UNIVERSITY
OF UDINE

Department of
Mathematics, Computer Science and Physics

MASTER THESIS IN
COMPUTER SCIENCE

Modeling and Verification of Cyber-Physical Spaces

CANDIDATE
Enrico Cesca

SUPERVISOR
Prof. Marino Miculan
(University of Udine)

CO-SUPERVISOR
Prof. Faber Wolfgang
(University of Klagenfurt)

Academic Year 2019-20

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
`https://www.dmif.uniud.it/`

In memory of my father *Fabio*, who always believed in me.
He has been the best father I could have.

Acknowledgements

I want to dedicate a few rows to all the people who have supported and helped me during my university path. I would first like to thank my thesis supervisor Prof. Marino Miculan of the Department of Mathematics, Computer Science and Physics of the University of Udine. I appreciated your availability, support, and patience in guiding me through this work. My gratitude extends to all the members of the Faculty of Computer Science of Udine for having allowed me to grow in this field of expertise.

I would also like to acknowledge Prof. Faber Wolfgang of the Department of Artificial Intelligence and Cybersecurity of the University of Klagenfurt, the second supervisor of this thesis. You and your colleagues gave me an incredible opportunity for personal and academic growth in Klagenfurt.

I must also thank my family, particularly my parents, for providing me with unfailing support and continuous encouragement throughout my years of study. Your dedication, unqualified love, sense of humor, and guidance have been more valuable than you can ever imagine. Finally, I would like to acknowledge my girlfriend for conveying to me strength and courage. Thanks for the patience you had.

Abstract

Nowadays, computers are ubiquitous, in the sense that they are everywhere. People live increasingly more in cyber-physical spaces, intended as hybrid environments of interrelated computational and physical entities. A suitable modeling approach and a formal verification have become necessary needs. This work presents a methodology to support the modeling of cyber-physical spaces and reasoning about their complex spatial properties. In particular, space, intended as discrete space, is modeled through a graph-based formalism, capable of describing, at once, the locations and the connections of components. Formal reasoning facilities are supported by the definition of a logic-based specification and a model checking procedure, allowing the verification of locally bounded and globally scoped properties of space. Finally, as a proof-of-concept, *jLibBig*, a Java library that can be used for modeling cyber-physical spaces, is extended with a spatial model checking procedure.

Contents

1 Introduction

Computing is transforming the world. Nowadays, artifacts that both calculate and communicate are permeating our lives, blurring the boundary between computational and physical worlds. Conceptually, such a hybrid environment is considered a *cyber-physical space* (CPSp), which consists of interrelated computational and physical entities. When dealing with cyber-physical spaces, modeling, as well as reasoning about space, are crucial challenges. Traditional verification techniques are well studied to analyze the temporal flavor, so the evolution of CPSp, while properties of space are typically not taken into account. The spatial properties of cyber-physical spaces can be classified into two kinds:

- *local* - properties of space locally bounded, which refer to direct, elementary relationships between entities or sets of entities, forming a structural pattern or anti-pattern;

- *global* - more advanced properties, like reachability, which are non-local in the sense that the entities of interest may be arbitrarily distributed in space.

The research community actively investigates several approaches to support reasoning about the properties of *one* of these kinds. For example, graphs and graph pattern matching [14] provide suitable methods for dealing with local spatial properties, while model checking based on various forms of spatial logic [26] provides a rigorous approach for verifying global properties. However, there is a considerable lack of approaches covering *both* of the above-listed kinds of properties simultaneously.

This work aims to support modeling of cyber-physical spaces and reasoning about complex spatial properties of the overall space. From a general viewpoint, the approach is partially similar to the one adopted in [27], but instead, every intermediate choice is different. In this work, a CPSp is intended as a discrete space, thus a structure that involves three concepts: *agent*, *locality*, and *connectivity*. In general, a discrete space gives rise to a graph structure in which nodes represent agents, locality denotes the placing of agents (including properties like adjacency and containment), and connectivity (which is locality independent) indicates the linking structure between nodes. The first choice, concerning the *modeling*, grounds on Directed Bigraphs [18], a bigraphical meta-model that generalizes both original Milner's [23] and Sassone-Sobocinski's [25] variants of bigraphs. Such graph-based formalism, capable of describing, at once, the locations and the connections of components, is appealing because it provides a range of general results and tools: libraries for bigraph manipulation [1, 8], graphical editors [15], etc.

The main contribution of this work is a method integrating several fundamental techniques for *reasoning* about complex spatial properties of a directed bigraphical model of cyber-physical space. Reasoning facilities are implemented adopting logic-based specification of properties and model checking procedures. The idea is to interpret directed bigraphical models as so-called *quasi-discrete closure spaces* [11], paving the way for adopting a spatial logic for closure spaces along with the functionality of a corresponding model checker [9]. In contrast to [27], the reasoning is based on a particular form of spatial logic for closure spaces called *spatial logic for directed bigraphs* (SLDB), which allows the specification of complex spatial properties of directed bigraphical models. Finally, as a proof-of-concept, the library for bigraph manipulation jLibBig [8] is extended with a spatial model checking procedure, adding the reasoning functionality to a library for modeling cyber-physical spaces [6].

The rest of the work is structured as follows. Section 1.1 introduces a smart office environment as a concrete example of cyber-physical space, used in the following chapters to show how modeling and reasoning are supported. Chapter 2 presents the approach to modeling space using directed bigraphs as the underlying modeling formalism. Chapter 3 shows how to reason about the space of directed bigraphical models. In particular, it describes the interpretation of space as a closure space, allowing the definition of an evaluation model. Moreover, it presents the new form of spatial logic for closure spaces, thought for specifying complex spatial properties of directed bigraphical models. Chapter 4 describes the implementation of the reasoning functionality in a library for modeling cyber-physical spaces, while Chapter 5 summarizes the contribution and draws directions for future work.

1.1 Cyber-Physical Spaces: Smart Office Example

This section introduces an example of cyber-physical space to show in action the modeling and reasoning approach presented in this work. Let consider an office environment inspired by the one of [27].

The cyber-physical space consists of a *smart office environment* that contains rooms; rooms may be connected through doors, which are either locked or unlocked. Each room may contain computers, printers, and users. A user may submit a job to a printer's spool through a computer that is connected to the same network as the respective printer; this may occur through arbitrary hops over networked computers. Jobs may be transferred from the spool to its associated printer if no other job is currently being processed. A printer may print only one job at a time, producing a document in the printer's tray. Besides the static structure, such a smart office needs to fulfill the following requirement:

- *Users must be able to print jobs, and the room where the respective printer is located must be reachable through unlocked doors to collect the printed documents.*

This requirement is an example of a complex spatial property of the cyber-physical space; it contains elementary predicates of different kinds. The predicate *the room where 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, a room's reachability under specific circumstances.

2

Modeling Space

Computing is transforming our world. Nowadays, artifacts that both calculate and communicate are permeating our lives. Thus, the term *ubiquitous computing* represents a vision that is being realized day by day, always more. To model ubiquitous systems of artifacts is hard. Since these systems may contain natural organisms, the modeling can also regard the interaction with and among humans, not only the interaction among artificial artifacts. Ubiquitous systems share a notion of *discrete space*, which involves just three concepts: *agent*, *locality*, and *connectivity* [24]. A discrete space gives rise to a graph structure where: nodes represent agents, locality denotes the *placing* of agents, including properties like adjacency and containment, and connectivity, which is locality independent, indicates the *linking* structure between nodes. Moreover, in such systems, placing and linking can be either physical, or virtual, or both. Section 2.1 describes *Directed Bigraphs*, a graph-based formalism for structures in ubiquitous computing, which deals with both containment and linking among entities. Section 2.2 presents our approach to modeling space using directed bigraphs, providing a graphical visualization of a directed bigraph modeling the smart office environment introduced in Section 1.1.

2.1 Directed Bigraphs

Directed bigraphs, introduced in [18], are a bigraphical meta-model that subsumes and generalizes both original Milner's [23] and Sassone-Sobocinski's [25] variants of bigraphs. Bigraphs are graph-like data structures capable of describing, at once, both the locations and the logical connections of components. Like an ordinary graph, a bigraph has nodes and edges, and the edges link the nodes. Unlike a standard graph, the nodes can be nested inside one another. Directed bigraphs are a generalization very suited for reasoning about *dependencies* and *request flows* between components. They have been used to design formal models of security protocols [17], access control [20], container-based systems [4], etc.

Directed bigraphs are composed of two orthogonal structures: a hierarchical *place graph* and a *directed link graph*, and each node is typed with a *control* of a polarized signature.

Definition 2.1.1 (Polarized Signature) *A* polarized signature *is a pair* $(\mathcal{K}, ar)$*, where* $\mathcal{K}$ *is the set of controls, and* $ar : \mathcal{K} \to \mathbb{N} \times \mathbb{N}$ *is a map assigning to each control its polarized arity, a pair* $\langle n, m \rangle$ *where* n, m *are the numbers of* positive *and* negative ports *of the control, respectively. Thus* $ar^+, ar^- : \mathcal{K} \to \mathbb{N}$ *are defined as shorthands for positive and negative ports of controls:* $ar^+ \triangleq \pi_1 \circ ar$*,* $ar^- \triangleq \pi_2 \circ ar$*.*

This definition of polarized signature also allows for *inward ports* in controls, whereas classical controls have only *outward ports* [23]. This turns up also in the definition of *points* and *handles*. The addition of negative ports enables us to represent more faithfully the dependencies between components.

To allow bigraph composition, a directed bigraph has an *inner* and an *outer* interface. An *interface* $I = \langle m, X\rangle$ is composed by a finite ordinal m, called *width*, and by a polarized interface $X = (X^-, X^+)$.

Definition 2.1.2 (Polarized Interface) *A* polarized interface *X corresponds to a pair (X^-, X^+), where X^- and X^+ are sets of names such that $X^- \cap X^+ = \emptyset$; the two sets are called* downward *and* upward *interfaces.*

It is assumed that names, node-identifiers, and edge-identifiers are drawn from free infinite sets, respectively $\mathcal{X}$, $\mathcal{V}$, and $\mathcal{E}$, disjoint from each other. A directed bigraph is composed of a *place graph* and a *directed link graph*. The following definition of place graph is the same as for Milner's bigraphs [23].

Definition 2.1.3 (Place Graph) *A hierarchical* place graph *$A : m \to n$ is a triple $A = (V, ctrl, prnt)$ having an inner face m and an outer face n, both finite ordinals. These index respectively the* sites *and* roots *of the graph. Furthermore, it has a finite set $V \subseteq \mathcal{V}$ of* nodes*, a* control map *$ctrl : V \to \mathcal{K}$ and an acyclic* parent map *$prnt : m \uplus V \to V \uplus n$.*

A directed link graph is a generalization of Milner's and Sassone-Sobocinski's link graphs. Input-linear (output-linear, respectively) link graphs are special cases of directed link graphs: restrict to empty upward (downward, respectively) interfaces. However, there are directed link graphs which are neither input-linear nor output-linear, nor any combination of these.

Definition 2.1.4 (Directed Link Graph) *A* directed link graph *$A : X \to Y$ is a quadruple $A = (V, E, ctrl, link)$ having an inner face X and an outer face Y, both finite subsets of $\mathcal{X}$, called respectively the* inner *and* outer names *of the directed link graph. Then, it as finite sets $V \subseteq \mathcal{V}$ of* nodes *and $E \subseteq \mathcal{E}$ of* edges*, a* control map *$ctrl : V \to \mathcal{K}$ and a* link map *$link : Pnt(A) \to Lnk(A)$ where*

$$Prt^+(A) \triangleq \sum_{v \in V} ar^+(ctrl(v)) \qquad Prt^-(A) \triangleq \sum_{v \in V} ar^-(ctrl(v))$$
$$Pnt(A) \triangleq X^+ \uplus Y^- \uplus Prt^+(A) \qquad Lnk(A) \triangleq X^- \uplus Y^+ \uplus E \uplus Prt^-(A)$$

with the following additional constraint:

$$\forall x \in Y^- \,.\, link(x) = y \implies y \notin Y^+ \tag{2.1}$$

The elements of $Pnt(A)$ are called the points *of A; the elements of $Lnk(A)$ are called the* handles *of A.*

The constraint (2.1) means that no downward outer name can be connected to an upward outer name; this definition is more accurate than the one with two additional constraints presented in [7].

Direct link graphs are graphically depicted similarly to ordinary link graphs, with the difference that edges are represented as vertices of the graph and not as hyper-arcs connecting points and names.

