CYBER-PHYSICAL CLOUD COMPUTING LAB
UNIVERSITY OF CALIFORNIA, BERKELEY

BIAGENTS – A BIGRAPHICAL AGENT MODEL FOR STRUCTURE-AWARE COMPUTATION

Eloi Pereira, Christoph Kirsch and Raja Sengupta

Working Papers
CPCC-WP-2012-08-01

CPCC Berkeley

August 2012

BiAgents - A Bigraphical Agent Model for Structure-Aware Computation

Eloi Pereira
Systems Engineering
Department of Civil and Environmental Engineering
UC Berkeley, USA
eloi@berkeley.edu

Christoph Kirsch
Department of Computer Sciences
University of Salzburg, Austria
christoph.kirsch@cs.uni-salzburg.at

Raja Sengupta
Systems Engineering
Department of Civil and Environmental Engineering
UC Berkeley, USA
raja@ce.berkeley.edu

ABSTRACT

This paper describes a modelling formalism, called bigraphical agents (BiAgents), for cyber-physical systems where computation are embedded into the physical world. Just as embedded computing models flow in time, BiAgents flow in both space and time.

On cyber-physical systems computation (i.e. software programs) are tightly coupled with the physical platform and environment. We seek for providing a model with a clear separation between what is virtual (software) and physical (execution machines and their environment). We choose to model the physical layer using bigraphs. The virtual layer is modelled using agents specified abstractly as an algebraic structure. Agents operate over a physical structure and can observe, control and migrate.

By making the virtual layer abstract we seek at leveraging the use of widely accepted programming languages and models of concurrency for specifying the agents. Thus, our model is the combination of two models: one for modelling the physical world and another to model the virtual layer. This approach contrasts with others found in the literature that seek at finding a single model for the overall system layers.

In the BiAgent model, multiple agents are composed with a bigraph. The agents are algebraic structures that model the virtual and bigraphs model the physical. The bigraph is our model of space. The flow of the physical world in time corresponds to a sequence of bigraphs. In the BiAgent model, computation can flow in space without flowing in time, or flow in time without flowing in space, or flow in both.

BiAgents are able to observe the physical structure, and map observations to control and migration actions. A BiAgent can exhibit two kinds of mobility: "physical" , when it moves along with its host, and "virtual", when it migrates to another host.

The purpose is a modelling formalism enabling the modeller to clearly separate the virtual and physical components of a model and limit the interaction of the two with a semantics.

Since BiAgents operate over a shared environment some ill-defined behaviours may emerge due to concurrency. We propose strategies for preventing these behaviours based on avoiding interleaving between observation and control and based on partitioning the physical structure where each agent operates in its own region.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

Categories and Subject Descriptors

F.1.0 [**Computation by abstract devices**]: General; F.1.1 [**Computation by abstract devices**]: Models of Computation; F.1.2 [**Computation by abstract devices**]: Modes of Computation—*Interactive and reactive computation, parallelism and concurrency*; I.2.11 [**Artificial intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Theory

Keywords

Models of Computation

1. INTRODUCTION

We propose a model for computational systems embedded in a physical world or seeking to control such a world. More particularly, our interest and the model is focused on physical worlds that change structure, such as worlds with moving people or vehicles, nomadic programs, or changing connectivity between entities as caused by vehicles moving in and out of wireless range, or people going on and off-line in applications such as Skype.

The model is named the BiAgent model, short for Bigraphical Agents. In this model, computational or virtual entities are agents and the physical entities or world is a Bigraph [18]. Entities such as vehicles, persons, computers or smartphones are nodes related by the bigraph placing or linking graphs. Agents model the disembodied or virtual entities such as computations or minds. They interact with the bigraph by *observing* it, being *hosted* in it, *migrating* in it, and *controlling* it. Control is simply Bigraph Reaction Rules (BRR) as formalized by Milner [18, 17]. The BiAgent model adds definitions for observing, hosting, and migrating.

Observing is like querying by Birkedal [7] but defined more abstractly. Migration and hosting are more particular to the BiAgent model. An agent in the BiAgent model is simply a sequential computation, i.e., agent time is linear. At each time advance the agent requests an observation, migration, or control action. With an appropriate Nerode equivalence [14], the agent in the BiAgent model would be a finite or infinite state machine. In this paper agent state is abstracted for clarity. Agent memory could be finite or denumerable.

Our manner of coupling bigraph and agents to constitute the BiAgent model reproduces the system representation in control theory. Control theory separates plant and control as illustrated on the left in figure 1. The analogs in the BiAgent

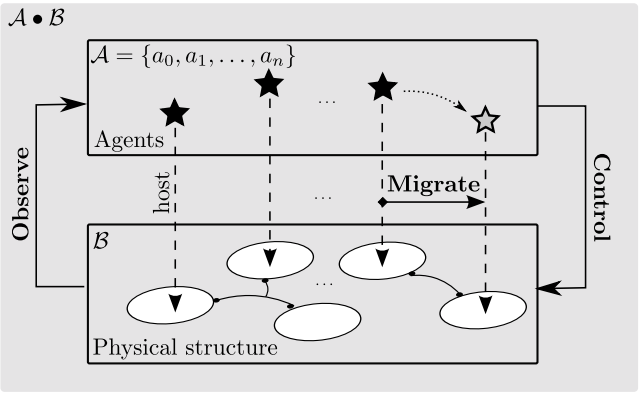


Figure 1: Block diagram depicting the control loop between agents and the physical structure.

model are bigraph and agents as on the right. Sensors on the left are analogous to bigraph queries on the right and actuators to bigraph reaction rules. The combination of agents and bigraph modeling computational systems embedded in a physical world evolves as follows. Agents schedule control actions in agent time that are bigraph reaction rules. When these rules take effect in bigraph time, the physical world flows from one bigraph to another, i.e., structure changes. This flow of bigraphs is a behavior of a Bigraph Reaction System (BRS), so named following [18]. When the flow in BRS time reflects in agent time, the bigraph transition may be observed by the agents and new migration or control actions scheduled by them to repeat the process.

Each agent or virtual entity has a host in the physical world. The host is a bigraph node. The host of an agent endows the virtual with a point of origin in the physical world. The observation and control actions scheduled by an agent are concretized with respect to this origin permitting all such actions to be inherently local. An agent can migrate from one host to another ? called a migration action. Since the host of an agent is its point of reference in the physical world, the results of its observation or control actions change as it migrates or moves in the physical world. In other words an agent can obtain different results from the same sensor or actuator, respectively bigraph query or reaction rule, as it moves in its bigraph, as would be natural when sensing or actuating with mobile systems like robots. This locality of sensing and actuation in control theory is analogous to context-awareness in computing [1].

The BiAgent model has a hybrid model of time like the Sifakis-Henzinger hybrid system model [13] or the Alur-Dill timed automaton model [3]. In these models an automaton advances as discrete time and clocks or differential equations as continuous time. The coupling of automaton and differential equation advances by inter-leaving discrete and continuous time advances. In the BiAgent model the execution of a bigraph reaction rule (control action) is a transition in the bigraph reaction system and is an advance of bigraph time. We call bigrah time *physical time* because a change in the bigraph models a change in a physical world. The execution of a migration does not change the bigraph, but only the relation between an agent and the bigraph, i.e., the hosting relation. We think of this as modeling a program transferring from one computer to another over a network. Migration would happen at network time-scales. So we call an advance in the hosting relation between agents and bigraph an advance in *network time*. Finally, each agent is a linear time computation advancing by scheduling observation, control, or migration actions. We call the advance of agent time *computation-time*.

The BiAgent model assumes agents, bigraph, and the hosting relation are coupled asynchronously, meaning like hybrid systems, a BiAgent model advances by inter-leaving computation time, network time, and physical, time. In the parlance of cyber-physical systems, both computation time and network time are cyber-time. In a trace of a BiAgent model advances in cyber-time are clearly separated from advances in physical time, and advances in cyber-time are further separated into advances in computation time or network time. The concurrency model for multiple agents is asynchronous, meaning multiple agents advance by interleaving time advances of single agents. Thus a particular advance of cyber-time is an advance of one and only one agent. The basic BiAgent model has no synchronous advances of agents. The interpretation of synchronous programs [6] in an asynchronous but concurrent environment [5, 21, 24] came after the development of the synchronous model itself. We have started our coupling of the cyber and physical in an asynchronous semantics and intend to borrow synchronization models from embedded computing after. The theorems in this paper are included to partly clarify how this might be done.

