventional manner, one would have to enumerate every possible way in which a computer and printer are connected through an arbitrary number of nodes; something infeasible. To that end, we take advantage of the expressive power of SLCS predicates on space to specify spatial, *global application conditions*. Reaction rules are now permitted to take the form [condition] $\mathsf{R} \longrightarrow \mathsf{R}'$, where condition is a spatial predicate (Formula 8). If the pattern identified by the redex R is matched and the (global) application condition condition is satisfied, the redex is rewritten by the reactum. Note that there is no syntactic constraint on the specification of non-local application conditions of a reaction.

In Formula 18, such a global application condition is specified preceding the redex. Using the "reach through" operator previously defined, we can encode the change primitive *jobToSpool* as follows:

$$\begin{aligned} &\big[\mathsf{PC_{lan}}\; \mathfrak{R}(\mathsf{lan} \mid \mathsf{wifi} \mid \mathsf{PC_{lan}})\; \mathsf{Printer_{sp}}\big] \\ &\mathsf{PC_{lan}} \;\|\; \mathsf{Printer_{sp}} \;\;\|\;\; \mathsf{Spool_{sp}.(-_0)} \;\|\; \mathsf{User_n.(Job_n)} \longrightarrow \\ &\qquad\qquad \mathsf{PC_{lan}} \;\|\; \mathsf{Printer_{sp}} \;\;\|\;\; \mathsf{Spool_{sp}.(Job_n \mid -_0)} \;\|\; \mathsf{User_n}. \end{aligned} \tag{18}$$

6 REASONING ON EVOLVING SPATIAL CONFIGURATIONS

Since spatial configurations may change over time as discussed above, a formal model is needed to systematically treat change and enable reasoning on the changing system's behavior. In Sec. 6.1, we illustrate state transition models that can be obtained from a BRS, serving as the underlying evaluation model for reasoning on evolving configurations with a temporal logic, discussed in Sec. 6.2. Finally, we discuss complexity considerations in Sec. 6.3.

6.1 Evolving Spatial Configurations

Let us assume that a CPSp is specified by a BRS, which describes an initial configuration and a set of reaction rules describing its possible evolution. To enable automated reasoning, we transform the specification into an equivalent transition system generally known as a Kripke Structure (KS [14]) – a tuple $\mathsf{K} = (\mathsf{S}, \mathsf{T}, \mathsf{I}, \mathsf{BP}, \mathsf{L})$ where:

- S is a set of states describing configurations of space,
- $\mathsf{R} \subseteq \mathsf{S} \times \mathsf{S}$ is a set of transitions between states,
- $\mathsf{I} \subseteq \mathsf{S}$ is a set of initial states,
- BP is a set of atomic propositions,
- $\mathsf{L} : \mathsf{S} \longrightarrow 2^{\mathsf{BP}}$ is a function that labels each state with the set of propositions that are true in that state.

States of K describe bigraphical configurations of space, while transitions describe how the configuration of the system can change by moving from one state to its successors. Starting from a BRS specification, interpreting it over a KS means describing its evolution based on the application of reaction rules. The set of propositions that label a state can be systematically generated by declaratively encoding the corresponding bigraph configuration. Starting from an initial state ($i \in \mathsf{I}$) of the system representing an initial configuration, the BRS-based specification is interpreted by generating states according to the reactions. At each step, a set of successor states is produced yielding the branching Kripke Structure, where transitions reflect the changes of a bigraphical configuration over time. Each of these configurations representing a snapshot of space are encoded in states (through BP propositions). Since in each state generically more than one reaction can be applicable, K is nondeterministic. In practice, computation of L and R corresponds to the matching problem [4] and can be automated by configuring existing frameworks (e.g. [43],[40]). We further consider that transitions of K are non-blocking, i.e. $\forall s \in \mathsf{S}, \exists s' \in \mathsf{S} : (s, s') \in \mathsf{R}$.

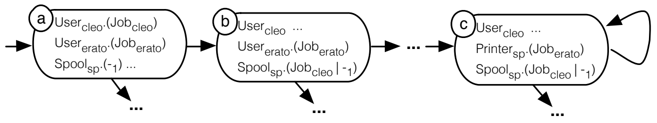
a
User$_{cleo}$.(Job$_{cleo}$)
User$_{erato}$.(Job$_{erato}$)
Spool$_{sp}$.(-$_1$) ...
b
User$_{cleo}$ ...
User$_{erato}$.(Job$_{erato}$)
Spool$_{sp}$.(Job$_{cleo}$ | -$_1$)
c
User$_{cleo}$ ...
Printer$_{sp}$.(Job$_{erato}$)
Spool$_{sp}$.(Job$_{cleo}$ | -$_1$)
...

Figure 3: KS fragment showing a violating sequence.