Intuitively, the basic idea of directed link graphs is to notice that names are not resources on their own, but only a way for denoting (abstract) resources, represented by the edges. Connections from controls to edges (through names) represent *resource requests* or *accesses*. Thus, we can discern a resource request flow, which starts from points (i.e., control ports), goes through names, and terminates in edges.

Finally, *directed bigraphs* are defined as the composition of standard place graphs and new directed link graphs. Figure 2.1 shows the fundamental anatomical elements of directed bigraphs.

Definition 2.1.5 (Directed Bigraph) *Let $I = \langle m, X\rangle$ and $O = \langle n, Y\rangle$ be two interfaces; a* directed bigraph *with polarized signature $\mathcal{K}$ from I to O is a tuple $G = (V, E, ctrl, prnt, link) : I \to O$ where:*

- *I and O are the* inner *and* outer *interfaces;*
- *V and E are the sets of nodes and edges;*
- *$ctrl$, $prnt$, $link$ are the* control, parent *and* link *maps;*

such that $G^L \triangleq (V, E, ctrl, link) : X \to Y$ is a directed link graph and $G^P \triangleq (V, ctrl, prnt) : m \to n$ is a place graph. G is denoted also as $\langle G^P, G^L\rangle$.

Note that Milner's bigraphs [23] correspond precisely to directed bigraphs with positive interfaces only and over signatures with only positive ports.

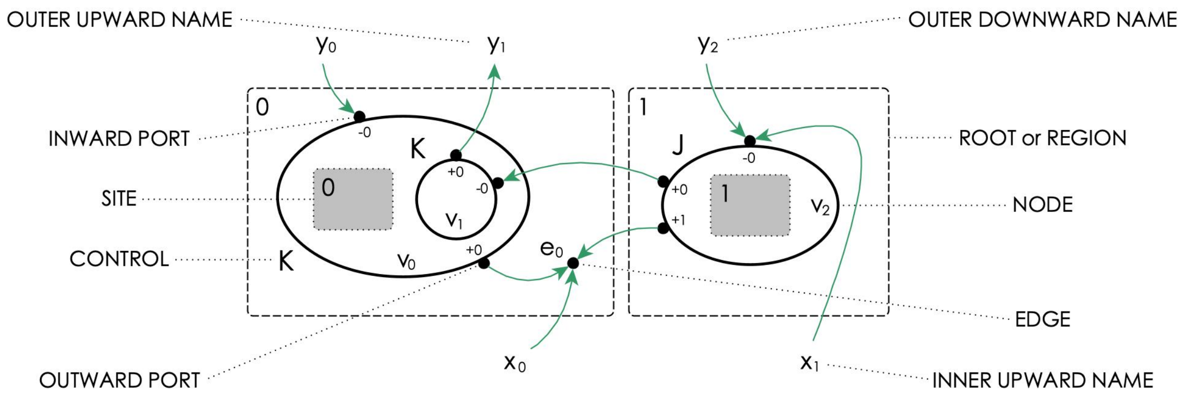

Figure 2.1: Anatomy of *Directed Bigraphs*.

Directed bigraphs can be described with a graphical representation or in an equivalent algebraic notation. What follows is an algebraic notation as used in the scope of this work (Formulas 2.2a-2.2f).

$-_i$	*site numbered* i	(2.2a)
$\mathsf{P}[\mathsf{p}_0]$	*node* p_0 *with control* P	(2.2b)
$\mathsf{P}[\mathsf{p}_0] \mid \mathsf{Q}[\mathsf{q}_0]$	*juxtaposition of nodes* p_0 *and* q_0	(2.2c)
$\mathsf{P}[\mathsf{p}_0].(-_0 \mid \mathsf{Q}[\mathsf{q}_0])$	*node* p_0 *contains site* $-_0$ *and node* q_0	(2.2d)
$\mathsf{P}[\mathsf{p}_0]_{\mathsf{x,y,z}}$	*node* p_0 *connected to handles* x, y *and* z	(2.2e)
$\mathsf{W} \parallel \mathsf{R}$	*juxtaposition of bigraphs* W *and* R	(2.2f)

P and Q are controls, while W and R represent directed bigraphs. Bigraphs can contain sites, a special kind of node (Formula 2.2a) that denotes a placeholder for unspecified nodes. Sites and nodes can be structured hierarchically through the containment relation, expressed in Formula 2.2d, or may be placed at the same hierarchical level (Formula 2.2c). Finally, each node can be connected to several handles: in Formula 2.2e, the node identified by p_0, with control P, is connected to the handles x, y, z.

2.2 Modeling Space with Directed Bigraphs

The adopted modeling approach grounds on *directed bigraphs*, a graph-based formalism for structures in ubiquitous computing. The idea of directed bigraphs is based upon 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 directed graph orthogonal to the containment structure. Figure 2.2 shows a graphical representation of a directed bigraph modeling the space of the smart office environment described in Section 1.1. Controls like Room, PC, User are examples of node types. Each control has its polarized arity, which defines the number of positive and negative ports. Sites, graphically represented as shaded boxes, denote the possible presence of unspecified nodes. In the case of Room, the sites indicate that there may be other PCs, Users, or in general, other nodes. Names, like secretary or director, may be used to identify particular nodes, while the name lan models an abstract resource that connects other entities. Note that the idea of modeling lan as a name rather than an edge allows referring to such resource. Finally, the dotted outer box graphically represents a root or region.

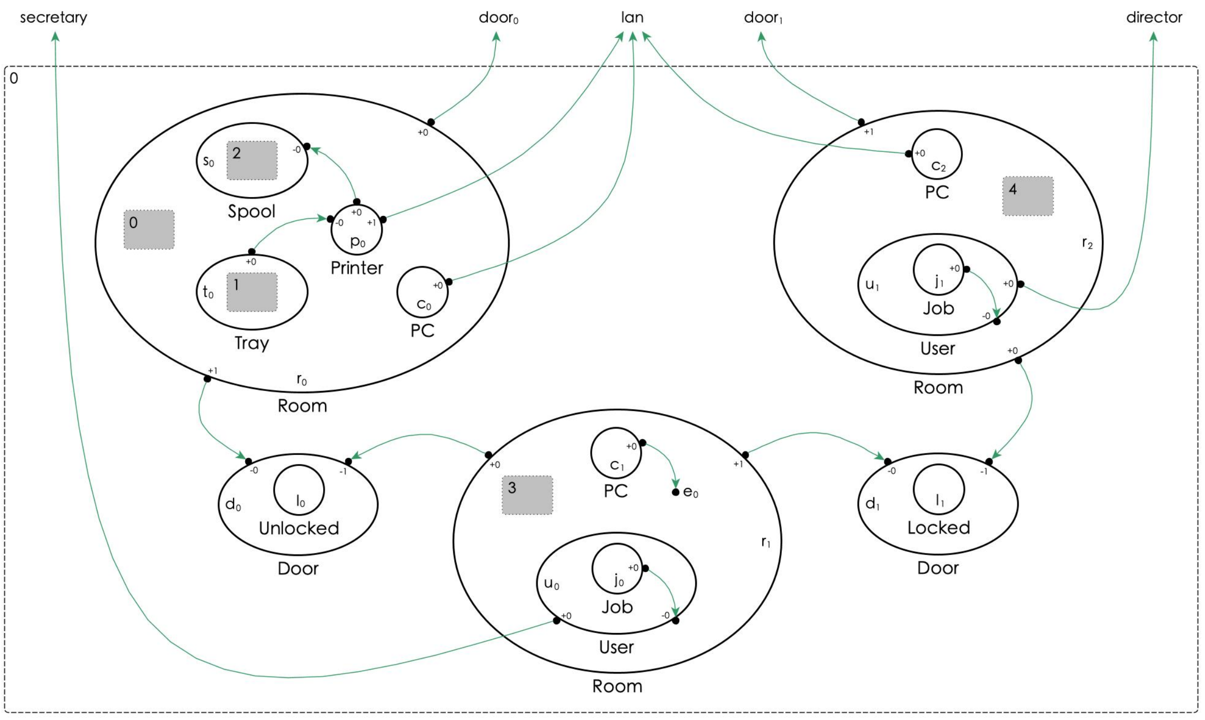

Figure 2.2: *Directed Bigraph* modeling the smart office environment of Section 1.1.

Using the algebraic notation defined in Section 2.1, the directed bigraph of Figure 2.2 can be represented as in Formula 2.3. Such notation can also be useful to represent directed bigraphical patterns.

$$\begin{aligned}&\mathsf{Room}[\mathsf{r}_0]_{\mathsf{door}_0,\mathsf{d}_0(0)}.\Big(-_0 \mid \mathsf{Tray}[\mathsf{t}_0]_{\mathsf{p}_0(0)}.(-_1) \mid \mathsf{Printer}[\mathsf{p}_0]_{\mathsf{s}_0(0),\mathsf{lan}} \mid \\ &\mathsf{Spool}[\mathsf{s}_0].(-_2) \mid \mathsf{PC}[\mathsf{c}_0]_{\mathsf{lan}}\Big) \mid \mathsf{Door}[\mathsf{d}_0].(\mathsf{Unlocked}[\mathsf{l}_0]) \mid \mathsf{Room}[\mathsf{r}_1]_{\mathsf{d}_0(1),\mathsf{d}_1(0)}.\Big(\\ &-_3 \mid \mathsf{PC}[\mathsf{c}_1]_{\mathsf{e}_0} \mid \mathsf{User}[\mathsf{u}_0]_{\mathsf{secretary}}.(\mathsf{Job}[\mathsf{j}_0]_{\mathsf{u}_0(0)})\Big) \mid \mathsf{Door}[\mathsf{d}_1].(\mathsf{Locked}[\mathsf{l}_1]) \mid \\ &\mathsf{Room}[\mathsf{r}_2]_{\mathsf{d}_1(1),\mathsf{door}_1}.\Big(\mathsf{PC}[\mathsf{c}_2]_{\mathsf{lan}} \mid \mathsf{User}[\mathsf{u}_1]_{\mathsf{director}}.(\mathsf{Job}[\mathsf{j}_1]_{\mathsf{u}_1(0)}) \mid -_4\Big)\end{aligned} \tag{2.3}$$

As defined in Section 2.1, a directed bigraph arises from two superimposed relations: *nesting* and *linking*. It could be interesting to consider each of the two resulting orthogonal structures separately.

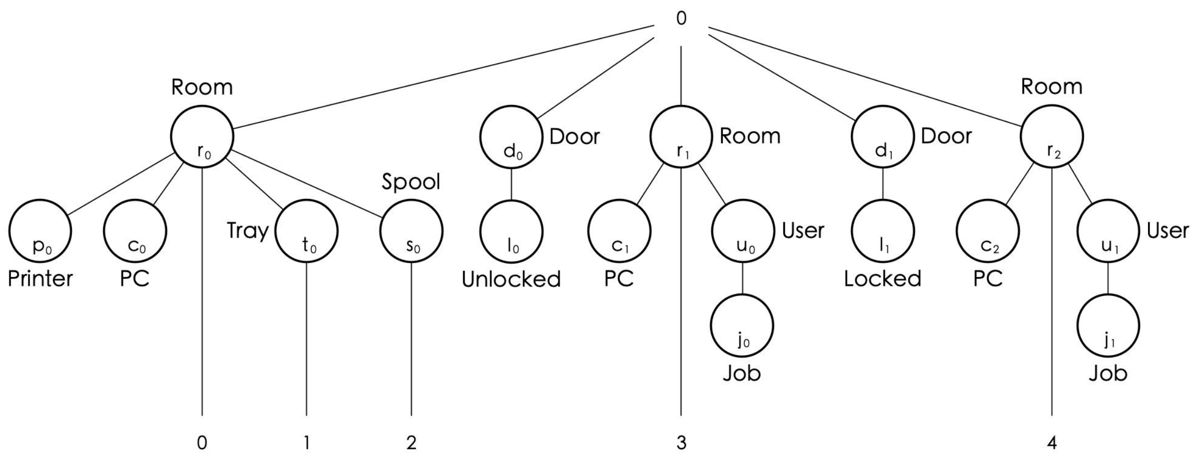

Figure 2.3: *Place Graph* derived from the directed bigraph of Figure 2.2.

Figure 2.3 shows a graphical representation of the *place graph*, while Figure 2.4 of the *directed link graph*, both derived from the directed bigraph of Figure 2.2, modeling the smart office environment.