Since the hybrid model of time can advance physical time without advancing cyber-time and vice-versa, it produces a hybrid model of mobility. An agent may advance in the physical world because its hosting bigraph node moves from one place to another, i.e., an advance in the bigraph placing graph. This kind of mobility entails an advance in physical time without one in cyber-time and is called physical mobility. But an agent can also move by migration, which entails an advance in network time but no advance in physical time. This is called cyber-mobility. Advances in computation time cannot produce mobility in the BiAgent model.

The bigraph as a model of this physical world has been of interest [7, 22] and extending it to model computational systems embedded in the physical world has been of interest as well. Debois [8] models the physical world and computation as two different BRSs that are composed together. The approach in [8] has the obvious advantage of modeling the cyber and physical in the same formalism, while the BiAgent separates modeling into two different ones ? bigraphs for the physical and agents for the cyber. However, we hope the BiAgent approach will obviate the need to develop new models for the cyber-physical by enabling re-use of models

from embedded computing for the cyber, while using bigraphs for the physical. For this reason, concurrent agents are abstractly defined here as sequential machines composed asynchronously. The atomic building block of many models is a sequential computation, e.g., the actor in a concurrent actor model [2] or a module in Lustre [11].

Thus we speculate that adding reliable inter-agent messaging should specialize BiAgents to BiActors meaning the agent world could be programmed in actor languages such as Erlang, Scala, and Cloud Haskell [4, 12, 10]. Adding a higher-level synchronous semantics to just the agent time, might specialize BiAgents to a BiLustre or BiGiotto, meaning the agent world would be programmed in these languages. Moreover models such as the pi-calculus [16] or SHIFT [9] have already been developed for the computational part of systems with changing structure, and we hope this BiAgent approach will permit their re-use via a hybrid formalism coupling them with bigraphs.

Our path to veracity of these speculations is to formalize the BiAgent model as done in this paper and work with others to implement biActors, biSimulink, biLustre, or bi-π. The elegance or ease experienced in this future implementation program will be the measure of the value of this BiAgent approach to the programming of computational systems controlling a physical world with changing structure.

We feel most large spatially distributed systems change structure, meaning engineers have built many computational systems to control systems with changing structure. For example, the Advanced Traffic Control Systems in the city of Los Angeles manages a few thousand traffic lights [23]. We expect the exact number under management changes from day to day, meaning the system is endowed with an architecture able to keep certain operating properties invariant as traffic lights drop in or out of a continuously operating system. Aircraft change location in the National Airspace System (NAS) as they fly and handoff from one controller to another, i.e., changing connectivity between controller and aircraft. The control of systems with changing structure is common.

1.1 Computations that flow in space

Let a denote an agent (virtual) operating in a physical world $\mathcal{B}$, denoted as $a \bullet \mathcal{B}$.This composition produces the following trace:

$$(B_0, h_0^a) \prec (B_1, h_1^a) \prec (B_2, h_2^a) \prec \dots$$

Here $(B_0, B_1, B_2, \dots)$ represents the evolution of the physical world in time and $(h_0^a, h_1^a, h_2^a, \dots)$ the locations of the agent in it. We formalize the B_i as bigraphs. Thus an evolution of the physical world is a sequence of bigraphs. We call this a physical trace or a bigraph trace. We formalize the location of an agent as a node in a bigraph and call the sequence $(h_0^a, h_1^a, h_2^a, \dots)$ a node trace or virtual trace, since the agent is virtual. We use h for location because we think of the bigraph node on which the agent is located as its *host*, i.e, the node hosts the agent.

The sequence of pairs is called a *BiAgent trace* or a bigrah-node trace. Each element (B, h) of the trace is simply a *BiAgent instance* or a bigraph-node pair. $\mathcal{B}$ will be formalized as an algebraic structure over bigraphs and is therefore called a bigraph structure or a *physical structure*.

Physical traces model changes in the physical structure (e.g. a car changes its position, a phone connects to a local cell, etc.) while virtual traces model the migration of agents from host to host (e.g. a music player application saves the current music being played on one device and starts playing the same music on another device). We start with a simple example which is used throughout the remainder of the paper.

Example 1. Figure 2 models a physical structure of cabs and pedestrians moving in a city from street to street. A pedestrian can hop-on to a cab, move around, and hop-off. Cabs can only drive inside roads and pedestrians are too lazy to walk. Figure 2 presents an instance of this physical structure. The figure depicts four roads ($road_i$, $i \in \{0, 1, 2, 3\}$)

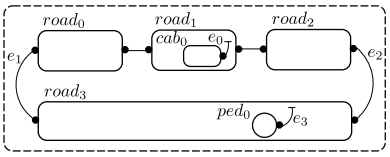


Figure 2: Cab and pedestrian example.

a cab (cab_0) and a pedestrian (ped_0). Links between roads represents physical adjacency while links between cabs and pedestrians represent communication between both entities (currently disconnected). Consider two agents, *cabAg* and *pedAg* hosted respectively into cab_0 and ped_0. Agent *cabAg* requires the cab to move non-deterministically from road to road. Let us call this action $MOVE$. Agent *pedAg* observes the physical structure looking for a cab. If it finds one it requires ped_0 to hop-on to the cab. When *pedAg* reaches its destination it requires its host to hop-off from the cab. These two actions are denoted as $HOPON$ and $HOPOFF$. Figure 3 presents a BiAgent trace for this example. *cabAg* is depicted as a black filled star while *pedAg* is depicted as a white filled star. If we remove the stars from the four models in Figure 3 they become bigraphs. With the stars the four graphs are BiAgent models of the sort $\{pedAg, cabAg\} \bullet \mathcal{B}$.

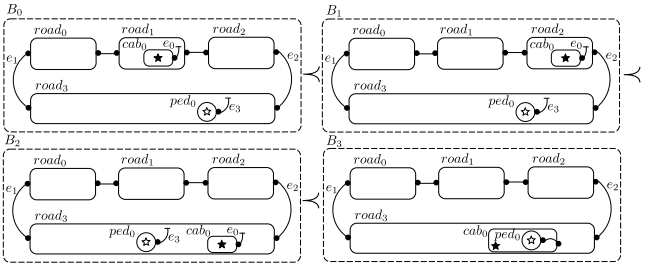


Figure 3: Example of a BiAgent trace.

We think of *pedAg* as the mind of the pedestrian and ped_0 as her body. Likewise *cabAg* models the mind of the cab driver and cab_0 the physical cab or, to be more concrete, the cab and the driver. If the cab were an automated vehicle then *cabAg* would model computation (disembodied) and cab_0 the computer, sensor, or vehicle embodiment of the computation. The purpose is a modelling formalism enabling the modeller to clearly separate the virtual and physical components of a model and limit the interaction of the two with a semantics.

The trace of Figure 3 is formally represented as:

$$t = (B_0, H_0) \overset{MOVE}{\prec} (B_1, H_0) \overset{MOVE}{\prec} (B_2, H_0) \overset{HOPON}{\prec} (B_3, H_0)$$

H_0 is the host distribution function with $H_0(cabAg) = cab_0$ and $H_0(pedAg) = ped_0$. We stack actions on top of $\prec$ to improve readability. Note that the host distribution does not change with time in this example even though the cab is moving.