Figure 3 shows a fragment of a KS corresponding to the evolution of the example. Propositions on states encode bigraphical configurations; state *a* represents the initial configuration of Fig. 1, where Users have print jobs. In state *b*, a user's Job has been moved to the Spool due to the execution of a *jobToSpool* reaction. In state *c*, that Job is still in the Spool, and another user's Job is in the Printer, indicating some previous execution of a *jobToPrinter* reaction. Having obtained a KS representing the evolution of a configuration through the application of reaction rules, properties can be expressed in a temporal logic and formally verified.

6.2 Reasoning on Evolving Configurations

In the following, we describe how to predicate over the behavior of spatial properties of a bigraphical space over time. To that end, we discuss an integration of our approach to spatial reasoning with Linear Temporal Logic (LTL [14]), a logic with modalities referring to time. Please note that this integration is not binding; other temporal logics may also be used, and the approach as presented in the following is generically applicable. Let *LTL ∘ SLCS over bigraphs* be the spatio-temporal language according to the following definition:

$$\tau ::= p \mid \top \mid \neg\tau \mid \tau \wedge \tau \mid C\ \tau \mid \tau\ S\ \tau \tag{19a}$$

$$\phi ::= \tau \mid \neg\phi \mid \phi \vee \phi \mid \ \bigcirc \phi \mid \phi \cup \phi. \tag{19b}$$

Formulae 19a-19b describe the structure of terms and property formulae of the language. The language consists of temporal and spatial fragments; the spatial fragment corresponds to SLCS as introduced in Sec. 4, and the temporal fragment to LTL. Evaluation models are structures K as previously illustrated. In the combined logic, spatial predicates are terms, and formulae are constructed by intermixing them with temporal operators. Spatial predicates express spatial relations of bigraphs and are interpreted over states of K, while LTL properties express temporal properties and are interpreted over sequences of K and verified through model checking.

The semantics of a formula ϕ is defined over an infinite sequence σ of truth assignments to spatial propositions π resulting from spatial terms of Formulae 19a. We denote the set of propositions that are true at position i of the sequence by $\sigma(i)$. We recursively define whether sequence σ satisfies formula ϕ at position i (denoted $\sigma, i \models \phi$) as:

$\sigma,\ i \models \pi, \Leftrightarrow \pi \in \sigma(i);$

$\sigma,\ i \models \neg\phi, \Leftrightarrow \sigma,\ i\neg \models \phi;$

$\sigma,\ i \models \phi_1 \vee \phi_2, \Leftrightarrow \sigma,\ i \models \phi_1$ or $\sigma,\ i \models \phi_2;$

$\sigma,\ i \models \phi, \Leftrightarrow \sigma,\ i+1 \models \phi;$

$\sigma,\ i \models \phi_1 \cup \phi_2, \Leftrightarrow \exists k \geq i : \sigma, k \models \phi_2\ \wedge \forall i \leq j < k : \sigma, j \models \phi_1.$

Intuitively, the formula $\bigcirc\phi$ expresses that ϕ is true in the the next position in the sequence, and the formula $\phi_1 \cup \phi_2$ expresses the property that ϕ_1 is true until ϕ_2 becomes true. The sequence σ satisfies formula ϕ if $\sigma, 0 \models \phi$. Given fundamental temporal operators $\bigcirc$ ("next") and $\cup$ ("until"), we can derive additional ones such as $\diamond\phi = \mathit{true} \cup \phi$ ("eventually") and $\square\phi = \neg \diamond \neg\phi$ ("always"). Finally, we can encode an LTL property corresponding to the requirement of the example. The property specifies the fact that it should always be the case that should an $User$ named u exist in the space, eventually she is able to reach the printer (which contains her printout) through unlocked doors (or rooms):

$$\square(\mathsf{User_u} \rightarrow \diamond\mathsf{User_u}\ \mathfrak{R}(\mathsf{Room_?} \vee \mathsf{Door_?}.(\mathsf{Unlocked})\ \mathsf{Printer_?}.(\mathsf{Job_u}))\,.$$

The property specification is finally verified against the KS K describing the evolution of the system over time, discovering possible states that represent violations using standard explicit-state verification techniques, for which on-the-fly checking is also possible [27]. For the evaluation of state predicates of LTL formulae, the spatial fragment of our approach is invoked. Note that we can specify as usual any kind of an LTL property, where spatial predicates are terms. The sequence of states shown in Fig. 3 violates the property stated for user cleo, as the spatial term after the eventually operator (for $u : cleo$) is never true in the execution; intuitively, the Room she is located in Fig. 1 has a locked door, and thus she cannot reach the printer where her Job is located (state c of Fig.3).

6.3 Reasoning Complexity

When reasoning on evolving space-dependent systems as presented in this paper, certain considerations on complexity of the procedures involved must be taken into account. Specifically, one has to consider the size of the bigraphical model, the spatial predication over the bigraphical closure model, the rewriting component interpreting the BRS over a KS as well as temporal verification. For engineering realistic-sized systems, one can configure the constituents of our approach to obtain manageable complexity bounds.