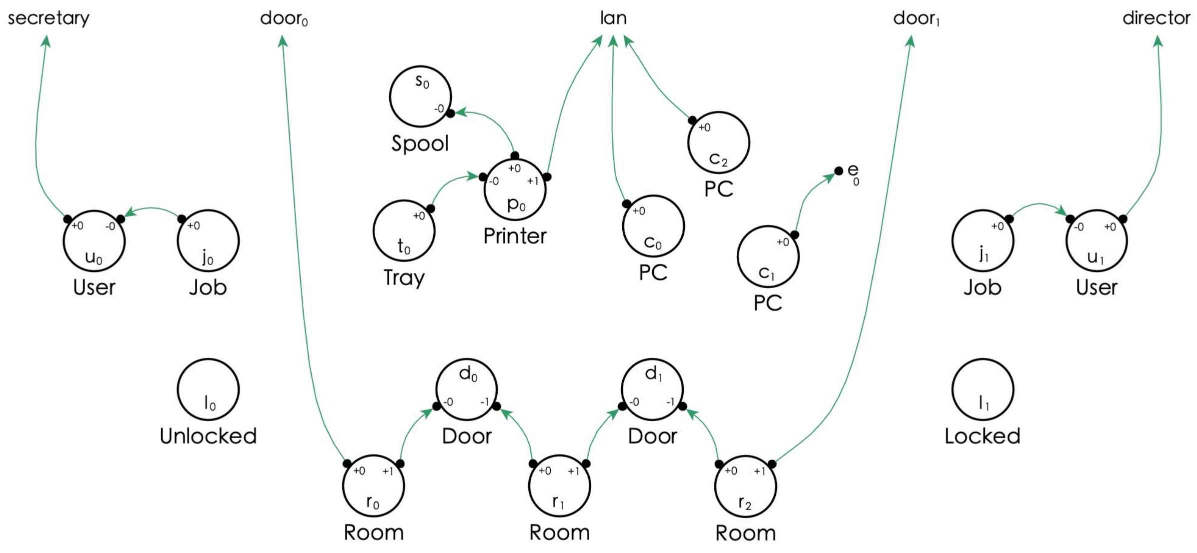

Figure 2.4: *Directed Link Graph* derived from the directed bigraph of Figure 2.2.

3 Reasoning on Space

This chapter describes how to reason about spatial properties of (finite) directed bigraphical models of space. The resulting approach, similar to the one adopted in [27], is based on a *spatial logic for closure spaces* (SLCS) [11]. In particular, SLCS is an extension of the topological semantics of modal logics to closure spaces. A closure space (also called *Čech closure space* or *preclosure space* in the literature) is a generalization of standard topological space, where idempotence of closure is not required. By using non-idempotent operators, graphs and topological spaces are treated uniformly, letting the topological and graph-theoretical notions of neighborhood coincide. Section 3.1 presents the most important aspects regarding closure spaces, most of which are explained in [16]. These concepts of *space* allow the definition of an evaluation model, providing a context for evaluating spatial properties. Thanks to this model, presented in Section 3.2, truth values of relations in the space described by a directed bigraph can be easily verified. Finally, Section 3.3 shows the syntax of a *spatial logic for directed bigraphs* (SLDB), a particular form of SLCS serving as a spatial logic for directed bigraphical models of space.

3.1 Quasi-Discrete Closure Spaces

Closure spaces were introduced by Čech [5]. A closure space is a notion originating from the field of mathematical topology, built upon what can be informally referred to as the "least possible enlargement" of a set. Formally, it is a set equipped with a *closure operator* obeying to certain laws.

Definition 3.1.1 (Closure Space) *A* closure space (*or* Čech 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 of* X *its* closure*, obeying to the following laws, for all* $A, B \subseteq X$*:*

1. $\mathcal{C}(\emptyset) = \emptyset$*;*
2. $A \subseteq \mathcal{C}(A)$*;*
3. $\mathcal{C}(A \cup B) = \mathcal{C}(A) \cup \mathcal{C}(B)$*.*

Closure spaces are a generalization of *topological spaces*. The axioms defining a closure space are also part of the definition of a *Kuratowski closure space*, which is one of the possible alternative definitions of a topological space. More precisely, a closure space is Kuratowski, therefore a topological space, whenever closure is *idempotent*, that is, also the axiom $\mathcal{C}(\mathcal{C}(A)) = \mathcal{C}(A)$ holds [11].

A number of ideas from the topological setting can be straightforwardly generalized to closure spaces. In the following, for $(X, \mathcal{C})$ a closure space, and $A \subseteq X$, let $\overline{A} = X \setminus A$ be the complement of A in X.

Definition 3.1.2 (Interior) *Let $(X, \mathcal{C})$ be a closure space, for each $A \subseteq X$:*

1. *the* interior $\mathcal{I}(A)$ *of A, dual to closure, is the set $\overline{\mathcal{C}(\overline{A})}$;*
2. *A is a* neighbourhood *of $x \in X$ if and only if $x \in \mathcal{I}(A)$;*
3. *A is* closed *if $A = \mathcal{C}(A)$, while it is* open *if $A = \mathcal{I}(A)$.*

As with topological spaces, the open sets are precisely the complements of the closed sets.

Lemma 3.1.1 *Let $(X, \mathcal{C})$ be a closure space, the following properties hold:*

1. *$A \subseteq X$ is open if and only if $\overline{A}$ is closed;*
2. *closure and interior are monotone operators over the inclusion order, that is:* $A \subseteq B \implies \mathcal{C}(A) \subseteq \mathcal{C}(B)$ *and* $\mathcal{I}(A) \subseteq \mathcal{I}(B)$*;*
3. *finite intersections and arbitrary unions of open sets are open.*

Next, the notion of *boundary*, which also applies to closure spaces, and two of its variants, namely the *interior* and *closure* boundary, can be defined.

Definition 3.1.3 (Boundary, Interior Boundary, Closure Boundary) *In a closure space $(X, \mathcal{C})$, the* boundary *of $A \subseteq X$ is defined as $\mathcal{B}(A) = \mathcal{C}(A) \setminus \mathcal{I}(A)$. The* interior boundary *is $\mathcal{B}^-(A) = A \setminus \mathcal{I}(A)$, and the* closure boundary *is $\mathcal{B}^+(A) = \mathcal{C}(A) \setminus A$.*

To this work's aim, it is essential to notice that a closure space may be derived starting from a *binary relation*, that is a *graph*. In particular, all finite spaces are in this form.

Definition 3.1.4 *Consider a set X and a relation $R \subseteq X \times X$. A closure operator is obtained from R as $\mathcal{C}_\mathcal{R}(A) = A \cup \{x \in X \mid \exists a \in A \,.\, (a, x) \in R\}$.*

One could also change Definition 3.1.4 so that $\mathcal{C}_\mathcal{R}(A) = A \cup \{x \in X \mid \exists a \in A \,.\, (x, a) \in R\}$, but this does not affect the presented theory. Indeed, one obtains the same results by replacing R with R^{-1} in statements of theorems that explicitly use R, and are not invariant under such change.

Proposition 3.1.1 *For any $R \subseteq X \times X$, the pair $(X, \mathcal{C}_\mathcal{R})$ is a closure space.*

Closure operators obtained by Definition 3.1.4 are not necessarily idempotent. Lemma 3.1.2 provides a necessary and sufficient condition. Let $R^=$ denote the least reflexive relation that includes R.

Lemma 3.1.2 *$\mathcal{C}_\mathcal{R}$ is idempotent if and only if $R^=$ is transitive.*

Proposition 3.1.2 *Given $R \subseteq X \times X$, in the space $(X, \mathcal{C}_\mathcal{R})$, we have:*

$$\mathcal{I}(A) = \{x \in A \mid \neg\exists a \in \overline{A} \,.\, (a, x) \in R\}$$
$$\mathcal{B}^-(A) = \{x \in A \mid \exists a \in \overline{A} \,.\, (a, x) \in R\}$$
$$\mathcal{B}^+(A) = \{x \in \overline{A} \mid \exists a \in A \,.\, (a, x) \in R\}$$

Closure spaces derived from a relation can be characterized as *quasi-discrete* spaces. The following result establishes an important relationship between neighborhoods and closure.

Definition 3.1.5 (Quasi-Discrete Closure Space) *A closure space $(X, \mathcal{C})$ is* quasi-discrete *if and only if one of the following equivalent conditions holds:*

1. *each $x \in X$ has a* minimal neighbourhood[1] N_x*;*
2. *for each $A \subseteq X$, $\mathcal{C}(A) = \bigcup_{a \in A} \mathcal{C}(\{a\})$.*

Theorem 3.1.1 *A closure space $(X, \mathcal{C})$ is quasi-discrete if and only if there is a relation $R \subseteq X \times X$ such that $\mathcal{C} = \mathcal{C}_R$.*

Example 3.1 Every graph induces a *quasi-discrete* closure space. For instance, consider the directed graph depicted in Figure 3.1. Let R be the binary relation induced by the graph edges, and let A and B denote respectively the set of nodes labelled with "A" and the set of nodes labelled with "B". The closure $\mathcal{C}_R(A)$ consists of all nodes labelled with "A" or with "B", so $\mathcal{C}_R(A) = A \cup B$. The interior $\mathcal{I}(A)$ corresponds to the subset of A composed of nodes with a *dashed border*, while the subet of A composed of nodes with a *solid border* corresponds to the interior boundary $\mathcal{B}^-(A)$. Finally, the closure boundary $\mathcal{B}^+(A)$ contains all nodes labelled with "B", formally $\mathcal{B}^+(A) = B$.

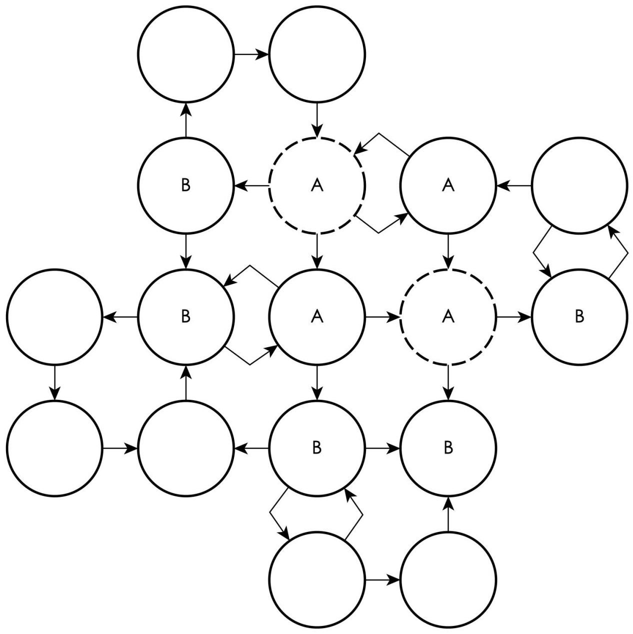

Figure 3.1: A graph inducing a *quasi-discrete* closure space.

Summing up, every graph (also directed graphs) induces a *quasi-discrete* closure space, where closure represents the "the least possible enlargement" of a set of nodes. Closure spaces are a generalization of *topological spaces*, for which topological notions like *interior* or *boundary* can be easily defined.

[1]A *minimal neighbourhood* of x is a set that is a neighbourhood of x (Definition 3.1.2) and is included in all other neighbourhoods of x.

3.2 Evaluation Model of Space

As a prerequisite to reasoning about spatial properties, it is necessary to develop an evaluation model. Section 3.2.1 describes how to derive closure spaces over directed bigraphs automatically and shows the closure space obtained from the directed bigraph of Figure 2.2 concerning the smart office environment. Thereupon, Section 3.2.2 defines the notion of *directed bigraphical closure model*, a directed bigraphical closure space equipped with a valuation function associating to each point of the closure space a set of atomic propositions that hold for that point.

3.2.1 Closure Spaces from Directed Bigraphs

As shown in Section 3.1, every graph for which the set of edges forms a binary relation induces a closure space, called a *quasi-discrete closure space*. Thus, the idea of obtaining a closure space from a bigraph, in the following referred to as a *directed bigraphical closure space*, is to transform a directed bigraph into a simple graph. Basically, this transformation is similar to the one presented in [27], but in this case, it results in a directed graph. Bigraphical nodes, sites, names, and edges are mapped to simple graph nodes. The *prnt* map, which defines the directed bigraph's hierarchical structure, is represented through other particular graph nodes, referred to as *places*. Figure 3.2 shows how the nesting relations of the pattern $\mathsf{P} \,.\, (\mathsf{CH_0}, \ldots, \mathsf{CH_n})$ are mapped into the simple graph. Let $\rho(\mathsf{P})$ denote the *place* node representing the nesting relations of P. Then, relations $(\mathsf{CH_0}, \mathsf{P}), \ldots, (\mathsf{CH_n}, \mathsf{P})$ of the *prnt* map are mapped to relations $(\mathsf{P}, \rho(\mathsf{P})), (\rho(\mathsf{P}), \mathsf{CH_0}), \ldots, (\rho(\mathsf{P}), \mathsf{CH_n})$ of the directed simple graph.