The flow of the agents in space can be made explicit by writing a projection $t^a = (p_0^a, h_0^a) \prec (p_1^a, h_1^a) \prec \ldots$ of t where each p_i^a is a projection of B_i retaining only the bigraph node containing the host h_i^a of agent a at time i. The projections of the trace t for $cabAg$ and $pedAg$ are, respectively:

$$t^{cabAg} = (road_1, cab_0) \overset{MOVE}{\prec} (road_2, cab_0) \overset{MOVE}{\prec} (road_3, cab_0) \overset{HOPON}{\prec} (road_3, cab_0) \quad (1)$$

$$t^{pedAg} = (road_3, ped_0) \overset{MOVE}{\prec} (road_3, ped_0) \overset{MOVE}{\prec} (road_3, ped_0) \overset{HOPON}{\prec} (cab_0, ped_0) \quad (2)$$

Consider another agent $countAg$ with the role of counting the number of pedestrians passing by $road_2$ inside a cab. Whenever a cab enters $road_2$, $countAg$ migrates from $road_2$ to the cab, counts the number of pedestrians inside the cab and returns to $road_2$. A valid projection of a trace for agent $contAg$ might be:

$$t^{countAg} = (0, road_2) \overset{MOVE}{\prec} (0, road_2) \overset{mgrt}{\prec} (road_2, cab_0) \overset{MOVE}{\prec} (0, road_2) \overset{mgrt}{\prec} (0, road_2) \overset{HOPON}{\prec} (0, road_2) \quad (3)$$

where 0 denotes the "container"[1] of all the roads and $mgrt$ denotes the migration action of an agent.

The remainder of this paper is concerned with making this vision formal. We start by introducing in Section 2 a model for defining physical structures based on Milner's bigraph model. We proceed in Section 3 by defining bigraphical agents operating concurrently over the same physical structure. Section 4 addresses concurrency issues emerging from the sharing of physical structures among agents. We finish in Section 5 with some conclusions and future work.

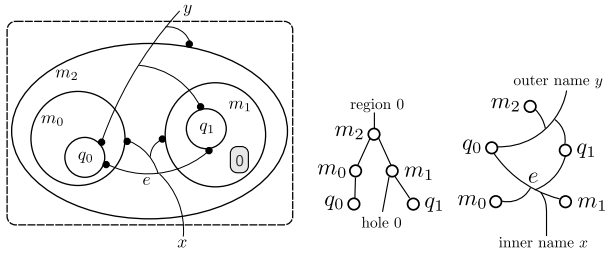
2. MODEL FOR PHYSICAL STRUCTURES

Our choice for modelling statical physical structures of the world relies on Robin Milner's bigraph model. In subsection 2.1 we present a formal overview of bigraphs. In subsection 2.2 we present Bigraph Reaction Rules (BRR) and Bigraph Reaction Systems (BRS) for modelling dynamics of bigraphs. We finish introducing an algebraic model for dynamical physical structures which aims at reducing non-determinism of BRS by constraining the application of BRRs.

2.1 Bigraphs

For the sake of making the paper complete we present the bigraphs formalism up to a sufficient level for understanding the overall paper. A reader familiar with bigraphs should be able to skip this subsection without compromising the understanding of the remainder of the paper. This explanation follows [18] as the main reference.

A bigraph is a mathematical structure defined essentially by two graphs with some extra machinery mainly for the sake of compositionality. A bigraph has a place graph - a forest that represents nested locality of components; and a link graph - an hypergraph that models connectivity between components. Figure 4 presents an example of a bigraph.



(a) Bigraph. (b) Placing and Linking graphs.

Figure 4: Example of a bigraph and the corresponding place and link graphs.

Place graphs are contained inside *regions* [2] and may also contain *holes*[3]. A hole of a given bigraph can be replaced by a region of another bigraph by means of composition. A node can have *ports*[4] which can be seen intuitively as points for link connections. Ports are assigned to nodes by *controls* which define the *arity* of a node (the number of ports). A set of controls is called a signature.

Definition 1. A *signature* Σ is a tuple $(\mathcal{K}, ar)$ where $\mathcal{K}$ is a set of kinds of nodes, called controls, and $ar : \mathcal{K} \to \mathbb{N}_0$ assigns an arity to each control.

For example, the bigraph of Figure 4(a) has the following signature: $\mathcal{K} = \{M : 1, Q : 2\}$.

In addition to hyperedges, a link graph may also contain *inner names* and *outer names*[5]. Just as one can fit regions inside holes, one can also merge inner names and outer names by means of composition.

Definition 2. An *interface* is a pair $\langle n, X \rangle$ where $n = \{0, 1, \ldots, n-1\}$ denotes the set of holes (or regions) and X denotes the set of inner names (or outer names). Holes and inner names are collectively called an *inner face* while regions and outer names are called an *outer face*. Interfaces will be useful later for defining composition.

Definition 3. A *bigraph* is formally a 5-tuple of the form:

$$(V, E, ctrl, prnt, link) : \langle m, X \rangle \to \langle n, Y \rangle$$

where V is a set of nodes, E is the set of hyperedges, $ctrl : V \to \mathcal{K}$ is a control map that assigns controls to nodes (i.e. arities), $prnt : m \uplus V \to V \uplus n$ is the parent map and defines the nested place structure, $link : X \uplus P \to E \uplus Y$ is the link map and defines the link structure. P denotes the set of ports of the bigraph and is formalized as $P = \{(v, i) | i \in \{0, 1, \ldots, ar(ctrl(v)) - 1\}\}$. $\langle m, X \rangle \to \langle n, Y \rangle$ provides the inner face and outer face of the bigraph, i.e. $\langle m, X \rangle$ is the

[1]This "container" is actually a bigraph region which is introduced in the next section.

[2]Regions are graphically represented by dashed rectangles and are also known as *roots*

[3]Holes are graphically represented as gray rectangles and are also known as *sites*

[4]Ports are graphically represented as black dots on the node.

[5]Names are graphically represented by a line connected on one tip to a port or an edge and the other tip is left loose.

inner face and $\langle n, Y\rangle$ is the outer face. The fact that bigraphs are defined as arrows with interfaces as domain and co-domain comes from category theory. Milner formalizes bigraphs as arrows in a symmetric partial monoidal category where objects are interfaces [18]. The reader does not need to be familiar with category theory for understanding this paper.

Example 2. The bigraph of Figure 4(a) is formally defined as follows.

$$B = (V, E, ctrl, prnt, link) : \langle 1, x\rangle \to \langle 1, y\rangle$$

where $V = \{m_0, m_1, m_2, q_0, q_1\}$, $E = \{e\}$,

$$ctrl(v) = \begin{cases} M : 1, & \text{if } v = m_i,\ i \in \{0,1,2\} \\ Q : 2, & \text{if } v = q_j,\ j \in \{0,1\} \end{cases}$$

$$prnt(v) = \begin{cases} 0, & \text{if } v = m_2 \\ m_0, & \text{if } v = q_0 \\ m_1, & \text{if } v = q_1 \text{ or } v = 0 \\ m_2, & \text{if } v = m_i,\ i \in \{0,1\} \end{cases}$$

$$link(l) = \begin{cases} e, & \text{if } l = (m_i, 0) \text{ or } l = (q_i, 0),\ i \in \{0,1\} \text{ or } l = x \\ y, & \text{if } l = (q_i, 1),\ i \in \{0,1\} \text{ or } l = (m_2, 0) \end{cases}$$

Definition 4. Let

$$A = (V_A, E_A, ctrl_A, prnt_A, link_A) : \langle m_a, X_a\rangle \to \langle n_a, Y_a\rangle$$

and

$$B = (V_B, E_B, ctrl_B, prnt_B, link_B) : \langle m_b, X_b\rangle \to \langle n_b, Y_b\rangle$$

The *composition* of bigraphs A and B, denoted by $A \circ B$, is a bigraph

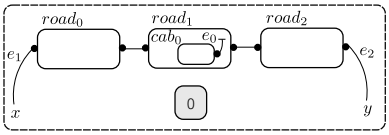
$$C = (V_C, E_C, ctrl_C, prnt_C, link_C) : \langle m_b, X_b\rangle \to \langle n_a, Y_a\rangle$$

where $V_C = V_A \uplus V_B$, $E_C = E_A \uplus E_B$, $ctrl_C = ctrl_A \uplus ctrl_B$. $prnt_C$ is obtained by "filling" the holes of A with regions of B while $link_C$ is obtained by "matching" the inner names of A with the outer names of B. By convention, the regions are matched to holes with the same indices and inner names are matched with outer names with the same name. For a formal definition of $prnt_C$ and $link_C$ please see [18], page 17. The composition $A \circ B$ is well defined iff the outer face of B is equal to the inner face of A, i.e. $\langle n_b, Y_b\rangle = \langle m_a, X_a\rangle$.