Regarding the spatial fragment, the matching problem of bigraphs is related to the sub-graph isomorphism problem. Specifically, matching of place graphs can be reduced to sub-forest isomorphism; subsequently, a complete match can be computed by introducing extra constraints expressing matching of link graphs [43]. Both the sub-graph and sub-forest isomorphism problems are NP-hard [45]. However, note that when bigraphs involved have a single root, the matching problem can be reduced to an instance of sub-tree isomorphism, solvable in polynomial time. In practice, this means absence of juxtaposition of bigraphs in reaction rules (operator $\|$ of Formula 1e). Note that the reactions *jobToSpool* and *jobToPrinter* presented utilize multiple roots, since the Spool resides in a different hierarchy, the position of which may not be known. An alternative modeling strategy could put the spool inside its respective Printer, thus avoiding specification of a bigraph juxtaposition in a reaction rule. Regarding evaluation of spatial predicates, the complexity of SLCS evaluation is linear in the product of the number of nodes and edges of the closure model and size of the spatial formula [10, 11].

Several combinations of a spatial logic (such as SLCS) with a temporal logic are possible, with quite different properties of the resulting combination regarding expressiveness, decidability and complexity [24]. In the context of this paper, we focus on a combination where temporal operators can be applied to spatial terms, but not the other way round [23] i.e., spatial operators cannot be applied to temporal terms. The rationale behind this constrained interaction between spatial and temporal aspects is twofold. Firstly, it represents the elementary step in adding a temporal dimension to a spatial logic, and more complex interactions with varying properties can be considered further. Secondly, by such a composition satisfiability of the logic as presented in Formulae 19a-19b is PSPACE-complete [34]. Overall, this interaction has been chosen since it proved in our experiments to be sufficiently powerful to express requirements and it supports practical verification.

7 EVALUATION

To provide tool support and a proof-of-concept implementation of our approach to modeling and reasoning about evolving cyber-physical systems, we realized a prototypical toolchain integrating bigraphical matching and rewriting, the SLCS topochecker [11] and LTL model checking. Thereupon, we evaluate our approach using a scenario concerning UAV emergency response in a smart city as a case study (Sec. 7.1). Our evaluation goals are twofold: First, we focus on the *applicability* of our approach, modeling the smart city's space as well as its dynamics in Sec. 7.2, and demonstrating the reasoning facilities of our approach for two analysis scenarios exposing typical design challenges in Sec. 7.3. Second, we demonstrate the scalability of our reasoning framework and present experimental results obtained from the case study in Sec. 7.4. We conclude with a critical discussion in Sec. 7.5.

7.1 Case Study: UAV Emergency Response in a Smart City

Unmanned Airborne Vehicles (UAVs) can be used as radio relay platforms in environments characterized by poor connectivity. These environments can be regions where no global connectivity exists, e.g. due to a disaster or even absence of line of sight between ground transmitters and receivers. We consider a setting of UAV-carried communication infrastructure [53] in a disaster scenario for smart city applications such as emergency response. The setting we present, including the model and its dynamics is a generalized case [18] which can be concretized for a variety of urban warfare, search and rescue, homeland security or surveillance scenarios where autonomous UAVs operate in a space-dependent environment and global system properties need to be formally verified.

Communication is disabled in a city due to a disaster; search and rescue must be performed. Parts of the city may be unsafe, and victims may be stranded in various locations. Autonomous UAVs are dispatched to locate and provide communication infrastructure to victims, leading them to safety. This scenario fits our reasoning approach particularly well. This is because UAVs move in the city environment in specific ways, utilizing *global* knowledge of the city map and *local* knowledge (limited by e.g. line of sight) of positions of neighbouring UAVs and victims. UAVs carry short-range antennae, and victims are able to connect when they are in the vicinity. If a UAV is close to a victim, it can lead her to a safe zone. A safe zone is some part of the city which can lead to a hospital. To utilize our approach, the designer specifies the model, the ways UAVs can move and desired properties of the system, specification steps illustrated in the following.

7.2 Modeling Space and its Dynamics

We obtain a bigraphical model of space for our smart city case study in two steps. To obtain the basic topological structure of a city, we automatically extract a bigraph from city models described in CityGML [29], a widely used XML-based standard for the exchange of city models. Subsequently, further entities of interest such as UAVs and disaster victims are placed in that model. A conceptual representation of the topological structure extracted from a CityGML model with 20 buildings is illustrated in Fig. 4. A 2D projection of the roads and buildings is shown in light grey in the background, while the conceptual bigraphical structure is shown in the foreground. The bigraph exposes the following placing structure: A City node serves as root of the extracted bigraph. It contains nodes of type Road which in turn contain nodes of type RoadSegment and Crossroad, a road segment representing the part of a road between two crossroads. Moreover, a City node contains nodes of type Block, a block representing the area surrounded by road segments. Blocks may contain an arbitrary number of Building nodes, each one representing a building. Auxiliary nodes (e.g. for City and Road) are not shown in Fig. 4 for sake of readability. Likewise, we abstain from presenting conceptual node attributes, notably the function of a building (e.g. hospital, airport, residential, etc.). As for the linking structure of the extracted bigraph, each building is connected to the building next to it (represented by blue links in Fig. 4), and to a block's surrounding road segment if it is located in the respective block boundary (represented by green links). Moreover, road segments are linked to the crossroads being connected by that road segment (represented by red links).