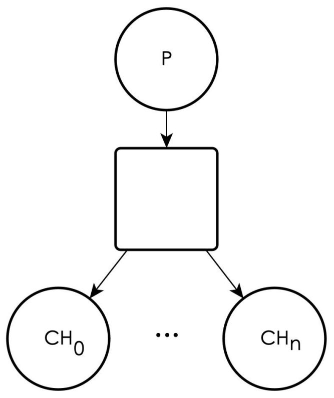

Figure 3.2: Representation of *nesting* relations in a *directed bigraphical closure space*.

In contrast to [27], the *link* map, which defines the linking structure of the directed bigraph, is transformed in a more complicated way. The resulting graph's structure is motivated by the need to represent all the necessary information to specify sophisticated spatial properties. In particular, the *link* map is represented through other particular nodes, referred to as *links*. Furthermore, to allow the specification of spatial properties involving reachability, it is also necessary to represent the inverse map $link^{-1}$, which is done through other particular nodes, referred to as *backs*. Figure 3.3 shows how the linking relations of the pattern $\mathsf{PT}_{\mathsf{H_0},\ldots,\mathsf{H_n}}$ are translated into the simple graph. Let $\ell(\mathsf{PT})$ and $\ell^{-1}(\mathsf{PT})$ denote respectively the *link* node and the *back* node associated to PT. Then, relations $(\mathsf{PT}, \mathsf{H_0}), \ldots, (\mathsf{PT}, \mathsf{H_n})$ of the *link* map are mapped to relations $(\mathsf{PT}, \ell(\mathsf{P})), (\ell(\mathsf{PT}), \mathsf{H_0}), \ldots, (\ell(\mathsf{PT}), \mathsf{H_n})$ and $(\mathsf{H_0}, \ell^{-1}(\mathsf{PT})), \ldots, (\mathsf{H_n}, \ell^{-1}(\mathsf{PT}))), (\ell^{-1}(\mathsf{PT})), \mathsf{PT})$ of the directed simple graph.

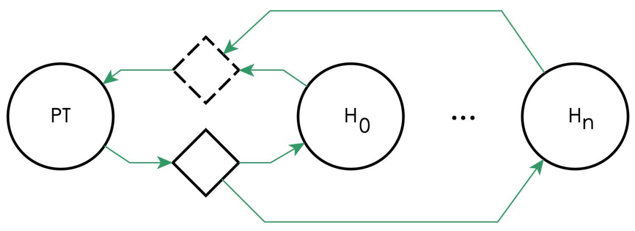

Figure 3.3: Representation of *linking* relations in a *directed bigraphical closure space.*

From now on, to simplify the visual understanding, the shape and the border style of nodes of the directed simple graph depend on the type of entity represented, as shown in Figure 3.4.

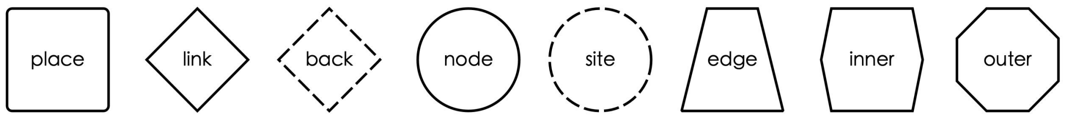

Figure 3.4: Node's *shapes* in a *directed bigraphical closure space.*

Definition 3.2.1 (Directed Simple Graph) *Let $I = \langle m, X\rangle$ and $O = \langle n, Y\rangle$ be two interfaces and $B = (V, E, ctrl, prnt, link) : I \to O$ a directed bigraph with polarized signature $\mathcal{K}$ from I to O. Let $\rho : n \uplus V \to \mathbb{N}$, $\ell : V \uplus X \uplus Y \to \mathbb{N}$ and $\ell^{-1} : V \uplus X \uplus Y \to \mathbb{N}$ be three bijective mappings denoting* place, link *and* back *nodes respectively. The* directed simple graph *$G = (N, A)$, where N is the set of nodes and $A \subseteq \{(x, y) \mid x, y \in N\}$ is the set of edges, is derived from B, yielding a bijective mapping:*

$$\tau : V \uplus m \uplus X \uplus Y \uplus E \uplus \rho \uplus \ell \uplus \ell^{-1} \to N$$

for which the following conditions hold:

- $\forall\ (ch, p) \in prnt : p \in n$ *there are* $x_1, x_2 \in N : x_1 = \tau(\rho(p))$, $x_2 = \tau(ch)$ *and* $(x_1, x_2) \in A$;
- $\forall\ (ch, p) \in prnt : p \in V$ *there are* $x_0, x_1, x_2 \in N : x_0 = \tau(p)$, $x_1 = \tau(\rho(p))$, $x_2 = \tau(ch)$ *and* (x_0, x_1), $(x_1, x_2) \in A$;

- $\forall\ (pt, h) \in link : pt \in ar^+(ctrl(v_0))$, $h \in ar^-(ctrl(v_1))$ *there are* $x_0, x_1, x_2, x_3 \in N : x_0 = \tau(v_0)$, $x_1 = \tau(\ell(v_0))$, $x_2 = \tau(\ell^{-1}(v_0))$, $x_3 = \tau(v_1)$, *and* (x_0, x_1), (x_1, x_3), (x_3, x_2), $(x_2, x_1) \in A$;
- $\forall\ (pt, h) \in link : pt \in ar^+(ctrl(v))$, $h \in X \uplus Y \uplus E$ *there are* $x_0, x_1, x_2, x_3 \in N : x_0 = \tau(v)$, $x_1 = \tau(\ell(v))$, $x_2 = \tau(\ell^{-1}(v))$, $x_3 = \tau(h)$, *and* (x_0, x_1), (x_1, x_3), (x_3, x_2), $(x_2, x_1) \in A$;
- $\forall\ (pt, h) \in link : pt \in X \uplus Y$, $h \in ar^-(ctrl(v))$ *there are* $x_0, x_1, x_2, x_3 \in N : x_0 = \tau(pt)$, $x_1 = \tau(\ell(pt))$, $x_2 = \tau(\ell^{-1}(pt))$, $x_3 = \tau(v)$, *and* (x_0, x_1), (x_1, x_3), (x_3, x_2), $(x_2, x_1) \in A$;
- $\forall\ (pt, h) \in link : pt \in X \uplus Y$, $h \in X \uplus Y \uplus E$ *there are* $x_0, x_1, x_2, x_3 \in N : x_0 = \tau(pt)$, $x_1 = \tau(\ell(pt))$, $x_2 = \tau(\ell^{-1}(pt))$, $x_3 = \tau(h)$, *and* (x_0, x_1), (x_1, x_3), (x_3, x_2), $(x_2, x_1) \in A$.

Figure 3.5 shows the directed simple graph obtained from the directed bigraph concerning the smart office environment of Figure 2.2. This graph may be interpreted as a *directed bigraphical closure space*: its nodes represent the points of the closure space. Topological notions of *closure*, *interior*, and *boundary variants* can be intuitively computed based on the adjacency relationships.

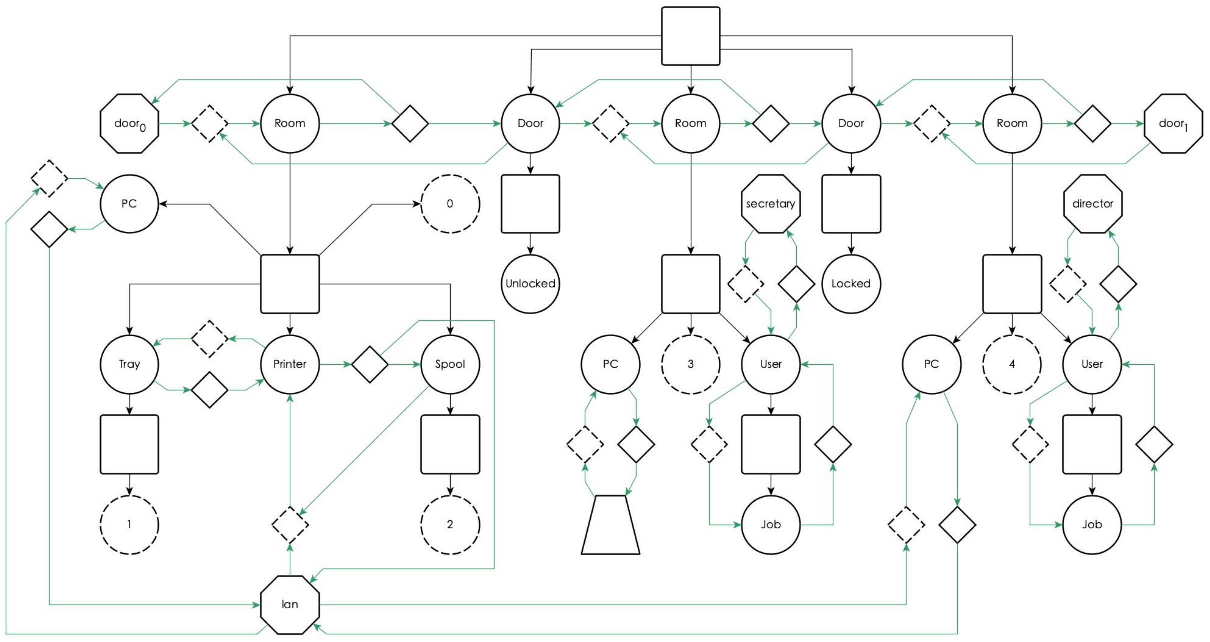

Figure 3.5: *Directed Bigraphical Closure Space* obtained from the Directed Bigraph of Figure 2.2.

Definition 3.2.2 (Directed Bigraphical Closure Space) *Let $B = (V, E, ctrl, prnt, link) : I \to O$ be a directed bigraph and $G = (N, A)$ a directed simple graph obtained from B. The* directed bigraphical closure space *is a pair $(N, \mathcal{C})$ where $\mathcal{C}$ is obtained from A as:*

$$\mathcal{C}(S) = S \cup \{n \in N \mid \exists\, s \in S : (s, n) \in A\}.$$

3.2.2 Directed Bigraphical Closure Model

In order to develop a closure model, the next step is to define a valuation function $\mathcal{V} : P \to 2^N$ that assigns to each point $x \in N$ of a directed bigraphical closure space $(N, \mathcal{C})$ the set of atomic propositions that hold for that point. Basically, the idea is that the valuation function $\mathcal{V}$ associates:

- $\forall\, x \in N$ a proposition of the set $\{place, link, back, node, site, inner, outer, edge\}$ that indicates the type of the node, according to Figure 3.4;
- $\forall\, x \in N$ with type *node* the control of the node;
- $\forall\, x \in N$ with type *inner* or *outer* the value of the name.

Example 2. To the nodes director and Spool of the directed bigraphical closure space of Figure 3.5 the valuation function $\mathcal{V}$ assigns the sets of propositions $\{outer, director\}$ and $\{node, Spool\}$ respectively.

Definition 3.2.3 (Directed Bigraphical Closure Model) *Let $I = \langle m, X \rangle$ and $O = \langle n, Y \rangle$ be two interfaces and $B = (V, E, ctrl, prnt, link) : I \to O$ a directed bigraph with signature $\mathcal{K}$. Let $(N, \mathcal{C})$ be the directed bigraphical closure space induced by the directed simple graph $G = (N, A)$ derived from B. Let $P = \mathcal{K} \uplus X \uplus Y \uplus \{node, site, inner, outer, edge, place, link, back\}$ be the set of atomic propositions. A* directed bigraphical closure model *$\mathcal{M} = ((N, \mathcal{C}), \mathcal{V})$ is a pair consisting of a directed bigraphical closure space $(N, \mathcal{C})$ and a valuation function $\mathcal{V} : P \to 2^N$ assigning to each proposition $p \in P$ a subset of N.*

3.3 Verification in Space

In Logics, there is quite an amount of literature focused on so-called *spatial* logics, a topological spatial interpretation of modal logics [28]. Dating back to early logicians such as Tarski, modalities may be interpreted using the concept of *neighborhood* in a topological space. The field of spatial logics is well developed in terms of descriptive languages and computability/complexity aspects. However, the frontier of current research does not yet address verification problems, and in particular, discrete models are still a relatively unexplored field. SLCS [11], a spatial logic for closure spaces, extends the topological semantics of modal logics to a more general setting, encompassing discrete, graph-based structures. Section 3.3.1 defines a *spatial logic for directed bigraphs* (SLDB), which consists of a particular form of SLCS with ad-hoc derived operators, thought to specify spatial properties over directed bigraphs. The idea is that such spatial properties are then translated from SLDB to SLCS and verified over a directed bigraphical closure model. Basically, this approach is the same adopted in [27], but with the difference that here space is modeled using directed bigraphs and the spatial logic is more expressive.