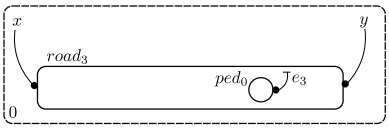
Figure 5 shows an example of composition. In this example, B' is composed with B'' to form the bigraph B in Figure 2. Composition is useful for several reasons. One possible use is abstraction, since it allows abstracting details from the models that might not be relevant or accessible at a given time. For example the bigraph B' of Figure 5(a) might represent a piece of information that got further complemented by another piece of information (bigraph B'' of Figure 5(b)) to form $B = B' \circ B''$ (Figure 2), a more complete structure of the physical world. The use of composition as means for abstraction and refinement is motivated by [19].

Milner also introduced another composition operator called the *tensor product*.

Definition 5. Let A and B be bigraphs that do not share inner or outer names. Then the *tensor product* of A and B, denoted by $A \otimes B$, is the juxtaposition of both bigraphs



(a) $B' : \langle 1, \{\mathtt{x}, \mathtt{y}\}\rangle \to \langle 1, \emptyset\rangle$



(b) $B'' : \langle 0, \emptyset\rangle \to \langle 1, \{\mathtt{x}, \mathtt{y}\}\rangle$

Figure 5: Example of two bigraphs, A and B where $A \circ B$ is the bigraph in Figure 2.

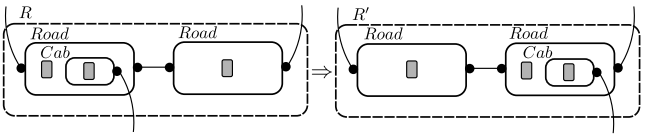


Figure 6: BRR specifying a cab moving from one road to another. This BRR is called $MOVE$.

where indices of holes and regions of B are made unique with respect to A. The names of tensor product are the union of the names of both bigraphs. The tensor product will be useful later when we introduce the concept of regions of influence.

2.2 Dynamics of physical structures

Bigraphs can model static physical structures. Dynamics are introduced by means of Bigraph Reaction Rules (BRR).

Definition 6. A *bigraph reaction rule* is a tuple (R, R', η) where R and R' are bigraphs called respectively *redex*[6] and *reactum*. The redex is the portion of the bigraph to be matched and the reactum is the bigraph that replaces the matched portion. η is called the instantiation map and indicates how parameters matched by holes in R are matched in R'. If η is the identity map, then we represent the rule as $R \to R'$. For the reminder of the paper we always assume η is the identity map, i.e. region i matches with hole i for all $i \in \mathbb{N}$.

Let $r = R \to R'$ be a reaction rule and A a bigraph. In order to perform the reaction r in A we need first to decompose A into $C \circ R \circ d$ where C represents the context and d represents the parameters inside the holes of R. Assuming that η is the identity map, we compose C with the reactum R' and with d to get the resulting A', i.e.

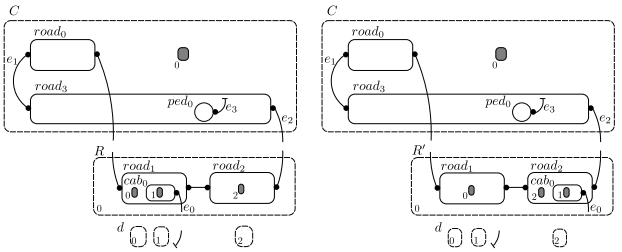
$$A = C \circ R \circ d \to A' = C \circ R' \circ d \tag{4}$$

Example 3. Consider the BRR of Figure 6. The rule models a cab moving from one road to another. Note that the rule is parametric, i.e. it can be applied regardless the nodes inside the road nodes and the cab node. For example, this rule allows modelling the cab travelling with or without passengers. Also note that redex and reactum are abstract bigraphs, i.e. nodes are only represented by their controls which can be seen as node's type. This allows the same

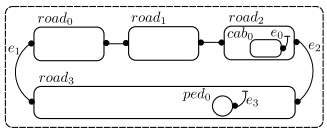
[6]Redex stands for "reducible expression".

rule to be applied to any concrete bigraph that matches the structure of the redex.

Figure 7 shows the steps for applying $MOVE$ from Figure 6 to the bigraph in Figure 2. Figure 7(a) shows a valid decompostion of the bigraph of Figure 2 such that $B = C \circ R \circ d$. Figure 7(b) shows the replacement of the redex with the reactum and Figure 7(c) shows the result of composing the context C with the reactum R' and with the parameters d.



(a) Decomposition of B. (b) Replace redex by reactum.



(c) $C \circ R' \circ d$.

Figure 7: Example of the application of a BRR.

Note that if we add a pedestrian node in region 1 of d than we could model a passenger moving along with a cab.

Let us now define a bigraph reaction system as per Perrone et al. [19].

Definition 7. A *bigraph reaction system* (BRS) is a tuple $(B, \mathcal{R})$ where B is a bigraph and $\mathcal{R}$ is a set of reaction rules.

Next we define a trace of a BRS according to the same reference [19].

Definition 8. A *trace* of a BRS $(B, \mathcal{R})$ is a sequence of bigraphs

$$t = (B_0, B_1, \ldots)$$

where $B_0 = B$ and for each B_i and B_{i+1} there exists a reaction rule $R \to R' \in \mathcal{R}$ a context C and parameters d such that for each i

$$B_i = C \circ R \circ d \to B_{i+1} = C \circ R' \circ d$$

The set of all traces starting from B and generated by rules $\mathcal{R}$ is denoted by $\mathcal{T}_B^{\mathcal{R}}$.

In a BRS $(B, \mathcal{R})$, a rule $R \to R'$ from $\mathcal{R}$ can be applied non-deterministically to any redex from B that matches the rule. This is due to the fact that R and R' are abstract bigraphs and thus may not have a unique match. We introduce a more deterministic approach where one is able to specify where rules must be applied. This is specified by defining control actions which are pairs (r, h) where r is a BRR and h is a bigraph node. The bigraph node must be part of the redex and the reactum when the BRR is applied. One can think of this node as a degree of freedom for specifying where the rule must be applied.

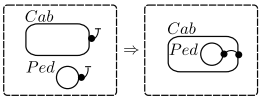


Figure 8: HOPON rule.

Definition 9. A *physical structure* is a tuple

$$\mathcal{B} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

where $\mathbb{B}$ is the bigraph space, $\mathcal{R}$ is a set of reaction rules, B_0 is the initial bigraph, $\mathbb{U} \subseteq \mathcal{R} \times V_{\mathbb{B}}$ is the control space (we denote $V_{\mathbb{B}}$ as the set of bigraph nodes), and $F : \mathbb{B} \times \mathbb{U} \hookrightarrow \mathbb{B}$ is the transition function (which is partially defined). The transition function takes the current bigraph B_i, and a control action $u_i \in \mathbb{U}$ to produce $B_{i+1} = F(B_i, u_i)$. F is given as follows:

$$F(B, (R \to R', h)) = \begin{cases} C \circ R' \circ d & \text{if } \exists dec3.dec3(B) = (C, R, d) \\ & \text{and } h \in V_R \text{and } h \in V_{R'} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5}$$

where $dec3 : \mathbb{B} \to \mathbb{B} \times \mathbb{B} \times \mathbb{B}$ provides a valid decomposition of a bigraph into three other bigraphs, i.e. $dec3(B) = (B', B'', B''')$ such that $B = B' \circ B'' \circ B'''$, V_R is the set of nodes matched by the redex and $V_{R'}$ is the set of nodes matched by the reactum. Each control action $u \in \mathbb{U}$ is a tuple $u = (r, h)$ where $r = R \to R'$ is a reaction rule and $h \in V_{\mathbb{B}}$ is a bigraph node which is required to be in the redex and the reactum. Have in mind that F may be undefined for some pairs (B, u).

Note that $(B_0, B_1 = F(B_0, u_0), \ldots, B_{i+1} = F(B_i, u_i), \ldots)$ forms a bigraph trace.