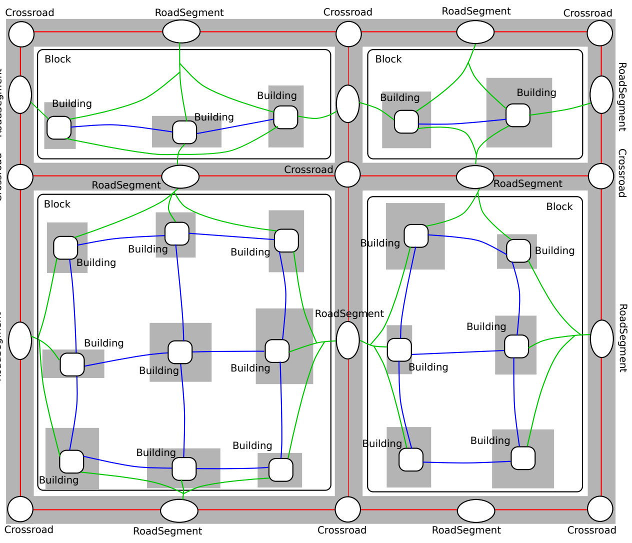

Figure 4: Bigraphical structure representation extracted from a city model with 20 buildings.

Subsequently, we model the changes inherent in the scenario. Movements of UAVs in the city environment must take into account key safety requirements, such as collision avoidance and boundary control [18]. UAV's may not fly over certain buildings (e.g. hospitals) and actively avoid collisions by avoiding to enter an area (e.g. a street) where another UAV is located. Specifically, UAVs generally move in accordance to the following rules; (i) from building to building, (ii) from building to road segment, (iii) from road segment (or crossroad) to another road segment (or crossroad) and (iv) from road segment to building. Conditions of movements include city no-fly zones and collision avoidance guards; no two UAVs should be over the same building, road segment or crossroad. Victims located by UAVs move with them until a safe zone is reached. Such changes are parametrically encoded in a BRS augmented with global application conditions.

7.3 Analysis Scenarios and Verification

In this section, we demonstrate the applicability of our reasoning facilities by considering two analysis scenarios of our case study, (A) verification of system requirements in early stages of design, and (B) analysis of simulation or historical output using trace checking.

7.3.1 Scenario A: Verification of System Requirements. Normally, UAVs follow some path planning strategy [42]; from all possible movements of a UAV at any point, a strategy selects the optimal, based on the strategy and local environmental conditions. Moreover, interesting problems arise with target search and surveillance scenarios, which can lead to complex controller algorithms [18]. We are not concerned with the design of a controller here, but with verifying properties of the system which concern *any* spatio-temporal decisions that the system of UAVs may take, operating in a city environment. Behaviors that may violate a global property of the system must be investigated, so every possible system behavior must be verified, possibly with an overall goal of using violating sequences to learn (or debug) a controller strategy. We consider a generic global requirement of the system, which states that if victims exist in the city, eventually all victims are safe. An LTL property encoding the requirement is found in Formula 21. Such a property can be used to validate the domain modeling, by verifying that the model and dynamics specified indeed lead to a valid system.

$$\begin{aligned}&\Box(\mathsf{Victim} \longrightarrow \diamond\neg\mathsf{Building}_{?}.(\mathsf{Victim} \mid -_0)) \\ &\wedge\ \mathsf{Victim}\ \mathfrak{R}(\mathsf{RoadSegment} \vee \mathsf{Crossroad})\ \mathsf{Hospital}.(-_1)).\end{aligned} \tag{21}$$

Note that verification in such a use case can bootstrap planning for adaptation; knowledge of the potential evolutions of the system can be a prerequisite for engineering adaptation at runtime. States in the evolution of the system where a critical situation manifests can be identified in advance, and countermeasures can be devised.

7.3.2 Scenario B: Trace Checking. In another scenario, given the model presented, we may be interested in verifying some property of an execution of the system. This execution may originate from historical behaviors of a deployed system or a simulation, as is typical in UAV controller design. In such a case, the evaluation model consists of a trace of spatial configurations and our approach can be used to verify a spatio-temporal trace property.