3.3.1 Spatial Logic for Directed Bigraphs

The *spatial logic for directed bigraphs* (SLDB) extends SLCS with some derived operators, thought to specify spatial properties over directed bigraphs. So before introducing the new ad-hoc logical operators of SLDB, it is necessary to outline the syntax and the semantics of SLCS.

The SLCS logic features two *spatial* operators: a "one step" modality, turning closure into a logical operator, and a binary *surround* operator, which is a spatial interpretation of the *until* operator. The spatial logical *surround* operator $\mathcal{S}$ is interpreted on points of a closure space. The basic idea is that point x satisfies $\phi\ \mathcal{S}\ \psi$ whenever it is included in an area A satisfying ϕ, and there is "no way out" from A unless passing through an area B that satisfies ψ. To turn this intuition into a mathematical definition, one should clarify the meaning of the words *area*, *included*, *passing*. More precisely, the SLCS formula $\phi\ \mathcal{S}\ \psi$ requires ϕ to hold at least on one point. The operator is similar to a *weak until* in temporal logics terminology, as there may be no point satisfying ψ, if ϕ holds everywhere.

Definition 3.3.1 (Syntax of SLCS) *The syntax of* SLCS *is defined by the following grammar, where p ranges over a set of* proposition letters *P:*

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

Here, $\top$ denotes true, $\neg$ is negation, $\wedge$ is conjuction, $\mathcal{C}$ is the *closure* operator, and $\mathcal{S}$ is the *surround* operator. Closure and operators of Figure 3.6 come from the tradition of topological spatial logics.

$$\bot \triangleq \neg\top \qquad \phi \vee \psi \triangleq \neg(\neg\phi \wedge \neg\psi) \qquad \Box\ \phi \triangleq \neg(\mathcal{C}\ \neg\phi)$$

$$\partial\ \phi \triangleq (\mathcal{C}\ \phi) \wedge (\neg\Box\ \phi) \qquad \partial^{-}\phi \triangleq \phi \wedge (\neg\Box\ \phi) \qquad \partial^{+}\phi \triangleq (\mathcal{C}\ \phi) \wedge (\neg\phi)$$

Figure 3.6: Derived operators of SLCS.

Figure 3.6 shows the definitions of some derived operators: besides the standard connectives, the *interior* ($\Box\,\phi$), the *boundary* ($\partial\,\phi$), the *interior boundary* ($\partial^{-}\phi$) and the *closure boundary* ($\partial^{+}\phi$).

Definition 3.3.2 (Semantics of SLCS) *Satisfaction $\mathcal{M}, x \models \phi$ of formula ϕ at point x in model $\mathcal{M} = ((N, \mathcal{C}), \mathcal{V})$ is defined by induction on terms, as follows:*

$$\begin{array}{lcl}
\mathcal{M}, x \models p & \Longleftrightarrow & x \in \mathcal{V}(p) \\
\mathcal{M}, x \models \top & \Longleftrightarrow & \mathit{true} \\
\mathcal{M}, x \models \neg\phi & \Longleftrightarrow & \mathcal{M}, x \not\models \phi \\
\mathcal{M}, x \models \phi \wedge \psi & \Longleftrightarrow & \mathcal{M}, x \models \phi \text{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \mathcal{C}\,\phi & \Longleftrightarrow & x \in \mathcal{C}(\{y \in X \mid \mathcal{M}, y \models \phi\}) \\
\mathcal{M}, x \models \phi\,\mathcal{S}\,\psi & \Longleftrightarrow & \exists A \subseteq X \,.\, x \in A \wedge \forall y \in A \,.\, \mathcal{M}, y \models \phi\, \wedge \\
 & & \quad \wedge\ \forall z \in \mathcal{B}^{+}(A) \,.\, \mathcal{M}, z \models \psi
\end{array}$$

By using the *surround* operator, operators concerning *reachability* can be derived. A point x satisfies $\phi\,\mathcal{R}\,\psi$ if and only if either ψ is satisfied by x or there exists a sequence of points after x, all satisfying ϕ, leading to a point satisfying both ψ and ϕ. In the second case, it is not required that x satisfies ϕ.

$$\phi\,\mathcal{R}\,\psi \stackrel{\triangle}{=} \neg((\neg\psi)\ \mathcal{S}\ (\neg\phi))$$

Thanks to reachability, a *strong reachability* operator $\mathcal{T}$ can be defined, such that $\phi\,\mathcal{T}\,\psi$ is satisfied for a point x if it satisfies ϕ and it can reach a point satisfying ψ while passing only by points satisfying ϕ.

$$\phi\,\mathcal{T}\,\psi \stackrel{\triangle}{=} \phi \wedge ((\phi \vee \psi)\ \mathcal{R}\ (\psi))$$

Finally, building upon these reachability operators, a more complex *reach through* operator $\Re$ can be defined. A point x satisfies $\phi\ \Re\ (\psi)\ \zeta$ if it satisfies ϕ and there is a sequence of points starting from x, all satisfying ψ, reaching a target point satisfying ζ.

$$\phi\ \Re\ (\psi)\ \zeta \stackrel{\triangle}{=} \phi\ \mathcal{T}\ (\psi\ \mathcal{T}\ \zeta)$$

The remaining part of this section defines the new ad-hoc derived operators of SLDB, thought for *directed bigraphical closure models*. These logical operators allow specifying spatial properties at the directed bigraph's level, so they remind the concepts of directed bigraphs.

The first group of ad-hoc derived operators concerns *locality*, expressed in the directed bigraph by its place graph. To understand the definitions of such operators, it is important to remember the directed bigraphical closure model construction described in Section 3.2. The nesting relations of the *prnt* map are represented through the use of *places*, particular nodes in which hold the atomic proposition *place*. Based on this observation, it is easy to note that each directed bigraph's entity could be connected in the directed bigraphical closure space with at most two places. Let *inner place* denote the entity's place, and *outer place* denote the place of the entity's parent. To define ad-hoc locality operators, it is necessary to use the following operators $\mathcal{I}$ and $\mathcal{O}$, concerning inner and outer places, respectively.

$$\mathcal{I}\ \phi \triangleq (\partial^{+}\phi) \wedge place \qquad\qquad \mathcal{O}\ \phi \triangleq place\ \mathcal{T}\ \phi$$

Using these auxiliary operators, it is possible to define the ad-hoc locality operators of Figure 3.7: the *parent* operator $\mathcal{P}$, the *siblings* operator $\mathcal{SI}$, the *children* operator $\mathcal{CH}$, and the *neighbors* operator $\mathcal{N}$.

$$\mathcal{P}\ \phi \triangleq node\ \mathcal{T}\ (\mathcal{O}\ \phi) \qquad\qquad \mathcal{SI}\ \phi \triangleq (\partial^{+}(\mathcal{O}\ \phi)) \wedge (\neg\phi)$$

$$\mathcal{CH}\ \phi \triangleq \partial^{+}(\mathcal{I}\ \phi) \qquad\qquad \mathcal{N}\ \phi \triangleq (\mathcal{P}\ \phi) \vee (\mathcal{SI}\ \phi) \vee (\mathcal{CH}\ \phi)$$

Figure 3.7: Ad-hoc *locality* operators of SLDB.

A point x satisfies $\mathcal{P}\ \phi$ if and only if it satisfies the proposition *node* and it is connected to a point satisfying $\mathcal{O}\ \phi$. Since the outer place of an entity corresponds to the place of the entity's parent, the only point connected to the outer place of x is the parent of x. Note that roots are not represented in the directed bigraphical closure space, so if it exists, the parent must satisfy the proposition *node*. A point x satisfies $\mathcal{SI}\ \phi$ if and only if it does not satisfy ϕ and it is part of the closure boundary of a point satisfying $\mathcal{O}\ \phi$. Being reachable from the outer place of an entity means having the same outer place of such an entity and so the same parent. It follows that the points having the same parent of an entity and being different from such an entity are the entity's siblings. A point x satisfies $\mathcal{CH}\ \phi$ if and only if it is part of the closure boundary of a point satisfying $\mathcal{I}\ \phi$. Since the inner place of an entity corresponds to the place used to represent the nesting relations with the entity's children, the closure boundary of the inner place corresponds to the entity's children. The *neighbors* operator $\mathcal{N}$ allows referring to the entity's neighborhood, including its parent, its siblings, and its children. The neighbors are entities considered to be near in terms of locality, ignoring the type of relationship involved.

Example 3. The following formulas use some of the *locality* operators of Figure 3.7. Formula 3.1a is satisfied by the PCs in the same Room of a User, so by PC[c$_1$] and PC[c$_2$] in Figure 2.2. The second property, Formula 3.1b, is satisfied by the Rooms that contain a Printer, so by Room[r$_0$] in Figure 2.2.

$$\mathsf{PC} \wedge (\mathcal{SI}\ \mathsf{User}) \tag{3.1a}$$

$$\mathsf{Room} \wedge (\mathcal{P}\ \mathsf{Printer}) \tag{3.1b}$$

The second group of ad-hoc derived operators concerns *connectivity*, expressed in the directed bigraph by its link graph. In particular, the linking relations of the *link* map are represented in the directed bigraphical model through the use of *links*, particular nodes in which hold the atomic proposition *link*. Furthermore, it is important to remember that also the relations of the inverse map $link^{-1}$ are represented in the directed bigraphical model. Such relations, represented through the use of *backs*, are necessary to allow the specification of properties involving reachability. Generally, an entity, connected to at least another entity in the directed bigraph, is connected to a *link* and is reachable from a *back* in the directed bigraphical closure space. To define one of the ad-hoc connectivity operators, it is necessary to use the following operator $\sim$, which allows referring to an entity, including its *link* and its *back*.

$$\sim \phi \triangleq \phi \vee ((\partial^+\phi) \wedge link) \vee (back\ \mathcal{T}\ \phi)$$

Figure 3.8 shows the definitions of the ad-hoc connectivity operators: the *points* operator $\mathcal{PT}$, the *handles* operator $\mathcal{H}$, and the *connectedThrough* operator $\mathcal{K}$.

$$\mathcal{PT}\ \phi \triangleq \partial^+((\partial^+\phi) \wedge back)$$
$$\mathcal{H}\ \phi \triangleq \partial^+((\partial^+\phi) \wedge link)$$
$$\phi\ \mathcal{K}\ (\psi)\ \zeta \triangleq \phi \wedge ((\sim \phi)\ \Re\ (\sim \psi)\ (\sim \zeta))$$

Figure 3.8: Ad-hoc *connectivity* operators of SLDB.

A point x satisfies $\mathcal{PT}\ \phi$ if and only if it is part of the closure boundary of the *back* of a point satisfying ϕ. Being one of the *points* of an entity means having a connection to such entity in the directed bigraph. It follows that the *points* of an entity are reachable via a relation of the inverse map $link^{-1}$. A point x satisfies $\mathcal{H}\ \phi$ if and only if it is part of the closure boundary of the *link* of a point satisfying ϕ. Being one of the *handles* of an entity means having a connection from such entity in the directed bigraph. It follows that the *handles* of an entity are reachable via a relation of the *link* map. Finally, a point x satisfies $\phi\ \mathcal{K}\ (\psi)\ \zeta$ if and only if it satisfies ϕ and there is a sequence of points starting from x, all satisfying ψ, reaching a target point satisfying ζ. Note that, by using the auxiliary operator $\sim$, reachability considers relations of both the *link* map and the inverse map $link^{-1}$.

Example 4. The following formulas use some of the *connectivity* operators of Figure 3.8. Formula 3.2a is satisfied by the PCs connected to lan, so by PC[c$_0$] and PC[c$_2$] in Figure 2.2. The second property, Formula 3.2b, is satisfied by the handles of Users, so by the names secretary and director in Figure 2.2.

$$\mathsf{PC} \wedge (\mathcal{PT}\ \mathsf{lan}) \tag{3.2a}$$
$$\mathcal{H}\ \mathsf{User} \tag{3.2b}$$

To sum up, the following definition describes the syntax of the *spatial logic for directed bigraphs*. Note that the set P of atomic propositions depends on the considered directed bigraph.