Example 4. Consider the cab and pedestrian example where *pedAg* has two reaction rules: HOPON and HOPOFF. HOPON is given in Figure 4. HOPOFF is the HOPON rule with redex and reactum inter-changed. The physical structure for this example is defined as:

$$\mathcal{B} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

where

$$\mathcal{R} = \{MOVE, HOPON, HOPOFF\}$$

$$\mathbb{U} = \{(HOPON, v), (HOPOFF, v), (MOVE, v') \\ | ctrl(v) = Ped \wedge ctrl(v') = Cab\}$$

The control space is defined such that for $u \in \mathbb{U}$, if $u = (HOPON, v)$ or $u = (HOPOFF, v)$ then v is of type Ped, and if $u = (MOVE, v')$ then v' is of type Cab, where $v, v' \in V_{\mathbb{B}}$. B_0 is the bigraph in Figure 2. The bigraph space $\mathbb{B}$ is the set of all bigraphs obtained by applying control actions through F (given by equation 5) starting at B_0.

For example, the control action $u = (MOVE, cab_0)$ on the initial bigraph given by Figure 2 is modelled as:

$$B_1 = F(B_0, u).$$

F is defined for this control action since it is possible to find a redex that includes cab_0. The resulting bigraph B_1 is in Figure 7(c).

3. BIAGENTS

In this section we introduce BiAgents - an algebraic model for agents operating over physical structures. The motivation behind modelling agents as algebraic structures comes from [20]. BiAgents are hosted by nodes from the physical structure, and are able to observe it, and compute the map from observations to control actions and migration actions. Control actions affect the physical structure by a semantics described in this section.

We start defining single BiAgents operating over a physical structure and then address the problem of multiple agents operating concurrently over the same physical structure.

3.1 Single BiAgent over a physical structure

We start by defining memoryless BiAgents, i.e. agents that only depend on the current information of the physical structure to compute the next action.

Definition 10. A *memoryless BiAgent* is a tuple

$$a = (\mathcal{O}, \mathcal{U}, host_0, obs, ctr, mgrt)$$

where $\mathcal{O} \subseteq \mathbb{B}$ is the observation space, $\mathcal{U} = \mathcal{R}_a \times V_{\mathbb{B}}$ is the control space ($\mathcal{R}_a$ denotes a set of BRRs), $host_0 \in V_{\mathbb{B}}$ is the bigraph node that hosts the agent, $obs : \mathbb{B} \times V_{\mathbb{B}} \to \mathcal{O}$, is the observation map that, given a bigraph and a host node, provides an observation, $ctr : \mathcal{O} \to \mathcal{U}$ is the control[7] map that given an observation produces a control action, and $mgrt : V_{\mathbb{B}} \times \mathcal{O} \to V_{\mathbb{B}}$ is the migration map which, given a new observation and the current host, provides the next host.

Note that BiAgents are partially defined. We need to compose them with a physical structure to make them meaningful.

Definition 11. Let $a \bullet \mathcal{B}$ denote a BiAgent

$$a = (\mathcal{O}, \mathcal{U}, host_0, obs, ctr, mgrt)$$

operating over a physical structure

$$\mathcal{B} = (\mathbb{B}, \mathcal{R}, \mathbb{U}, B_0, F)$$

Let $(B_0, B_1, B_2, \ldots)$ be a bigraph trace of $\mathcal{B}$. Then a can perform one of the following operations on $\mathcal{B}$ at a given time i:

1. **Observe**: $obs(B_i, h_i^a)$ where h_i^a denotes the host of a at time i
2. **Migrate**: $mgrt(h_i^a, o)$ which provides the next host of a considering the current host and some observation o
3. **Act**: $u_i = ctr(o)$ which provides the next control action for some observation o. u_i is then applied to the transition function of $\mathcal{B}$ producing $B_{i+1} = F(B_i, u_i)$

Note that in order to migrate or act, the BiAgent must observe the physical structure at least once. Also note that hosts make BiAgents local operators, i.e. they can locally observe the underlying bigraph and act or migrate over a confined region. Moreover, changing a BiAgent host may change what it can observe and what part of the bigraph it can change.

We call observations, migrations and actions as *events*.

[7]Note that we use the term "control" in the sense of control theory. Do not confuse with "controls" which specify the arity of bigraph nodes.

Example 5. Consider $pedAg \bullet \mathcal{B}$ where $pedAg$ is a memoryless BiAgent defined as follows:

$$pedAg = (\mathcal{O}_p, \mathcal{U}_p, h_0^p, obs_p, ctr_p, mgrt_p)$$

where $\mathcal{O}_p$ is the observation space (all possible observations of the physical structure),

$$\mathcal{U}_p = \{(HOPON, ped_0), (HOPOFF, ped_0)\}$$
$$h_0^p = ped_0$$
$$ctr_p(o_i) = \begin{cases} (HOPON, h_i^p) & \text{if } c : Cab \in V_{o_i} \wedge \\ & prnt(h_i^p) \neq destRoad \\ (HOPOFF, h_i^p) & \text{if } prnt(h_i^p) : Cab \wedge \\ & prnt(prnt(h_i^p)) = destRoad \end{cases}$$

and $mgrt_p(o_i, h_i^p) = ped_0$

The observation map $o_i = obs_p(B_i, h_i^p)$ is defined such that o_i is a bigraph that contains the nodes $h_i^p = ped_0$, $prnt(h_i^p)$ (i.e. the current road) and all the children of $prnt(h_i^p)$, i.e. everything contained on the current road. Recall Example 1, equation 2, for an example of a trace of $pedAg$.

BiAgents defined at Definition 10 are memoryless due to the fact observations only entail information from a single bigraph which is assumed to be a projection of the current bigraph. In order to enable control and migration which takes the overall structural history we redefine BiAgents overloading the observation space such that each observation is now a trace of bigraphs.

Definition 12. A *BiAgent* is a tuple

$$a = (\mathcal{O}, \mathcal{U}, host_0, obs, ctr, mgrt)$$

where $\mathcal{O}$ is overloaded as $\mathcal{O} \subseteq \mathcal{T}_{\mathbb{B}}$ which $\mathcal{T}_{\mathbb{B}}$ denotes every possible bigraph trace. We must also override obs such that the domain is now traces of tuples of the form (B, h). Thus, $obs : \bigcup_{i=1}^{\infty} (\mathbb{B} \times V_{\mathbb{B}})^i \to \mathcal{O}$. A BiAgent allows the implementation of controllers and migration algorithms as functions of the overall structural history. For the sake of simplicity, all the examples in this paper are memoryless.

$a \bullet \mathcal{B}$ may not be always defined since a may request a control action u_i such that $F(B_i, u_i)$ is undefined or may try to migrate to a node which does not exist in B_i. We denote these events as *ill-defined events*.

Example 6. Recall Example 5. Consider that due to an incorrect observation $pedAg$ observes a cab at B_0 (see Figure 2) although it is not there. $pedAg$ requests the execution of $u = (HOPON, ped_0)$. $F(B_0, u)$ is undefined which makes u an ill-defined event.

Definition 13. Consider $a \bullet \mathcal{B}$. a is *feedback controller* of $\mathcal{B}$ if at each instance i and for an observation $o_i = obs_a(B_i, h^a)$, $u_i = ctr_a(o_i)$ such that $F(B_i, u_i)$ is defined, i.e. a always choose well-defined control actions for a given observed bigraph.

Definition 14. Consider $a \bullet \mathcal{B}$. a is *feedback migrator* of $\mathcal{B}$ if at each instance i and for an observation $o_i = obs_a(B_i, h^a)$, $h_{i+1}^a = mgrt_a(h_i^a, o_i)$ such that $h_{i+1}^a \in V_{B_i}$, i.e. a always migrates to nodes of the observed bigraph.

4. CONCURRENCY BETWEEN BIAGENTS

The composition of a set of BiAgents $\mathcal{A}$ with a physical structure $\mathcal{B}$ is denoted as $\mathcal{A}\bullet\mathcal{B}$. We assume an asynchronous interleaved *writing* semantics [15], i.e. two agents cannot act or migrate simultaneously . The order in which agents can operate is non-deterministically determined. Figure 1 presents a block diagram depicting the closed-loop interactions between BiAgents and a physical structure.

We must be able to formally assert if an execution of $\mathcal{A}\bullet\mathcal{B}$ is well-defined. In order to be able to make such assertions we need to formalize the concept of a BiAgent trace of $\mathcal{A}\bullet\mathcal{B}$.