For instance, we may be interested in evaluating some UAV swarm planning algorithm, that claims to ensure that path plans for UAVs result in passes by points of interest of the space not too often. This "clumping" problem is encountered in a variety of domains [13, 35]. The property of Formula 22, describes the situation where a UAV_k in a Building is found near (with some degree of nearness *n*) to some other UAV_v, occurring twice while a Victim is not present. Given the equivalence $\psi \mathbin{\mathtt{W}} \phi = (\Box\psi) \vee (\psi \mathbin{\mathtt{U}} \phi)$ and relevant spatial predicates π and ξ below, we can encode it as bounded existence in LTL.

$$\begin{aligned}&\pi = (\mathcal{N}^n\,\mathsf{UAV}_\mathsf{k}.(-_2)) \wedge \mathsf{UAV}_\mathsf{v}.(-_1), \\ &\xi = \neg\mathsf{Building}_{?}.(\mathsf{UAV}_\mathsf{v}.(-_1) \mid \mathsf{Victim} \mid -_0) : \\ &\diamond\ \xi \longrightarrow (\neg\xi \mathbin{\mathtt{U}} (\xi \wedge (\neg\pi \mathbin{\mathtt{W}} (\pi \mathbin{\mathtt{W}} (\neg\pi \mathbin{\mathtt{W}} (\pi \mathbin{\mathtt{W}} \Box\neg\pi)))))).\end{aligned} \tag{22}$$

7.4 Experimental Results

To obtain suitable datasets for our experiments, we automatically generate CityGML models using Random3Dcity [3] and transform them into bigraphs, into which we randomly insert UAVs and victims. Bigraphical model sizes for three cities of different size are presented in Table 1; column *Bigraph* refers to the number of nodes and links in the corresponding bigraphs. The city models, BRS and property specifications are available at [49].

To conduct experiments for Scenario A, we interpret a BRS encoding UAV movements as a KS. At every state, for each possible movement of every UAV, the UAV performs the movement if possible, i.e. there is a match of the corresponding reaction rule's redex and global application conditions are fulfilled in the induced bigraphical closure model of the state. If a state reflecting the produced configuration does not exist in the KS, it is created. The process continues until no new states are produced – the KS fully interprets the BRS[1]. Subsequently, we proceed to verify the property of Formula 21. In our experimental setup, we consider 10 disaster victims along with 5 autonomous UAVs which roam in test cities of different sizes. Column *KS size* of Table 1 refers to states (and transitions) of the Kripke structure obtained from a BRS. Experiments were conducted on an Intel 2.5GHz processor; execution times are reported in column 4 of Table 1. Note that analysis times reflect the *total* time taken, which includes *i)* interpretation of the BRS augmented with global spatial application conditions, *ii)* spatial and *iii)* LTL model checking.

Table 1: Experimental setup and results of the UAV disaster scenario for system verification and trace checking.

	Bigraph	KS Size	Analysis Scenario A	Trace Size	Analysis Scenario B
city1	301	137(206)	256 sec	137	14 sec
city2	534	383(525)	9 min	383	55 sec
city3	670	1258(1845)	67 min	1258	307 sec

For Scenario B involving trace checking, we randomly generate a sequence of UAV movements for each city (column 5), using the same datasets and basic experimental setup as for Scenario A – the trace size in each city is intended to allow comparison with a KS of same state size. Spatio-temporal verification times of the properties of Formula 22 upon each trace are reported in column 6 of Table 1.

7.5 Discussion

We have demonstrated that by using our framework, powerful reasoning can be supported over the BRS-based model of an evolving space such as the smart city environment considered in our case study. By utilizing both global and local application conditions on reaction rules, we were able to encode complex primitives of change. Moreover, regarding the interplay of spatial predicates with temporal behavior, complex properties can be expressed. However, from our experience modeling a complex, realistic case study as the one presented and considering the perspective of practitioners aiming to use our approach, interfaces and tooling integration would go a long way in supporting specification. On the other hand, while our reference implementation is an unoptimized prototype, experimental results indicate feasibility for relevant models.

As evident by the diversity of performance results for full BRS interpretation and verification (of a KS – Scenario A) with respect to only verification (in the form of trace checking – Scenario B), we note that the biggest contributor to analysis overhead is the rate of change in the BRS. This signifies that our approach of engineering space-dependent systems is fit for design-time verification where time can largely not be an issue, and on the other hand suggests that trace checking (and runtime verification) is highly feasible.

8 RELATED WORK

In this paper, we presented a methodology and framework where the reasoning support is realized as an integration of fundamental techniques. Consequently, we primarily classify related work into two categories. First, we review theoretical foundations on formally modeling space and its dynamics, positioning our work. Thereafter, we discuss related approaches integrating spatio-temporal reasoning techniques with a focus on engineering dependable systems.

[1]It may also be bounded to a lookahead horizon, to address performance concerns.

Topological relations have been traditionally considered in the context of database systems, query languages [32] and logics for the spatial data analysis in Geographical Information Systems [5]. The focus has been in relations that exist between regions, lines, and points of a geometric model [19, 20]. Additionally, query languages for topological properties of spatial databases have been extensively studied [39], and recent approaches have focused on several aspects such as counting properties and path comparisons [22]. Conversely, in our conception space is discrete and is modeled as a specific graph structure arising from two relations that can be considered *topological*, between arbitrary entities. In addition, spatial logics have also been studied in the context of process calculi [8], where the typical theme is predication against the structure of agents, also with applications to graph databases [7]. However, our choice of using a spatial logic of closure spaces over some other logic is multifold. Closure spaces are a generic mathematical framework which for our purposes serves as the interface between arbitrary binary relations and our modeling formalism. Moreover, the choice of bigraphs, a formalism based on graph structures is motivated by its well-defined dynamics semantics; BRS (and graph transformation systems in general), can subsume other formalisms which deal with change such as process calculi, for engineering applications. Overall, all constituents of our approach are compatible with each other since they are based on elementary relations of entities, the basic building block being graphs inducing quasi-discrete spaces.