Definition 3.3.3 (Syntax of Spatial Logic for Directed Bigraphs) *Let $I = \langle m, X\rangle$, $O = \langle n, Y\rangle$ be two interfaces and $B = (V, E, ctrl, prnt, link) : I \to O$ a directed bigraph with polarized signature $\mathcal{K}$ from I to O. Let $P = \mathcal{K} \uplus X \uplus Y \uplus \{node, site, inner, outer, edge\}$ be a set of atomic propositions. The syntax of* SLDB *is defined by the following grammar, where p ranges over the set P:*

$$\phi ::= p \mid \top \mid \perp \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \mathcal{P}\ \phi \mid \mathcal{SI}\ \phi \mid \mathcal{CH}\ \phi \mid \mathcal{N}\ \phi \mid \mathcal{PT}\ \phi \mid \mathcal{H}\ \phi \mid \phi\ \mathcal{K}\ (\psi)\ \zeta$$

Here, $\top$ denotes true, $\perp$ denotes false, $\neg$ is negation, $\wedge$ is conjuction, $\vee$ is disjunction, $\mathcal{P}$ is the *parent* operator, $\mathcal{SI}$ is the *siblings* operator, $\mathcal{CH}$ is the *children* operator, $\mathcal{N}$ is the *neighbors* operator, $\mathcal{PT}$ is the *points* operator, $\mathcal{H}$ is the *handles* operator, and $\mathcal{K}$ is the *connectedThrough* operator.

To understand the expressive power of this spatial logic, the following examples define some spatial properties thought over the Smart Office Environment of Figure 2.2.

Example 5. Suppose to verify if in the Smart Office Environment there are PCs disconnected to lan that are in the same Room of a User with a Job to do. Such computers need to be connected because otherwise, they are useless. This spatial property, expressed in Formula 3.3, is satisfied by PC[$\mathsf{c_1}$].

$$\mathsf{PC} \land (\neg \overbrace{\mathcal{PT}\ \mathsf{lan}}^{\textit{points connected to } \mathsf{lan}}) \land \underbrace{(\mathcal{SI}\ (\mathsf{User} \land (\mathcal{P}\ \mathsf{Job})))}_{\textit{in a room with } \mathsf{User.(Job)}} \tag{3.3}$$

Example 6. Suppose to verify if in the Smart Office Environment there are Users, with a Job to do, that are in the same Room of a PC connected via lan to a Printer. Such users are lucky because they can work. This spatial property, expressed by Formula 3.4, is satisfied by User[$\mathsf{u_1}$].

$$\overbrace{\mathsf{User} \land (\mathcal{P}\ \mathsf{Job})}^{\mathsf{User.(Job)}} \land \underbrace{(\mathcal{SI}\ (\mathsf{PC}\ \mathcal{K}\ (\mathsf{lan})\ \mathsf{Printer}))}_{\textit{in a room with a } \mathsf{PC} \textit{ connected throguh } \mathsf{lan} \textit{ to a } \mathsf{Printer}} \tag{3.4}$$

Example 7. Suppose to verify if in the Smart Office Environment there are PCs which are in a Room such that it is possible to reach the Room of a Printer. This property is interesting because if no Printer is not physically reachable, there is no way to take the printed document. Such spatial property, expressed by Formula 3.5, is satisfied by PC[$\mathsf{c_0}$] and PC[$\mathsf{c_1}$].

$$\mathsf{PC} \land (\mathcal{CH}\ \underbrace{((\overbrace{\mathsf{Room} \land (\mathcal{P}\ \mathsf{PC})}^{\mathsf{Room.(PC)}})\ \mathcal{K}\ (\mathsf{Room} \lor (\mathsf{Door} \land (\mathcal{P}\ \mathsf{Unlocked})))\ (\overbrace{\mathsf{Room} \land (\mathcal{P}\ \mathsf{Printer})}^{\mathsf{Room.(Printer)}}))}_{\mathsf{Room.(PC)} \textit{ connected throguh } \mathsf{Room} \textit{ or } \mathsf{Door.(Unlocked)} \textit{ to a } \mathsf{Room.(Printer)}}) \tag{3.5}$$

Example 8. Suppose to verify if in the Smart Office Environment there are Users that are in a Room with no unlocked doors. Such users are closed inside a Room: it could be that they have the keys to open the doors or not. Such spatial property, expressed by Formula 3.6, is satisfied by User[$\mathsf{u_1}$].

$$\mathsf{User} \land (\mathcal{CH}\ \underbrace{(\mathsf{Room} \land (\mathcal{P}\ \mathsf{User}) \land (\neg \overbrace{\mathcal{PT}\ (\mathsf{Door} \land (\mathcal{P}\ \mathsf{Unlocked}))}^{\textit{points connected to } \mathsf{Door.(Unlocked)}}))}_{\mathsf{Room.(User)} \textit{ not connected to } \mathsf{Door.(Unlocked)}}) \tag{3.6}$$

In the model checking procedure described in Section 4.3, SLDB formulas are evaluated over the whole space represented by the directed bigraphical closure model. Such an evaluation yields the set of points of the directed bigraphical closure space where the formula holds. This approach is different from the one adopted in [27], where the evaluation returns a truth value indicating the existence of points in the bigraphical closure model for which the evaluated formula is true.

4 Implementation

This chapter provides a proof-of-concept implementation of the presented approach to modeling and reasoning about cyber-physical spaces. Basically, the resulting tool [6] integrates a spatio-temporal model checker and a Java library to model directed bigraphs. Section 4.1 describes Topochecker [9], the spatio-temporal model checker used as a spatial model checker to check properties over SLDB. Section 4.2 presents jLibBig [8], a Java library for Directed Bigraphs, and shows the encoding of the directed bigraph of Figure 2.2 concerning the smart office environment. Finally, Section 4.3 describes the proof-of-concept implementation of a spatial model checking procedure in jLibBig. This procedure allows verifying spatial properties defined in SLDB over Directed Bigraphs modeled using jLibBig.

4.1 Topochecker: Topological Model Checker

Topochecker [9] is a spatio-temporal model checker implemented in the functional language OCaml. It grounds on the so-called *snapshot model* [21], consisting of a temporal model (e.g., a Kripke frame) where each state is, in turn, a closure model, and spatial formulas replace atomic formulas of the temporal fragment. Snapshot models may be susceptible to state-space explosion problems as spatial formulas could be needed to be recomputed at every state. However, since the verification procedure of Section 4.3 adopts topocheker as a spatial model checker, these problems are not taken into account.

Currently, topochecker checks a spatial extension of CTL named *spatio-temporal logic of closure spaces* (STLCS), which permits arbitrary mutual nesting of its spatial and temporal fragments [10].

Definition 4.1.1 (Syntax of STLCS) *Let P be a set of propositions letters. The syntax of* STLCS *is defined by the following grammar, where p ranges over the set P:*

$$\phi ::= p \mid \top \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \mathcal{N}\ \phi \mid \phi_1\ \mathcal{S}\ \phi_2 \mid \mathcal{A}\ \psi \mid \mathcal{E}\ \psi \qquad\qquad \psi ::= \mathcal{X}\ \phi \mid \phi_1\ \mathcal{U}\ \phi_2$$

STLCS features the path quantifiers of CTL [12, 2], the *for all paths* operator $\mathcal{A}$ and the *there exists a path* operator $\mathcal{E}$. As in CTL, such operators must be followed by one of the path-specific quantifiers, such as[1] $\mathcal{X}\ \phi$ (*next*), $\mathcal{F}\ \phi$ (*eventually*), $\mathcal{G}\ \phi$ (*globally*), $\phi_1\ \mathcal{U}\ \phi_2$ (*until*), but unlike CTL, ϕ, ϕ_1, and ϕ_2 are formulas that may make use of the *closure* operator $\mathcal{N}$ or the *surround* operator $\mathcal{S}$.

[1]Some temporal operators may be derived from the ones of Definition 4.1.1.

A model $\mathcal{M}$ is composed of a Kripke structure $(S, \mathcal{T})$, where S is a non-empty set of states and $\mathcal{T}$ is a non-empty relation on states, and a closure space $(X, \mathcal{C})$, where X is a set of points and $\mathcal{C}$ the closure operator. Every state s has an associated valuation function $\mathcal{V}_s$, making $((X, \mathcal{C}), \mathcal{V}_s)$ a closure model. Equivalently, valuations have type $S \times X \to 2^P$, where P is the set of atomic propositions, thus, the valuation depends both on states and points of the space. Intuitively, there is a set of possible worlds, i.e., the states in S, and a spatial structure represented by a closure space. In each possible world, there is a different valuation of atomic propositions, inducing a different "snapshot" of the spatial situation which "evolves" over time. It is assumed that the structure $(X, \mathcal{C})$ does not change over time.

Definition 4.1.2 (Topochecker Evaluation Model) *A model $\mathcal{M} = ((S, \mathcal{T}), (X, \mathcal{C}), \mathcal{V}_{s \in S})$ is a structure where $(X, \mathcal{C})$ is a closure space, $(S, \mathcal{T})$ is a kripke frame, and $\mathcal{V}$ is a family of valuations, indexed by states. For each $s \in S$, it follows $\mathcal{V}_s : P \to \mathcal{P}(X)$.*

A path in the Kripke structure is a sequence of *spatial models* $((X, \mathcal{C}), \mathcal{V}_s)$ indexed by instants of time.

Definition 4.1.3 (Path of the Kripke Structure) *Given a Kripke frame $\mathcal{K} = (S, \mathcal{T})$, a* path *$\sigma$ is a function from $\mathbb{N}$ to S such that for all $n \in \mathbb{N}$ it follows $(\sigma(i), \sigma(i+1)) \in \mathcal{T}$. Let $\mathcal{P}_s$ denote the set of infinite paths in $\mathcal{K}$* rooted *at s, that is, the set of paths σ with $\sigma(0) = s$.*

Definition 4.1.4 (Semantics of STLCS) *Satisfaction $\mathcal{M}, x, s \models \phi$ of formula ϕ at point $x \in X$ and state $s \in S$ in model $\mathcal{M} = ((S, \mathcal{T}), (X, \mathcal{C}), \mathcal{V}_{s \in S})$ is defined by induction on terms, as follows:*

$$\begin{array}{lcl}
\mathcal{M}, x, s \models p & \iff & x \in \mathcal{V}_s(p) \\
\mathcal{M}, x, s \models \top & \iff & \mathit{true} \\
\mathcal{M}, x, s \models \neg\phi & \iff & \mathcal{M}, x, s \not\models \phi \\
\mathcal{M}, x, s \models \phi_1 \vee \phi_2 & \iff & \mathcal{M}, x, s \models \phi_1 \ \mathit{or}\ \mathcal{M}, x, s \models \phi_2 \\
\mathcal{M}, x, s \models \mathcal{N}\, \phi & \iff & x \in \mathcal{C}(\{y \in X \mid \mathcal{M}, y, s \models \phi\}) \\
\mathcal{M}, x, s \models \phi_1\, \mathcal{S}\, \phi_2 & \iff & \exists A \subseteq X \,.\, x \in A \wedge \forall y \in A \,.\, \mathcal{M}, y, s \models \phi_1 \wedge \\
 & & \quad \wedge\ \forall z \in \mathcal{B}^+(A) \,.\, \mathcal{M}, z, s \models \phi_2 \\
\mathcal{M}, x, s \models \mathcal{A}\, \psi & \iff & \forall \sigma \in \mathcal{P}_s \,.\, \mathcal{M}, x, \sigma \models \psi \\
\mathcal{M}, x, s \models \mathcal{E}\, \psi & \iff & \exists \sigma \in \mathcal{P}_s \,.\, \mathcal{M}, x, \sigma \models \psi \\
\mathcal{M}, x, \sigma \models \mathcal{X}\, \phi & \iff & \mathcal{M}, x, \sigma(1) \models \phi \\
\mathcal{M}, x, \sigma \models \phi_1\, \mathcal{U}\, \phi_2 & \iff & \exists n \,.\, \mathcal{M}, x, \sigma(n) \models \phi_2 \ \mathit{and}\ \forall n' \in [0, n) \,.\, \mathcal{M}, x, \sigma(n') \models \phi_1
\end{array}$$