4.1 BiAgent traces

In this subsection we discuss which conditions are necessary for a BiAgent trace to be well-defined. Let us remind the concept of a BiAgent trace by stating it formally.

Let (B, H) be a BiAgent instance where $B \in \mathbb{B}$ is a bigraph and $H : \mathcal{A} \to V_{\mathbb{B}}$ be a host distribution function. Consider the relation $\prec$ between BiAgent instances such that

$$(B_i, H_i) \prec (B_{i+1}, H_i) \text{ or}$$
$$(B_i, H_i) \prec (B_i, H_{i+1})$$

i.e. given $(B_i, H_i) \prec I$, I is either a change in the physical structure or a change in the hosting distribution due to a migration.

Definition 15. A *BiAgent trace* is a sequence:

$$t = (t_0, t_1, t_2, \ldots) \text{ where } t_i \prec t_{i+1}, i \in \mathbb{N}. \quad (6)$$

For the sake of readability we represent traces as:

$$t = t_0 \prec t_1 \prec t_2 \prec \ldots \quad (7)$$

The ith element of the sequence t is denoted as t_i. Sometimes we stack a label on top of the relation symbol $\prec$ denoting which event triggered the change between the two BiAgents instances. Recur to Example 1 for examples of BiAgents traces.

We proceed by defining the conditions that make a BiAgent trace a well-defined execution trace of $\mathcal{A}\bullet\mathcal{B}$.

Definition 16. Let $t = t_0 \prec t_1 \prec \ldots$ where $t_i = (B_i, H_i)$. t is a *well-defined BiAgents trace* of $\mathcal{A}\bullet\mathcal{B}$ if:

$$\forall i.\forall a.H_i(a) \in V_{B_i} \quad (8)$$
$$\forall i.B_{i+1} = F(B_i, u) \text{ for some } u \quad (9)$$

The first condition requires that, at every BiAgent instance, all agents must hosted by an node of the current bigraph. The second condition requires the next bigraph B_{i+1} to be obtained by $F(B_i, u)$.

THEOREM 1. *Consider a trace t of a model $a \bullet \mathcal{B}$. If a is feedback controller and feedback migrator then t is well-defined.*

PROOF. Since there is only one agent and it is feedback controller then, by Definition 13, for each i, $B_{i+1} = F(B_i, u_i)$ where u_i is choose such that F is defined. Furthermore, by Definition 14, a always migrates to a node in the current bigraph. Thus $H_i(a) \in V_{B_i}$. Since both conditions of Definition 16 are satisfied then t is well-defined. □

According to Theorem 1, a single agent which is both feedback controller and migrator operating over a physical structure always produces well-defined BiAgent traces. Although, this is not guaranteed for multiple agents.

Example 7. Recall agents *pedAg* and *cabAg* of Example 1. Assume that both agents are feedback controllers and feedback migrators. Consider that *pedAg* gets an observation of the physical structure that contains cab_0 (which hosts *cabAg*). Then it chooses to apply the rule $u = (HOPON, ped_0)$ (which is a well-defined event up to the observed bigraph). Although, before this rule is applied *cabAg* applies the rule $u' = (MOVE, cab_0)$ which makes the cab move to another road. This makes $F(B, u)$ undefined and, thus, u an ill-defined event. These behaviours can be seen as a form of race-condition. We denote them as *ill-defined concurrent behaviours*.

In order to observe these behaviours from BiAgent traces, the traces must be labelled with observation, action and migration events. Thus we introduce a new definition for BiAgent traces, where each relation symbol on the trace is labelled with an event and with information of which agent performed each event. In order to define these new traces we introduce the following relation:

$$I \overset{\lambda}{\triangleright} I'$$

where I and I' are BiAgents instances and $\lambda \in \{obs^a, ctr^a, mgrt^a\}$ is a label that indicates which kind of event was executed: *obs* denotes an observation, *ctr* denotes a control action, and *mgrt* denotes a migration. The BiAgent that performed the event is appended as a superscript.

Example 8. Consider a model $\{a_0, a_1\} \bullet \mathcal{B}$. Let t be an example of a BiAgents trace according to the new definition:

$$t = (B_0, H_0) \overset{obs^{a_0}}{\triangleright} (B_0, H_0) \overset{ctr^{a_0}}{\triangleright} (B_1, H_0) \overset{obs^{a_1}}{\triangleright} (B_1, H_0) \overset{mgrt^{a_1}}{\triangleright} (B_1, H_1) \quad (10)$$

t shows that a_0 observed and changed the physical structure and later a_1 observed and migrated into another node.

There are at least two basic strategies for avoiding these ill-defined concurrent behaviours: prevent interleaving between observation and action events; and prevent "sharing" resources of the physical structure. We formalize these strategies and introduce a third one which is a combination of the first two. We also discuss the price to pay in terms of scalability and/or expressiveness.

4.2 Atomic read-write semantics

This technique prevents ill-defined concurrent behaviours by forcing atomicity between an observation and an action/migration. In other words, if a BiAgent a_0 observes the physical structure, no other BiAgent can either observe or act/migrate until a_0 decides to act/migrate. For simplicity we assume that BiAgents are equipped with an identity reaction rule $id_h = B_h \to B_h$ (where B_h is a bigraph with a single node, h) that does not change the physical structure. This rule can be used whenever a BiAgent wants to observe but not to change the physical structure.

Definition 17. A model $\mathcal{A}\bullet\mathcal{B}$ has *atomic read-write semantics* if for every BiAgents trace

$$I_0 \overset{\lambda_0}{\triangleright} I_1 \overset{\lambda_1}{\triangleright} I_2 \overset{\lambda_2}{\triangleright} \ldots$$

if $\lambda_i = obs^a$ then λ_{i+1} is either ctr^a or $mgrt^a$.

THEOREM 2. *Let $\mathcal{A}$ be a set of feedback controllers & migrators. Consider a trace t of a model $\mathcal{A} \bullet \mathcal{B}$ with atomic read-write semantics. Then t is well-defined.*

PROOF. Without lost of generality, consider $\mathcal{A} = \{a_0, a_1\}$ where a_0 and a_1. By assumption, both are feedback controllers & migrators. Since they operate under an atomic read-write semantics the possible interleaving of events are:

$$I \overset{obs^{a_0}}{\triangleright} I \overset{ctr^{a_0}/mgrt^{a_0}}{\triangleright} I' \overset{obs^{a_1}}{\triangleright} I' \overset{ctr^{a_1}/mgrt^{a_1}}{\triangleright} I''$$

and

$$J \overset{obs^{a_1}}{\triangleright} J \overset{ctr^{a_1}/mgrt^{a_1}}{\triangleright} J' \overset{obs^{a_0}}{\triangleright} J' \overset{ctr^{a_0}/mgrt^{a_0}}{\triangleright} J''$$

Since for each of the above traces, BiAgent instances at observation time for a given agent are the same as the BiAgent instances at control/migration time for the same agent, then the agent is always going to perform a well-defined event. □

Example 9. Consider Example 1 with agents *cabAg* and *pedAg* operating under an atomic read-write semantics. The following trace is an example of a possible execution:

$$(B_1, H_0) \overset{obs^{cabAg}}{\triangleright} (B_1, H_0) \overset{MOVE}{\triangleright} (B_2, H_0) \overset{obs^{pedAg}}{\triangleright}$$
$$(B_2, H_0) \overset{HOPON}{\triangleright} (B_3, H_0)$$

The trace shows that *cabAg* observed B_1 and change it to B_2 (by applying rule $MOVE$), then *pedAg* observed the structure and changed it to B_3 (by applying the rule $HOPON$).

Atomic read-write semantics is a simple technique and usually not too cumbersome to implement. Although, since it implies blocking the overall physical structure, this technique may introduce scalability problems. These issues are more critical when the number of agents is too large, or agents perform heavy computations.

4.3 Partitioning semantics

Next we introduce a strategy for preventing ill-defined concurrent behaviours. This strategy relies on the fact that BiAgents operate locally. The main idea is if two agents operate over different regions they do not share resources and, thus, they do not race. We call the region where an agent can operate its *region of influence*.