In our approach, for the spatial fragment we build upon the work of Ciancia et al. on spatial logics [11, 12], configuring the SLCS logic over evolving models of space built from bigraphical structures and used for components such as spatial model checking and rewriting application conditions. SLCS was proposed for quasi-discrete closure spaces, and applications have arisen on grid-like structures of space. However, the underlying model for evaluating the satisfaction relation in our case is a bigraphical closure model. Moreover, our contribution is the integration of the two ways of predicating; a powerful selection of entities through bigraph matching is used as propositions in the context of SLCS used to predicate over relations in space. An extension of [11] was used [10] for spatio-temporal reasoning, using CTL as the temporal fragment; however, in contrast to our work, the model of space at each state of the Kripke Structure is fixed, there is no concept of modeling dynamics and only primitive propositions (i.e., not patterns) are considered.

Bigraphical-specific Bilog [15] has been proposed as composition of two logics, a place graph and a link graph logic, enabling unified predication over both structures. A subset has been implemented [44], where a class of predicates is checked by reduction to bigraph matching. The closure space-based approach we utilize is generic, guarantees compatibility with the underlying graph representation and flexibility regarding existing tools and reasoning methods. In the approach reflected in this paper, graphs are found in states of a transition system representing behavior – this model is then checked against a logic specification. Graph-interpreted temporal logics in graph transition systems are studied in [25] generally, where a decision procedure is also provided. We note that compositional verification with graph transformation [30] would enable reasoning on incomplete behavioral specifications of evolving CPSp, something we identify as future work. Klein and Giese propose a visual approach for jointly specifying structural and temporal properties using Timed Story Scenario Diagrams [33], abstaining from use of both spatial and temporal logics by specifying temporal properties as sets of valid orderings of structural patterns, restricting the approach to specification of local spatial properties.

In addition to theoretical advances, CPS have motivated applications of spatio-temporal reasoning. The subject of topology of cyber and physical spaces has been previously studied. In previous work [52], we considered the Ambient Calculus [9], and moved to bigraphs as a meta-calculus allowing for more expressiveness and complex reconfiguration operators within adaptive security in cyber-physical systems [51]. However, spatial predicates used were locally bounded. In [47], a combination of metric and spatial logics has been proposed for verification of safety properties in CPS. In [46], a composition of a topologic and temporal logic is presented over hybrid automata. A combination of CTL and SLCS is developed [13] to study bike sharing systems while runtime verification of spatio-temporal behaviors of complex systems is studied in [38], extending Signal Spatio-Temporal Logic with SLCS. In contrast to these approaches, we focus on combining evolving spaces with spatio-temporal verification. However to the best of our knowledge, integration of spatio-temporal and evolutionary dimensions with a focus on engineering dependable systems has not been considered.

9 CONCLUSIONS

In this paper, we proposed a methodology and technical framework for engineering evolving cyber-physical spaces with a particular focus on modeling and verification. We used bigraphs and an extended form of Bigraphical Reactive Systems to model space and its dynamics. Concerning analysis and verification, such a reactive system is interpreted as a Kripke Structure representing an evolving space, thus enabling explicit-state verification of properties of interest. Such properties involve both spatial and temporal aspects. We restrict the combination of the components of our framework such that complexity of spatio-temporal reasoning in evolving spaces is manageable. We demonstrated the applicability of our approach by considering modeling and verification of a disaster scenario in a smart city environment, and presented experimental results of the scalability of the procedures involved for different city model sizes.

Although our work has been motivated by emerging problems in engineering dependable cyber-physical spaces, the notion of *discrete space* as treated in this paper may be considered as a fundamental concept and basic principle of abstraction in many areas of computing. Network topologies, object-oriented program structures or software architectures are traditional examples which inherently build on an abstract notion of discrete space. In fact, bigraphs and BRS have been adopted, e.g., for modeling cloud workload specifications [48], socio-technical and human behavior aspects [2], network protocols [6], and software architectures [41]. To that end, we believe that our approach can be applied in a much broader scope to generally tackle a considerable amount of the manifold challenges arising from engineering dependable evolving systems.

ACKNOWLEDGMENTS

This work was partially supported by ERC Advanced Grant no.227977 (SMScom). The authors would like to thank Vincenzo Ciancia, Michele Loreti and Rocco de Nicola for insightful discussions on spatial reasoning, and Filip Biljecki for his support on Random3Dcity.

REFERENCES

[1] Kaith Emerson Ballard. 1960. Leibniz's theory of space and time. *Journal of the History of Ideas* 21, 1 (1960), 49–65.