The command to run the topological model checker is `topochecker EXPERIMENT_FILENAME` where `EXPERIMENT_FILENAME` is the experiment description file's name with extension *.topochecker*. Code 4.1 shows the experiment description file *test.topochecker*, generated by the model checking procedure of Section 4.3. An experiment description file consists of a model declaration, a macro declaration, and a set of commands. The first row of Code 4.1 corresponds to the model declaration, where `KRIPKE_FILENAME`, `SPACE_FILENAME`, `EVAL_FILENAME` are the names of the files containing the Kripke structure $(S, \mathcal{T})$, the closure space $(X, \mathcal{C})$, and the valuation functions $\mathcal{V}_{s \in S}$, respectively. The macro declaration is an optional list of statements of the form `Let ide = FORMULA;` or `Let ide(arg1,...,argN) = FORMULA;`. As shown in Code 4.1, this allows the definition of derived logical operators, in particular the ones of the *spatial logic for directed bigraphs* (SLDB). Finally, experiment description files contain at least one command `Check "COLOR" FORMULA;` where `COLOR` is a hexadecimal used to color the output for the specified formula.

```
Kripke "KRIPKE_FILENAME" Space "SPACE_FILENAME" Eval "EVAL_FILENAME";

// Topological Operators of SLCS:
Let interior(x) = ! (N (! x));
Let boundary(x) = (N x) & (! interior(x));
Let interiorBoundary(x) = x & (! interior(x));
Let closureBoundary(x) = (N x) & (! x);

// Operators concerning Reachability:
Let weakReachability(x, y) = ! ((! y) S (! x));
Let reachability(x, y) = x & weakReachability((x | y), y);
Let reachabilityThrough(x, y, z) = reachability(x, reachability(y, z));

// ******************************************************************************* //
//                         Spatial Logic for Directed Bigraphs (SLDB)                //
// ******************************************************************************* //

Let outer(x) = reachability([place], x);
Let inner(x) = closureBoundary(x) & [place];
Let tilde(x) = x | (closureBoundary(x) & [link]) | reachability([back], x);

// Ad-hoc Locality Operators:
Let parent(x) = reachability([node], outer(x));
Let siblings(x) = closureBoundary(outer(x)) & (! x);
Let children(x) = closureBoundary(inner(x));
Let neighbours(x) = parent(x) | siblings(x) | children(x);

// Ad-hoc Connectivity Operators:
Let points(x) = closureBoundary(closureBoundary(x) & [back]);
Let handles(x) = closureBoundary(closureBoundary(x) & [link]);
Let connectedThrough(x, y, z) = x & reachabilityThrough(tilde(x), tilde(y), tilde(z));

// Topochecker Output Filename:
Output "./dbcs/out-state";

Check "0xFF0000" FORMULA;
```

Code 4.1: *test.topochecker* Experiment File.

The Kripke frame $(S, \mathcal{T})$ and the quasi-discrete closure space $(X, \mathcal{C})$, contained in `KRIPKE_FILENAME` and `SPACE_FILENAME`, are defined as a directed graph in the dot language. The valuation functions $\mathcal{V}_{s \in S}$ are defined in the csv file `EVAL_FILENAME`. Each row of such file takes the form `s,x,prop1,prop2,...,propN`, where `s` is a node identifier of the Kripke structure, `x` is a node identifier of the closure space, and `prop1,prop2,...,propN` are the atomic propositions associated to state `s` and point `x`.

4.2 jLibBig: Java Library for Directed Bigraphs

Directed bigraphs are a modern, graphical meta-model capable of describing, at once, both the locations and the logical connections of (possibly nested) components. Bigraphs, in general, are appealing because they provide a range of generic results and tools. In particular, *jLibBig* [8] is a Java library for bigraphs and bigraphical reactive systems [23, 22]. The std package provides a standard implementation of Milner's original variants. The classes Bigraph and BigraphBuilder offer methods for assembling bigraphs programmatically, but there is also the possibility to import them from some textual representations. The ldb package implements the directed variants of bigraphs and bigraphical reactive systems [18, 19].

```
/** Polarized Signature with Controls of the Smart Office Environment **/
DirectedSignatureBuilder polarizedSignatureBuilder = new DirectedSignatureBuilder();

polarizedSignatureBuilder.add("Room", true, 2, 0);
polarizedSignatureBuilder.add("Door", true, 0, 2);
polarizedSignatureBuilder.add("Locked", true, 0, 0);
polarizedSignatureBuilder.add("Unlocked", true, 0, 0);
polarizedSignatureBuilder.add("PC", true, 1, 0);
polarizedSignatureBuilder.add("Printer", true, 2, 1);
polarizedSignatureBuilder.add("Tray", true, 1, 0);
polarizedSignatureBuilder.add("Spool", true, 0, 1);
polarizedSignatureBuilder.add("User", true, 1, 1);
polarizedSignatureBuilder.add("Job", true, 1, 0);

DirectedSignature polarizedSignature = polarizedSignatureBuilder.makeSignature();
```

Code 4.2: Econding of the *polarized signature* of the directed bigraph of Figure 2.2.

The remaining part of this section describes the encoding using jLibBig of the directed bigraph of Figure 2.2 concerning the smart office environment. Code 4.2 shows the definition of the *polarized signature*, where the controls with their respective polarized arities are defined. As an example, the control Printer has two outward and one inward port. Given the signature, it is possible to assemble the directed bigraph. Firstly, Code 4.3 shows the *outer interface*'s encoding, which is composed of five upward names: door0, door1, lan, secretary, and director. Then, Code 4.4 shows the encoding of all the nesting and linking relations of the *place* and *link graphs* of Figures 2.3 and 2.4.

```
/** Outer Interface of the Directed Bigraph **/
DirectedBigraphBuilder directedBigraphBuilder =
  new DirectedBigraphBuilder(polarizedSignature);

OuterName door0 = directedBigraphBuilder.addAscNameOuterInterface(0, "door0");
OuterName door1 = directedBigraphBuilder.addAscNameOuterInterface(0, "door1");
OuterName lan = directedBigraphBuilder.addAscNameOuterInterface(0, "lan");
OuterName secretary = directedBigraphBuilder.addAscNameOuterInterface(0, "secretary");
OuterName director = directedBigraphBuilder.addAscNameOuterInterface(0, "director");
```

Code 4.3: Econding of the *outer interface* of the directed bigraph of Figure 2.2.

```
/** Nesting and Linking relations of the Directed Bigraph **/
Root root = directedBigraphBuilder.addRoot(0);

Node d0 = directedBigraphBuilder.addNode("Door", root);
directedBigraphBuilder.addNode("Unlocked", d0);

Node d1 = directedBigraphBuilder.addNode("Door", root);
directedBigraphBuilder.addNode("Locked", d1);

Node r0 = directedBigraphBuilder.addNode("Room", root);
directedBigraphBuilder.relink(door0, r0.getOutPort(0));
directedBigraphBuilder.relink(d0.getInPort(0), r0.getOutPort(1));
directedBigraphBuilder.addSite(r0);

Node c0 = directedBigraphBuilder.addNode("PC", r0);
Node s0 = directedBigraphBuilder.addNode("Spool", r0);
Node p0 = directedBigraphBuilder.addNode("Printer", r0);
Node t0 = directedBigraphBuilder.addNode("Tray", r0);
directedBigraphBuilder.relink(s0.getInPort(0), p0.getOutPort(0));
directedBigraphBuilder.relink(p0.getInPort(0), t0.getOutPort(0));
directedBigraphBuilder.addSite(t0);
directedBigraphBuilder.addSite(s0);

Node r1 = directedBigraphBuilder.addNode("Room", root);
directedBigraphBuilder.relink(d0.getInPort(1), r1.getOutPort(0));
directedBigraphBuilder.relink(d1.getInPort(0), r1.getOutPort(1));
directedBigraphBuilder.addSite(r1);

Node c1 = directedBigraphBuilder.addNode("PC", r1);
Node u0 = directedBigraphBuilder.addNode("User", r1);
Node j0 = directedBigraphBuilder.addNode("Job", u0);
directedBigraphBuilder.relink(c1.getOutPort(0));
directedBigraphBuilder.relink(secretary, u0.getOutPort(0));
directedBigraphBuilder.relink(u0.getInPort(0), j0.getOutPort(0));

Node r2 = directedBigraphBuilder.addNode("Room", root);
directedBigraphBuilder.relink(d1.getInPort(1), r2.getOutPort(0));
directedBigraphBuilder.relink(door1, r2.getOutPort(1));
directedBigraphBuilder.addSite(r2);

Node c2 = directedBigraphBuilder.addNode("PC", r2);
Node u1 = directedBigraphBuilder.addNode("User", r2);
Node j1 = directedBigraphBuilder.addNode("Job", u1);
directedBigraphBuilder.relink(lan, c0.getOutPort(0), p0.getOutPort(1), c2.getOutPort(0));
directedBigraphBuilder.relink(director, u1.getOutPort(0));
directedBigraphBuilder.relink(u1.getInPort(0), j1.getOutPort(0));

DirectedBigraph directedBigraph = directedBigraphBuilder.makeBigraph();
```

Code 4.4: Econding of the *nesting* and *linking relations* of the directed bigraph of Figure 2.2.

As shown, jLibBig provides an easy way to model directed bigraphs. This work aims to extend jLibBig with a model checking procedure that allows spatial verification over directed bigraphs. The idea, shown in Code 4.5, is to implement a method `check()` which automatically verifies a spatial property (specified in the *spatial logic for directed bigraphs*) over the modeled directed bigraph.

```
/**
 * Properties of Space over Directed Bigraphs (SLDB):
 * "[Room] & parent([Printer])"
 * "[PC] & points([lan])"
 * "[User] & points([director])"
 * "[PC] & (! points([lan])) & siblings([User] & parent([Job]))"
 * ...
 */
ArrayList<Owned> res = directedBigraph.check(
  "[User] & parent([Job]) & siblings(connectedThrough([PC], [lan], [Printer]))");
```

Code 4.5: Spatial Model Checking over the directed bigraph of Figure 2.2.

4.3 Spatial Model Checking in jLibBig

Chapter 3 described how to reason about spatial properties of directed bigraphical models of space, in particular, the automated derivation of a *directed bigraphical closure model* and the definition of a *spatial logic for directed bigraphs* (SLDB), which is a specific form of SLCS serving as a spatial logic for directed bigraphical closure models. Since Section 4.2 shows an easy way to model directed bigraphs with jLibBig, the next step is to provide a spatial model checking procedure that allows verification of spatial properties over the modeled directed bigraphs. Thus, this section describes the method `check()` shown in Code 4.6, which evaluates SLDB formulas over the space represented by a directed bigraph.

```
/**
 * @param formula: a spatial property expressed in SLDB
 * @return the list of entities that satisfies the formula
 */
public ArrayList<Owned> check(String formula) { ... }
```

Code 4.6: Spatial Model Checking method for directed bigraphs in jLibBig.

To understand how the method `check()` works, it is necessary to analyze its implementation [6]. However, for space reasons, this section describes an equivalent pseudo-code that is more straightforward and more readable than the source code. Basically, the method consists of three phases: the derivation of the directed bigraphical closure model, the execution of the resulting experiment using topochecker, and the transformation of the output in a suitable format for jLibBig. Note that, as described in Section 4.1, Topochecker is a spatio-temporal model checker that works with a specific evaluation model. Fortunately, such an evaluation model can be obtained by combining the directed bigraphical closure model with a fictitious Kripke structure. The Kripke structure and the directed bigraphical closure space are defined as directed graphs, using *jDot*, a Java library to generate GraphViz dot files [3].

```
/** Kripke Structure **/
jdot.Graph kripke = new jdot.Graph("kripke");
kripke.addNode(new jdot.Node("0"));
try {
  FileWriter kripkeFile = new FileWriter(WORKING_DIRECTORY + KRIPKE_FILENAME);
  kripkeFile.write(kripke.toDot());
} catch (IOException e) { ... }
```

Code 4.7: Construction of the *Kripke structure* in method check().

