Bigraphical Models of Context-aware Systems

Lars Birkedal
Søren Debois
Ebbe Elsborg
Thomas Hildebrandt
Henning Niss

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: +45 72 18 50 00
Web: www.itu.dk

Bigraphical Models of Context-aware Systems

L. Birkedal, S. Debois, E. Elsborg, T. Hildebrandt, and H. Niss

IT University of Copenhagen (ITU)
{birkedal,debois,elsborg,hilde,hniss}@itu.dk

Abstract. As part of ongoing work on evaluating Milner's bigraphical reactive systems, we investigate bigraphical models of *context-aware systems*, a facet of ubiquitous computing. We find that naively encoding such systems in bigraphs is somewhat awkward; and we propose a more sophisticated modeling technique, introducing *plato-graphical models*, alleviating this awkwardness. We argue that such models are useful for simulation and point out that for reasoning about such bigraphical models, the bisimilarity inherent to bigraphical reactive systems is not enough in itself; an equivalence between the bigraphical reactive systems themselves is also needed.

1   Introduction

The theory of *bigraphical reactive systems*, due to Milner and co-workers, is based on a graphical model of mobile computation that emphasizes both locality and connectivity [14, 18, 21]. A bigraph comprises a place graph, representing locations of computational nodes, and a link graph, representing interconnection of these nodes. We give dynamics to bigraphs by defining reaction rules that rewrite bigraphs to bigraphs; roughly, a bigraphical reactive system (BRS) is a set of such rules. Based on methods of the seminal [15], any BRS has a labelled transition system, the behavioural equivalence (bisimilarity) of which is a congruence.

There are two principal aims for the theory of bigraphical reactive systems: (1) to model ubiquitous systems [28], capturing mobile locality in the place graph and mobile connectivity in the link graph; and (2) to be a meta-theory encompassing existing calculi for concurrency and mobility. To date, the theory has been evaluated only wrt. the second aim: We have bigraphical understanding of Petri nets [17], π-calculus [12, 14, 13], CCS [21], mobile ambients [12], HOMER [4], and λ-calculus [18, 19].

The present paper initiates the evaluation of the first aim. We investigate modeling of *context-aware systems*, a vital aspect of ubiquitous systems. A context-aware application is an application that adapts its behaviour depending on the context at hand [26], interpreting "context" to mean the situation in which the computation takes place [9]. The canonical example of such a situation is the location of the device performing the computation; systems sensitive to location are called *location-aware*. As an example, a location-aware printing system could send a user's print job to a printer close by. (For notions of context different from location, refer to [27]; for large-scale practical examples, see [1].)

To observe changes in the context, context-aware systems typically include a separate context sensing component that maintain a model of the current context. Such models are known as context models [11] or, more specifically, location models [2]. The above-mentioned location-aware printing system would need to maintain a model of the context that supports finding the printer closest to a given device.

Such models are informal. There are only very few formal models of context-aware computing (refer to [10] for an overview). We point out Context Unity [25]; in spirit, our proposal is somewhat closer to process calculi than Context Unity is. On

the other hand, we do use that in bigraphs, we get to make our own reaction rules, an unusual feature for traditional process calculi.

In overall terms, our contribution is two-fold.

– We find, perhaps surprisingly, that naively modeling context-aware systems as BRSs is somewhat awkward; and
– we propose a more sophisticated modeling technique, in which the perceived and actual context are both explicitly represented as distinct but overlapping BRSs. We call such models *Plato-graphical*.

The remainder of this paper is organized as follows. In Section 2, we introduce bigraphs and bigraphical reactive systems. In Section 3, we discuss naive bigraphical models of location-aware systems. In Section 4, we introduce our Plato-graphical models of context-aware systems. In Section 5, we present two example models. In Section 6, we discuss. Finally, in Section 7, we conclude and note future work.

2 Bigraphs and Bigraphical Reactive Systems

We introduce bigraphs by example (formal definitions of [14, 21] are repeated in Appendix A). Readers acquainted with bigraphs may skip this section.
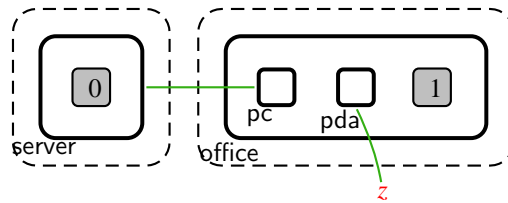
Here is a bigraph, *A*:
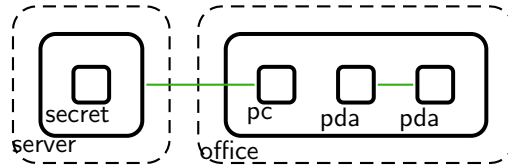
It has *nodes* (vertices), indicated by solid boxes. Each node has a *control*, written in sans serif. Each control has a number of *ports*; ports can be linked by *edges*, indicated by lines. Here, the controls secret and office have no ports, all other controls have one port. Nodes can be nested, indicated by containment. The two outermost dashed boxes indicate *roots*. Roots have no controls; they serve solely to separate different nesting hierarchies.

The bigraph *A* ostensibly models two physically separate locations (because of the two roots). The first contains a server, which in turn contains secret data; the second contains an office, which in turn contains a PC and two PDAs. The server and the PC are connected, as are the PDAs.