[2] Steve Benford, Muffy Calder, Tom Rodden, and Michele Sevegnani. 2016. On Lions, Impala, and Bigraphs: Modelling Interactions in Physical/Virtual Spaces. *ACM Trans. Comput.-Hum. Interact.* 23, 2, Article 9 (May 2016), 56 pages. DOI: http://dx.doi.org/10.1145/2882784

[3] Filip Biljecki, Hugo Ledoux, and Jantien Stoter. 2016. Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.* (2016), 51–59. DOI: http://dx.doi.org/10.5194/isprs-annals-III-4-W1-51-2016

[4] Lars Birkedal, Troels Christoffer Damgaard, Arne John Glenstrup, and Robin Milner. 2007. Matching of Bigraphs. *Electronic Notes in Theoretical Computer Science* 175, 4 (2007), 3–19.

[5] Roger S Bivand, Edzer Pebesma, and Virgilio Gómez-Rubio. 2013. *Spatial Data Import and Export*. Springer.

[6] Muffy Calder and Michele Sevegnani. 2014. Modelling IEEE 802.11 CSMA/CA RTS/CTS with stochastic bigraphs with sharing. *Formal Aspects of Computing* 26, 3 (2014), 537–561.

[7] Luca Cardelli and Luís Caires. 2001. A spatial logic for concurrency. In *TACS*, Vol. 1. 1–37.

[8] Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. 2002. A spatial logic for querying graphs. In *Automata, Languages and Programming*. Springer, 597–610.

[9] Luca Cardelli and Andrew D. Gordon. 1998. Mobile Ambients. In *Proc. of the 1st Int. Conf. on Foundations of Software Science and Computation Structure*. 140–155.

[10] Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. 2015. An experimental spatio-temporal model checker. In *International Conference on Software Engineering and Formal Methods*. Springer, 297–311.

[11] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. 2014. Specifying and verifying properties of space. In *Theoretical Computer Science*. Springer, 222–235.

[12] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. 2016. Model Checking Spatial Logics for Closure Spaces. *Logical Methods in Computer Science* 12, 4 (2016). DOI: http://dx.doi.org/10.2168/LMCS-12(4:2)2016

[13] Vincenzo Ciancia, Diego Latella, Mieke Massink, and Rytis Pakauskas. 2015. Exploring spatio-temporal properties of bike-sharing systems. In *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2015 IEEE International Conference on*. IEEE, 74–79.

[14] Edmund M Clarke, Orna Grumberg, and Doron A Peled. 1999. *Model Checking*. MIT press.

[15] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. 2005. *Spatial logics for bigraphs*. Springer.

[16] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence* 18, 03 (2004), 265–298.

[17] Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. 1997. Algebraic Approaches to Graph Transformation-Part I: Basic Concepts and Double Pushout Approach.. In *Handbook of Graph Grammars*. 163–246.

[18] Christopher M Eaton, Edwin KP Chong, and Anthony A Maciejewski. 2016. Multiple-Scenario Unmanned Aerial System Control: A Systems Engineering Approach and Review of Existing Control Methods. *Aerospace* 3, 1 (2016), 1.

[19] Max J Egenhofer, Andrew U Frank, and Jeffrey P Jackson. 1989. *A topological data model for spatial databases*. Springer.

[20] Max J Egenhofer and John Herring. 1990. Categorizing binary topological relations between regions, lines, and points in geographic databases. *The* 9 (1990), 94–1.

[21] Alexander Faithfull, Gian Perrone, and Thomas Hildebrandt. 2013. Big red: A development environment for bigraphs. In *Selected Revised Papers from the 4th Intl. Workshop on Graph Computation Models (GCM 2012)*, Vol. 61.

[22] Diego Figueira and Leonid Libkin. 2015. Path Logics for Querying Graphs: Combining Expressiveness and Efficiency. In *Logic in Computer Science (LICS), 2015 30th Annual ACM/IEEE Symposium on*. IEEE, 329–340.

[23] Marcelo Finger and Dov M Gabbay. 1992. Adding a temporal dimension to a logic system. *Journal of Logic, Language and Information* 1, 3 (1992), 203–233.

[24] David Gabelaia, Roman Kontchakov, Ágnes Kurucz, Frank Wolter, and Michael Zakharyaschev. 2005. Combining Spatial and Temporal Logics: Expressiveness vs. Complexity. *J. Artif. Intell. Res.(JAIR)* 23 (2005), 167–243.

[25] Fabio Gadducci, Reiko Heckel, and Manuel Koch. 1998. A fully abstract model for graph-interpreted temporal logic. In *International Workshop on Theory and Application of Graph Transformations*. Springer, 310–322.

[26] Antony Galton. 2003. A generalized topological view of motion in discrete space. *Theoretical Computer Science* 305, 1 (2003), 111–134.

[27] Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. 1996. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV*. Springer, 3–18.

[28] Arne John Glenstrup, Troels Christoffer Damgaard, Lars Birkedal, and Espen Højsgaard. 2008. An Implementation of Bigraph Matching. (2008).

[29] Gerhard Gröger, Thomas H Kolbe, Angela Czerwinski, Claus Nagel, and others. 2008. OpenGIS city geography markup language (CityGML) encoding standard, version 1.0. 0. (2008).