Code 4.7 shows the construction of the fictitious Kripke structure, which consists of a graph with only one state. Such structure is necessary with topochecker but is meaningless in this work focused on spaces instead of systems. The main part of the method check() is shown in Codes 4.8 and 4.9, where the directed bigraphical closure model is derived. The idea is to construct the directed bigraphical closure space and the associated valuation function simultaneously. Firstly, starting from the *locality* point of view, nodes and sites are represented in the directed graph by adding nodes for which the valuation function assigns the corresponding set of propositions. Then, nesting relations are represented in the directed graph by adding *place* nodes and edges, according to the derivation described in Section 3.2.1.

```
/** Space Structure + Evaluation Function **/
jdot.Graph space = new jdot.Graph("space");
StringBuilder eval = new StringBuilder();

/** Nodes, Sites and Nesting Relations **/
int id = 0;
Queue<Entity> q = new Queue<Entity>();
q.add(directedBigraph.roots());
while (!q.isEmpty()) {
  Entity p = q.poll();
  List<Entity> cs = p.children();
  if (!cs.isEmpty()) {
    space.addNode(place(p),id); eval.append("0," + id + ",place"); id++;
    if (p.isNode()) {
      space.addEdge(p,place(p));
    }
  }
  for (Entity c : cs) {
    if (c.isSite()) {
      space.addNode(c,id);  eval.append("0," + id + ",site"); id++;
    } else {
      q.add(c);
      space.addNode(c,id);  eval.append("0," + id + ",node," + c.control) id++;
    }
    space.addEdge(place(p),c);
  }
}
```

Code 4.8: Construction of the *closure space* and the *valuation function* in method check(). 1/2

```
/** Edges, Names and Linking Relations **/
for (Entity e : directedBigraph.edges()) {
  space.addNode(e,id); eval.append("0," + id + ",edge"); id++;
}
for (Entity i : directedBigraph.inners()) { ... }
for (Entity o : directedBigraph.outers()) {
  space.addNode(o,id); eval.append("0," + id + ",outer," + o.value); id++;
}
for (Entity i : directedBigraph.inners()) { ... }
for (Entity o : directedBigraph.outers()) {
  if (o.isDownward()) {
    Entity h = o.handle();
    space.addNode(link(o),id); eval.append("0," + id + ",link"); id++;
    space.addNode(back(o),id); eval.append("0," + id + ",back"); id++;
    space.addEdge(o,link(o));
    space.addEdge(back(o),o);
    space.addEdge(link(o),h);
    space.addEdge(h,back(o));
  }
}
for (Entity n : directedBigraph.nodes()) {
  List<Entity> ps = n.outPorts();
  if (!ps.isEmpty()) {
    space.addNode(link(n),id); eval.append("0," + id + ",link"); id++;
    space.addNode(back(n),id); eval.append("0," + id + ",back"); id++;
    space.addEdge(n,link(n));
    space.addEdge(back(n),n);
  }
  for (Entity p : ps) {
    Entity h = p.handle();
    space.addEdge(link(n),h);
    space.addEdge(h,back(n));
  }
}
try {
  FileWriter spaceFile = new FileWriter(WORKING_DIRECTORY + SPACE_FILENAME);
  spaceFile.write(space.toDot());
  FileWriter evalFile = new FileWriter(WORKING_DIRECTORY + EVAL_FILENAME);
  evalFile.write(eval.toString());
} catch (IOException e) { ... }
```

Code 4.9: Construction of the *closure space* and the *valuation function* in method check(). 2/2

To complete the construction of the directed bigraphical closure model, it is necessary to consider also the *connectivity* component. Thus, edges and names are represented in the directed graph by adding nodes for which the valuation function assigns the corresponding set of atomic propositions. Furthermore, linking relations are represented in the directed graph by adding *link*, *back* nodes, and edges, again according to the derivation described in Section 3.2.1. At this point, the method `check()` has completed the automated construction of the evaluation model, which basically corresponds to the directed bigraphical closure model. As shown in Code 4.10, the next phase is to define the *experiment description file* and run the tool topochecker. Code 4.1 shows the resulting file *test.topochecker*, which specifies the files' names containing the evaluation model and the SLDB formula to check.

```
/** Test Definition **/
StringBuilder test = new StringBuilder();
try {
  test.append(
    "Kripke" + KRIPKE_FILENAME +  "Space" + SPACE_FILENAME + "Eval" + EVAL_FILENAME + ";");
  test.append((String) Files.readAllBytes(Paths.get(WORKING_DIRECTORY + LOGIC_FILENAME)));
  test.append("Check 0xFF0000 " + formula + ";");
} catch (IOException e) { ... }
try {
  FileWriter testFile = new FileWriter(WORKING_DIRECTORY + TEST_FILENAME);
  testFile.write(test.toString());
} catch (IOException e) { ... }

/** Test Execution **/
try {
  Runtime.getRuntime().exec(TOPOCHECKER + " " + WORKING_DIRECTORY + TEST_FILENAME);
} catch (IOException e) { ... }
```

Code 4.10: Construction and execution of the *experiment file* in method check().

Once the verification terminates, the method `check()` returns the directed bigraph's entities that satisfy the formula (Code 4.11). To do so, it simply returns the entities for which the correspondent points in the experiment output are colored with the formula color indicated in the experiment file.

Overall, the method strongly depends on the SLDB evaluation, which in this work is accomplished by using the spatio-temporal tool topochecker as a spatial model checker over directed bigraphical closure models. The complexity of topochecker is *linear* in the size of the closure space and the formula [11].

```
/** Output Analysis **/
ArrayList<Owned> output = new ArrayList<Owned>();
try {
  File myObj = new File(WORKING_DIRECTORY + OUTPUT_FILENAME);
  Scanner myReader = new Scanner(myObj);
  while (id > 0) {
    String row = myReader.nextLine();
    String [] rowArray = row.split(" ");
    if (rowArray[1].equals("[fillcolor=\"#FF0000\",style=\"filled\"];")) {
      output.add(directedBigraph.getEntity(id));
    }
    id--;
  }
} catch (FileNotFoundException e) { ... }
return output;
```

Code 4.11: Transformation of Topochecker's output in method check().

5 Conclusions

This work proposed a methodology to deal with the challenges arising from cyber-physical spaces. In particular, it describes an approach focused on *modeling* and *reasoning* about space. Cyber-physical spaces, intended as hybrid environments of interrelated computational and physical entities, have been modeled using Directed Bigraphs [18], a bigraphical meta-model that subsumes and generalizes both original Milner's [23] and Sassone-Sobocinski's [25] variants of bigraphs. Such graph-based formalism is capable of describing, at once, the locations and the connections of components.

Concerning reasoning, directed bigraphical models of space have been interpreted as so-called quasi-discrete closure spaces. This choice allowed adopting a spatial logic for closure spaces along with the functionality of a corresponding model checker [9]. The main contribution of this work has been the definition of a particular form of spatial logic for closure spaces called *spatial logic for directed bigraphs* (SLDB), with a corresponding evaluation model. Thanks to this, complex spatial properties, with both locally bounded and globally scoped predicates, of directed bigraphical models can be specified.

Furthermore, as a proof-of-concept, the library jLibBig [8] has been extended with a spatial model-checking procedure, adding the verification functionality to a library that can be used for modeling cyber-physical spaces [6]. The utility of the resulting procedure strongly depends on the efficiency of the underlying model checker. Currently, the *complexity* of the model-checking algorithm is linear in the product of the formula's size and the number of points and edges of the closure space. It follows that the construction of the directed bigraphical closure space plays a fundamental role in the overall efficiency. However, the structure, and thus the size, of such closure space is motivated by the expressivity power of the spatial logic for directed bigraphs. Concerning this, Section 5.1 proposes alternative constructions to reduce the closure space's size, but for which fewer spatial properties can be specified.

Although this work has been motivated by the challenges of cyber-physical spaces, the resulting approach can be applied to every other kind of directed bigraphical model. The notion of *discrete space* 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 classic examples that inherently build on the abstract notion of discrete space. Directed bigraphs are a generalization very suited for reasoning about dependencies and request flows between components, such as those found in client-server or producer-consumer scenarios. Thus, the proposed approach can be applied whenever directed bigraphs represent a suitable modeling formalism.

5.1 Future Work

As future work, there are several potential evolutions of the proposed approach. It may be interesting to review the reasoning approach as a composition of two kinds of reasoning: one concerning *locality*, another concerning *connectivity*. The idea comes from BiLog [13], a spatial logic for Milner's variant of bigraphs, which has been proposed as a composition of two logics: a *place graph* logic and a *link graph* logic. The two kinds of partial reasoning, about locality rather than connectivity, may be useful because they involve different space structures of less size, decreasing the computational cost.

Similarly, another possibility is to define a reasoning approach that considers only locally bounded properties of space. The verification of globally bounded spatial properties requires representing both the linking relations and their inverse, increasing the size of the considered space. All previous reasoning variants are less expensive and may still be sufficient in a context different from cyber-physical spaces. It could be exciting to investigate the application of the proposed approach, or one of its variants, in another scenario where directed bigraphs are still a suitable modeling formalism.

Finally, another significant evolution of the presented work may be to include the temporal flavor, moving from cyber-physical spaces to *cyber-physical systems*. The dynamics of systems can be expressed as *rewriting rules*, yielding to a Directed Bigraphical Reactive System [19]. The integration of temporal reasoning, resulting in a spatio-temporal approach, can be achieved in more than one way.

Bibliography

[1] Giorgio Bacci, Davide Grohmann, and Marino Miculan. Dbtk: a toolkit for directed bigraphs. In *International Conference on Algebra and Coalgebra in Computer Science*, pages 413–422. Springer, 2009.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT press, 2008.

[3] Gerald Boersma. jdot: A java library to generate graphviz dot files. `https://github.com/gboersma/jdot`, March 2020.

[4] Fabio Burco, Marino Miculan, and Marco Peressotti. Towards a formal model for composable container systems. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, pages 173–175, 2020.

[5] Eduard Čech, Zdeněk Frolík, and Miroslav Katětov. Topological spaces. 1966.

[6] Enrico Cesca and Marino Miculan. Model checking in space over directed bigraphs with jlibbig. `https://github.com/enricocesca/bigraphical-verification`, February 2021.

[7] Alessio Chiapperini, Marino Miculan, and Marco Peressotti. Computing embeddings of directed bigraphs. In *International Conference on Graph Transformation*, pages 38–56. Springer, 2020.

[8] Alessio Chiapperini, Marino Miculan, and Marco Peressotti. jlibbig: A java library for bigraphs and bigraphical reactive systems. `https://bigraphs.github.io/jlibbig/`, May 2020.

[9] Vincenzo Ciancia and Gina Belmonte. Topochecker: A topological model checker. `https://github.com/vincenzoml/topochecker`, November 2019.

[10] Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. An experimental spatio-temporal model checker. In *SEFM 2015 Collocated Workshops*, pages 297–311. Springer, 2015.

[11] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Specifying and verifying properties of space. In Josep Diaz, Ivan Lanese, and Davide Sangiorgi, editors, *Theoretical Computer Science*, pages 222–235, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[12] Edmund M Clarke Jr, Orna Grumberg, Daniel Kroening, Doron Peled, and Helmut Veith. *Model checking.* MIT press, 2018.

[13] Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Spatial logics for bigraphs. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors,

Automata, Languages and Programming, pages 766–778, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

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

[15] Alexander John Faithfull, Gian Perrone, and Thomas T Hildebrandt. Big red: A development environment for bigraphs. *Electronic Communications of the EASST*, 61, 2013.

[16] Antony Galton. A generalized topological view of motion in discrete space. *Theoretical Computer Science*, 305(1-3):111–134, 2003.

[17] Davide Grohmann. Security, cryptography and directed bigraphs. In *International Conference on Graph Transformation*, pages 487–489. Springer, 2008.

[18] Davide Grohmann and Marino Miculan. Directed bigraphs. *Electronic Notes in Theoretical Computer Science*, 173:121–137, 2007.

[19] Davide Grohmann and Marino Miculan. Reactive systems over directed bigraphs. In *International Conference on Concurrency Theory*, pages 380–394. Springer, 2007.

[20] Davide Grohmann and Marino Miculan. Controlling resource access in directed bigraphs. *Electronic Communications of the EASST*, 10, 2008.

[21] Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. *Spatial Logic + Temporal Logic = ?*, pages 497–564. Springer Netherlands, Dordrecht, 2007.

[22] Robin Milner. Bigraphical reactive systems. In Kim G. Larsen and Mogens Nielsen, editors, *CONCUR 2001 — Concurrency Theory*, pages 16–35, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

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

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

[25] Vladimiro Sassone and Pawel Sobocinski. Reactive systems over cospans. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 311–320. IEEE, 2005.

[26] Zhucheng Shao and Jing Liu. Spatio-temporal hybrid automata for cyber-physical systems. In *International Colloquium on Theoretical Aspects of Computing*, pages 337–354. Springer, 2013.

[27] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. Modeling and verification of evolving cyber-physical spaces. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 38–48, New York, NY, USA, 2017. Association for Computing Machinery.

[28] Johan van Benthem and Guram Bezhanishvili. *Modal Logics of Space*, pages 217–298. Springer Netherlands, Dordrecht, 2007.