Definition 18. The region of influence of an agent a over a bigraph B is a set of bigraph nodes from B that contains the host of a and all the nodes from the redexes of rules $\mathcal{R}_a$ applied B with the current host of a. Let $regInf : \mathcal{A} \times V_{\mathbb{B}} \times \mathbb{B} \to \mathcal{P}(V_{\mathbb{B}})$ be a function that takes an agent a, the host of a, and the current bigraph B, and returns the region of influence of a. $regInf$ is formally given by:

$$regInf(a, h^a, B) = V \tag{11}$$

where

$$V \subseteq V_B \tag{12}$$
$$h^a \in V \tag{13}$$
$$\forall (R \to R') : \mathcal{R}_a. \tag{14}$$
$$(\exists dec3.dec3(B) = (C, R, d) \wedge h^a \in V_R) \Rightarrow V_R \subseteq V$$

Intuitively, $regInf(a, h^a, B)$ gives the nodes from B that can be affected by the application of BRRs of BiAgent a when it is hosted by h^a.

For example, recall Figure 3. The region of influence of *pedAg* at B_0 and B_1 is $\{ped_0\}$. At B_2 and B_3 the region of influence changes to $\{ped_0, cab_0\}$.

THEOREM 3. *V_B is the least upper bound (supremum) of $regInf(a, h^a, B)$.*

PROOF. By Definition 18, since $regInf(a, h^a, B) \subseteq V_B$, then V_B is an upper bound of $regInf(a, h^a, B)$. $regInf(a, h^a, B) = V_B$ when the redexes of rules from $\mathcal{R}_a$ cover the overall bigraph B. Thus, V_B is the least upper bound of $regInf(a, h^a, B)$. □

THEOREM 4. *h^a is the greatest lower bound (infimum) of $regInf(a, h^a, B)$.*

PROOF. By the Definition 18, $h^a \in regInf(a, h^a, B)$ thus h^a is a lower bound of $regInf(a, h^a, B)$. It is also the greatest lower bound since the only other lower bound is $\emptyset$ and $regInf(a, h^a, B)$ is always non-empty (it contains at least h^a). □

For the remainder of the paper, assume that agents are only able to observe, act and migrate over their regions of influence.

We use the concept of regions of influence for defining partitioned semantics.

Definition 19. A model $\mathcal{A} \bullet \mathcal{B}$ has partitioning semantics if, for every possible BiAgents trace

$$I_0 \overset{\lambda_0}{\triangleright} I_1 \overset{\lambda_1}{\triangleright} I_2 \overset{\lambda_2}{\triangleright} \dots$$

if $\lambda_i = obs^a$ then λ_{i+1} is given by one of the following cases:

1. $\lambda_{i+1} = e^a$, where e^a is an arbitrary event (an observation, migration or action) of a
2. $\lambda_{i+1} = e^{a'}$, where $e^{a'}$ is an arbitrary event of a' outside the region of influence of a.

For each bigraph B and hosting distribution H such that $H(a_i) = h^{a_i}$[8], $P = \{r | r = regInf(a_i, h^{a_i}, B), a_i \in \mathcal{A}\}$ forms a partition of B[9]. Note that this partition may change over time due to migration of agents (which changes the location where rules are locally applied) or due to changes in the physical structure.

Example 10. Figure 9 shows a snapshot of an execution of four BiAgents (represented as stars over their host node). a_2 and a_3 have disjoint regions of influence and thus can not violate the partitioning semantics. The regions of influence of a_0 and a_1 overlap which may lead to a ill-defined concurrent behaviour. The regions of a_1 and a_2 are disjoint although the nodes share a link. Nonetheless, this is still acceptable as per the partitioning semantics.

THEOREM 5. *Let $\mathcal{A}$ be a set of BiAgents which are feedback controllers & migrators. Consider a trace t of the model $\mathcal{A} \bullet \mathcal{B}$ with partitioned semantics. Then t is well-defined.*

[8]For the sake of an uncluttered notation we drop the time index i whenever the understanding of the example does not depend on it.

[9]Without lost of generality we assume that the partition fully covers V_B. Although this might not be true, the parts of V_B that are not covered by the partition are irrelevant regarding the analysis of ill-defined behaviours.

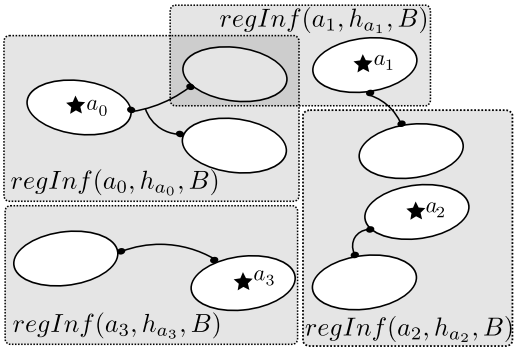


Figure 9: Example of regions of influence of agents.

PROOF. Consider the trace:

$$I_0 \overset{\lambda_0}{\triangleright} I_1 \overset{\lambda_1}{\triangleright} I_2 \overset{\lambda_2}{\triangleright} \dots$$

Since the model has partitioning semantics if $\lambda_i = obs^a$ then λ_{i+1} is given by one of the cases of Definition 19. If it is given by case 1, then λ_{i+1} is an event from a, which is by assumption a feedback controller & migrator. Thus, λ_{i+1} is a well-defined event. If λ_{i+1} is given by case 2, then it is an event from another agent a' outside the region of influence of a. Since a' is also assumed to be feedback controller & migrator then λ_{i+1} is also well-defined since the previous event was just an observation of a which does not change the BiAgent instance. □

Next we prove that atomic read-write semantics is a stronger requirement than partitioned-based semantics.

THEOREM 6. *If a model $\mathcal{A} \bullet \mathcal{B}$ has atomic read-write semantics then it also has partitioning semantics.*

PROOF. Since $\mathcal{A}\bullet\mathcal{B}$ has atomic read-write semantics then, for all traces $I_0 \overset{\lambda_0}{\triangleright} I_1 \overset{\lambda_1}{\triangleright} I_2 \overset{\lambda_2}{\triangleright} \dots$, if $\lambda_i = obs^a$ then λ_{i+1} is either ctr^a or $mgrt^a$, which also satisfies the first condition of Definition 19. Thus, $\mathcal{A} \bullet \mathcal{B}$ has also partitioned-based semantics. □

Next we propose two techniques that uses Milner's bigraphical composition operators $\circ$ and $\otimes$ to define partitioning semantics for a model $\mathcal{A} \bullet \mathcal{B}$.

4.3.1 Tensor-based partition

Definition 20. A bigraph B and host distribution H form a *tensor-based partition* $P = \{V_{B^{a_0}}, V_{B^{a_1}}, \dots, V_{B^n}\}$ if $B = B^{a_0} \otimes B^{a_1} \otimes \dots \otimes B^{a_n}$, where $V_{B^{a_k}} = regInf(a_k, H(a_k), B)$ and $n = |\mathcal{A}| - 1$.

THEOREM 7. *Let $\mathcal{A}\bullet\mathcal{B}$ be a concurrent model. Let B_i denote the current bigraph at time i and H_i the current hosting distribution. If, for each i, B_i and H_i form a tensor-based partition then $\mathcal{A} \bullet \mathcal{B}$ has partitioning semantics.*

PROOF. By definition 5, if $B_i = B_i^{a_0} \otimes B_i^{a_1} \otimes \dots \otimes B_i^{a_{|\mathcal{A}|-1}}$ is well defined then all $B_i^{a_k}$ have disjoint nodes and links. Thus, for every two agents a and a', $regInf(a, h^a, B_i) \cap regInf(a', h^{a'}, B_i) = \emptyset$. Which means that for every trace t, if $\lambda_i = obs^a$ (a observes the physical structure B_i) then λ_{i+1} can be either e^a or $e^{a'}$ where a and a' have disjoint regions of influence. □

Note that Definition 20 requires a non-overlapping partition of the physical structure which might be impractical. Next we introduce another technique in which the partition is required to be defined by the bygraph composition $\circ$. This composition operator allows sharing of links between regions thus producing a partition weaker then the tensor-based partition.