[30] Reiko Heckel. 1998. Compositional verification of reactive systems specified by graph transformation. In *International Conference on Fundamental Approaches to Software Engineering*. Springer Berlin Heidelberg, 138–153.

[31] Timo Kehrer, Christos Tsigkanos, and Carlo Ghezzi. 2016. An EMOF-Compliant Abstract Syntax for Bigraphs. *CoRR* abs/1612.01638 (2016).

[32] Jessie Kennedy, Peter Barclay, and others. 1996. A survey of query languages for geographic information systems. (1996).

[33] Florian Klein and Holger Giese. 2007. Joint structural and temporal property specification using timed story scenario diagrams. In *International Conference on Fundamental Approaches to Software Engineering*. Springer, 185–199.

[34] Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. 2007. Spatial logic+ temporal logic=? In *Handbook of spatial logics*. Springer, 497–564.

[35] Alexander G Madey and Gregory R Madey. 2013. Design and evaluation of UAV swarm command and control strategies. In *Proceedings of the Agent-Directed Simulation Symposium*. Society for Computer Simulation International, 7.

[36] Robin Milner. 2006. Pure bigraphs: Structure and dynamics. *Information and computation* 204, 1 (2006), 60–122.

[37] Robin Milner. 2009. *The Space and Motion of Communicating Agents*. Cambridge University Press.

[38] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti, and Mieke Massink. 2015. Qualitative and quantitative monitoring of spatio-temporal properties. In *Runtime Verification*. Springer, 21–37.

[39] Christos H Papadimitriou, Dan Suciu, and Victor Vianu. 1996. Topological queries in spatial databases. In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, 81–92.

[40] Gian Perrone, Søren Debois, and Thomas T. Hildebrandt. 2013. A Verification Environment for Bigraphs. *Innovations in Systems and Software Engineering* 9, 2 (2013), 95–104.

[41] Alejandro Sánchez, Luìs Soares Barbosa, and Daniel Riesco. 2013. Verifying bigraphical models of architectural reconfigurations. In *Theoretical Aspects of Software Engineering (TASE), 2013 International Symposium on*. IEEE, 135–138.

[42] Jesús Sánchez-García, José Manuel García-Campos, SL Toral, DG Reina, and Federico Barrero. 2016. An Intelligent Strategy for Tactical Movements of UAVs in Disaster Scenarios. *International Journal of Distributed Sensor Networks* 2016 (2016).

[43] Michele Sevegnani and Muffy Calder. 2015. Bigraphs with Sharing. *Theor. Comput. Sci.* 577 (2015), 43–73.

[44] Michele Sevegnani, Chris Unsworth, and Muffy Calder. 2010. *A SAT Based Algorithm for the Matching Problem in Bigraphs with Sharing*. Technical Report. University of Glasgow.

[45] Ron Shamir and Dekel Tsur. 1997. Faster subtree isomorphism. In *Theory of Computing and Systems, 1997., Proceedings of the Fifth Israeli Symposium on*. IEEE, 126–131.

[46] Zhucheng Shao and Jing Liu. 2013. *Spatio-temporal Hybrid Automata for Cyber-Physical Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 337–354. DOI: http://dx.doi.org/10.1007/978-3-642-39718-9_20

[47] Haiying Sun, Jing Liu, Xiaohong Chen, and Dehui Du. 2015. Specifying Cyber Physical System Safety Properties with Metric Temporal Spatial Logic. In *2015 Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 254–260.

[48] Christos Tsigkanos and Timo Kehrer. 2016. On Formalizing and Identifying Patterns in Cloud Workload Specifications. In *13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, Venice, Italy, April 5-8, 2016*. 262–267. DOI: http://dx.doi.org/10.1109/WICSA.2016.52

[49] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. 2017. Accompanied material and data for this paper. http://home.deib.polimi.it/tsigkanos/fse17. (2017).

[50] Christos Tsigkanos, Timo Kehrer, Carlo Ghezzi, Liliana Pasquale, and Bashar Nuseibeh. 2016. Adding static and dynamic semantics to building information models. In *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*. ACM, 1–7.

[51] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. 2016. On the Interplay Between Cyber and Physical Spaces for Adaptive Security. *IEEE Transactions on Dependable and Secure Computing* PP, 99 (2016), 1–1. DOI: http://dx.doi.org/10.1109/TDSC.2016.2599880

[52] Christos Tsigkanos, Liliana Pasquale, Claudio Menghi, Carlo Ghezzi, and Bashar Nuseibeh. 2014. Engineering Topology Aware Adaptive Security: Preventing Requirements Violations at Runtime. In *Proc. of the 22nd Int. Requirements Engineering Conf.* 203–212.

[53] Junfei Xie, Firas AI-Emrani, Yixin Gu, Yan Wan, and Shengli Fu. 2016. UAV-Carried Long Distance Wi-Fi Communication Infrastructure. In *AIAA Infotech@ Aerospace*. 0747.