4.3.2 Composition-based partition

Definition 21. A bigraph B and host distribution H form a *tensor-based partition* $P = \{V_{B^{a_0}}, V_{B^{a_1}}, \dots, V_{B^{a_n}}\}$ if $B = B^{a_0} \circ B^{a_1} \circ \dots \circ B^{a_n}$, where $V_{B^{a_k}} = regInf(a_k, H(a_k), B)$.

THEOREM 8. *Let $\mathcal{A}\bullet\mathcal{B}$ be a concurrent model. Let B_i denote the current bigraph at time i and H_i the current hosting distribution. If, for each i, B_i and H_i form a composition-based partition then $\mathcal{A} \bullet \mathcal{B}$ has partitioning semantics.*

PROOF. Since the bigraph composition operator also requires disjoint node sets, this proof is similar to the one of Theorem 7. The only difference relies on the fact that bigraph composition allows sharing of edges. Let ed be an edge shared between two nodes n_0 and n_1 of B_i. Consider a decomposition $B_i = B_i^{a_0} \circ B_i^{a_1}$ where n_0 is in $B_i^{a_0}$ and n_1 in $B_i^{a_1}$. Then $B_i^{a_0}$ has an inner name x while $B_i^{a_1}$ as an outer name x such that when the composition is performed the two names fuse together and form the edge ed. Since each BiAgent get a name from "breaking" the edge (respectively an inner and outer name) they can manipulate their own name without affecting the other name. The only requirement is that must be satisfied is that B_{i+1} is still a valid composition of $B_{i+1}^{a_k}$ for each k. □

Enforcing a semantics based on non-overlapping partitions of the physical structure while only being able to share links is still too strong since this strategy prevents nodes being shared by different regions. For example, consider the example with the cab BiAgent and pedestrian BiAgent. At a given point both BiAgents might get access to the cab node at the same time (for example, when the pedestrian is travelling inside the cab). Thus, the regions of influence of both BiAgents must necessarily overlap. Next we propose a solution that allows overlapping partitions.

4.4 Atomicity-by-need semantics

We propose an overlapping partitioning semantics. There aren't any bigraph composition operator that allows overlapping regions. Thus, we treat them as regions *per se*. If two regions overlap, we create three new regions, one per each non-overlapping parts and one for the overlapping part. For example, the bigraph of Figure 9 is decomposed into five different regions: the region of influence of a_2 and a_3 (which do not overlap), the regions of a_0 and a_1 except the overlapping part, and the overlapping region itself. The *atomicity-by-need* strategy consists of operating according to a atomic read-write semantics in the overlapping parts.

The concurrent semantics with overlapping partitions can be formalized as follows. Let B_i be decomposed as:

$$B_i = N_i \circ O_i$$

where $N_i = N_i^{a_0} \circ N_i^{a_1} \circ \dots N_i^{a_n}$ represents the non-overlapping parts of the regions of influence of each agent, and $O_i = O_i^{a_0 \cap a_1} \circ O_i^{a_0 \cap a_1 \cap a_2} \circ \dots$ represents the overlapping regions.

If an agent observes a region from N_i then it operates according to the composition-based concurrent semantics. If an agent observes a region from O_i then it operates over that region according to an atomic read-write semantics.

It is trivial to prove that atomicity-by-need also prevents ill defined concurrent behaviours since this strategy is the combination of two semantics that also prevent these behaviours to occur. The price to pay for implementing this strategy is the required coordination between agents for blocking over the overlapping areas. The asymptotic complexity of checking if areas overlap is $O(m * n^2)$ where $n = |\mathcal{A}|$ and m is the number of nodes on the physical structure.

5. CONCLUSIONS AND FUTURE WORK

In this paper we introduce BiAgents, a concurrent model of computation for modelling networked systems of mobile computing devices which operate over environments with dynamic structure. The model entails a set of algebraically defined agents which operate concurrently over a physical structure modelled as a bigraph reaction system. BiAgents are hosted by nodes of the physical structure and are able to locally observe and act. Moreover, BiAgents are able to migrate from one host to another. We claim that this model offers a natural way of modelling computation that flows in space.

BiAgents may face concurrency issues due to the fact that they operate in a shared environment. We present strategies for preventing ill-defined concurrent behaviours based on a standard atomic read-write semantics and a partitioning semantics. The latter takes advantage of the fact that BiAgents operate locally to create a partition of the physical structure where each agent operates over its own region. We also introduce a hybrid approach called atomicity-by-need which uses a combination of the above strategies. Since BiAgents have both a model of time and space they are a natural abstraction for specifying these strategies.

We choose to model BiAgents algebraically for the sake of separation of concerns. With this approach we keep them abstract from programming models and inter-process communication semantics which are not part of the scope of this paper.

As future work we want to explore different models of concurrency for modelling communication between BiAgents. We are currently working on a new approach using the actor model of concurrency for that purpose. After this step is concluded we plan to address the design of a simple domain specific language for demonstrating the use of BiAgents.

6. REFERENCES

[1] G. Abowd, M. Ebling, G. Hung, H. Lei, and H. Gellersen. Context-aware computing. *Pervasive Computing, IEEE*, 99(3):22–23, 2002.

[2] G. Agha. *Actors: a model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, USA, 1986.

[3] R. Alur and D. Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[4] J. Armstrong. *Programming Erlang: Software for a Concurrent World*. Pragmatic Bookshelf, 2007.

[5] A. Benveniste, B. Caillaud, and P. Le Guernic. From synchrony to asynchrony. *CONCUR?99 Concurrency Theory*, pages 776–776, 1999.

[6] A. Benveniste, P. Caspi, S. Edwards, N. Halbwachs, P. Le Guernic, and R. De Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[7] L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, and H. Niss. Bigraphical models of context-aware systems. In *Foundations of software science and computation structures*, pages 187–201. Springer, 2006.

[8] S. Debois. Computation in the informatic jungle. *To appear. Draft available at* `http://www.itu.dk/people/debois/pubs/computation.pdf`, 2010.

[9] A. Deshpande, A. Gollu, and L. Semenzato. The shift programming langauge for dynamic networks of hybrid automata. In *IEEE Transactions on Automatic Control*, volume 43. April 1998.

[10] J. Epstein, A. Black, and S. Peyton-Jones. Towards haskell in the cloud. In *Proceedings of the 4th ACM symposium on Haskell*, pages 118–129. ACM, 2011.

[11] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous data flow programming language lustre. *Proceedings of the IEEE*, 79(9):1305–1320, 1991.

[12] P. Haller and M. Odersky. Scala actors: Unifying thread-based and event-based programming. *Theoretical Computer Science*, 410(2-3):202–220, 2009.

[13] T. Henzinger. The theory of hybrid automata. In *Logic in Computer Science, 1996. LICS'96. Proceedings., Eleventh Annual IEEE Symposium on*, pages 278–292. IEEE, 1996.

[14] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to automata theory, languages, and computation*. Addison-wesley, 2007.

[15] R. Milner. Calculi for synchrony and asynchrony. *Theoretical computer science*, 25(3):267–310, 1983.

[16] R. Milner. *Communicating and mobile systems: the π-calculus*. Cambridge Univ Press, 1999.

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

[18] R. Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[19] G. Perrone, S. Debois, and T. Hildebrandt. Bigraphical refinement. *Refinement Workshop - Electronic Proceedings in Theoretical Computer Science*, 2011.

[20] B. Staruch. Extensions of partial structures and their application to modelling of multiagent systems. In *Monitoring, Security, and Rescue Techniques in Multiagent Systems*, volume 28, pages 293–304. Springer Berlin, 2005.

[21] S. Tripakis, C. Pinello, A. Benveniste, A. Sangiovanni-Vincent, P. Caspi, and M. Di Natale. Implementing synchronous models on loosely time triggered architectures. *Computers, IEEE Transactions on*, 57(10):1300–1314, 2008.

[22] L. Walton and M. Worboys. An algebraic approach to image schemas for geographic space. *Spatial Information Theory*, pages 357–370, 2009.

[23] M. Zennaro. *A framework for the development of traffic control systems*. PhD thesis, UC Berkeley, 2008.

[24] M. Zennaro and R. Sengupta. Distributing synchronous programs using bounded queues. In

Proceedings of the 5th ACM international conference on Embedded software, pages 325–334. ACM, 2005.