Here is another bigraph, *B*:

B resembles *A*, except that the content of server has been replaced with a *site* $-_0$, one of the pdas has been replaced by a site $-_1$, and there is an *inner name z* connected to the remaining pda. Using sites and names, we can define composition of bigraphs. For that, here is yet another bigraph *C*:
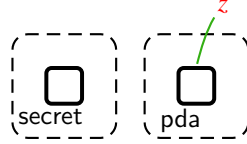
C has an *outer name z*. The bigraphs *B* and *C* compose to form *A*, i.e., $A = B \circ C$. Composition proceeds by plugging the roots of *C* into the sites of *B* (in order), and fusing together the connections $\text{pda} \to z$ (in *C*) and $z \to \text{pda}$ (in *B*) removing the name *z* in the process.

One cannot compose arbitrary bigraphs. For $U \circ V$ to be defined, *U* must have exactly as many sites as *V* has roots, and the inner names of *U* must be precisely the outer names of *V*. The sites and inner names are collectively called the *inner face*; similarly, the roots and outer names are called the *outer face*. *A* has inner face $\langle 0, \emptyset \rangle$ and outer face $\langle 2, \emptyset \rangle$; we write $A : \langle 0, \emptyset \rangle \to \langle 2, \emptyset \rangle$. Similarly, $B : \langle 2, \{z\} \rangle \to \langle 2, \emptyset \rangle$ and $C : \langle 0, \emptyset \rangle \to \langle 2, \{z\} \rangle$.

The graphical representation used above is handy for modeling, but unwieldy for reasoning. Fortunately, bigraphs have an associated term language [6, 16], which we use (albeit in a sugared form) in the sequel. The language is summarized in Table 1. Here are, in order of increasing complexity, term representations of the bigraphs *A*,

| Term	Meaning
$U \parallel V$	Concatenation (juxtaposition) of roots.
$U \mid V$	Concatenation (juxtaposition) of children. (collect the children of *U* and *V* under one root.)
$U \circ V$	Composition.
$U(V)$	Nesting. *U* contains *V*.
$\mathsf{K}_{\vec{x}}(U)$	Ion. Node with control K of arity $|\vec{x}|$, ports connected to the outer names of vector $\vec{x}$. The node contains *U*.
1	The *barren* (empty) root.
$-_i$	Site numbered *i*.
$/x.U$	*U* with outer name *x* replaced by an edge.
x/y	Connection from inner name *y* to outer name *x*.

Table 1. Sugared term language for bigraphs.

3

B and *C*.

$$C = \mathsf{secret} \parallel \mathsf{pda}_z$$
$$A = /x./y.\mathsf{server}_x(\mathsf{secret}) \parallel \mathsf{office}(\mathsf{pc}_x \mid \mathsf{pda}_y \mid \mathsf{pda}_y)$$
$$B = /x./y.\mathsf{server}_x(-_0) \parallel \mathsf{office}(\mathsf{pc}_x \mid \mathsf{pda}_y \mid -_1) \mid y/z$$

Notice how, in *B*, edges are specified by first linking nodes to the same name, then converting that name to an edge using the closure '/'.

We give dynamics to bigraphs by defining reaction rules. Example:
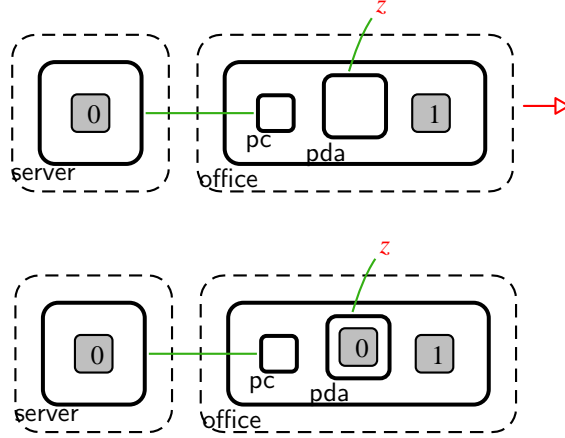
$$/x.\mathsf{server}_x(-_0) \parallel \mathsf{office}(\mathsf{pc}_x \mid \mathsf{pda}_z \mid -_1)$$
$$\longrightarrow /x.\mathsf{server}_x(-_0) \parallel \mathsf{office}(\mathsf{pc}_x \mid \mathsf{pda}_z(-_0) \mid -_1)$$

This rule might model that if a PC in some office is linked to a server, a PDA in the same office may use the PC as a gateway to copy data from the server. The rule matches the bigraph *A* above, taking secret to the site $-_0$ and pda_y to the site $-_1$, rewriting *A* to

$$A' = /x./y.\mathsf{server}_x(\mathsf{secret}) \parallel \mathsf{office}(\mathsf{pc}_x \mid \mathsf{pda}_y(\mathsf{secret}) \mid \mathsf{pda}_y)$$

(We omit details on what it means to match connections; refer to one of [14, 21].)

It is occasionally convenient to limit the contexts in which a reaction rule applies [3], i.e., we might want to limit the above example reaction rule to apply only in the left wing of the building. To this end, bigraphs can be equipped with a *sorting* [12, 21, 20, 17]. A sorting consists of a set of *sorts* (or types); all inner and outer faces are then enriched with such a sort. Further, a sorting must stipulate some condition on bigraphs, we then restrict our attention to the bigraphs that satisfy that condition, thus outlawing some contexts. Obviously, removing contexts may ruin the congruence property of the induced bisimilarity; [12] and [20] give different sufficient conditions for a sorting to preserve that congruence property.

This concludes our informal overview of bigraphs. Now on to the models.

4

3 Naive Models of Location-aware Systems

In this section, we attempt to model location-aware systems naively in bigraphs. We will find the naive approach to be somewhat awkward.

We use the place and link graphs for describing locations and interconnections directly, and we use reaction rules to implement both *reconfiguration* of the context and *queries* on the context. The former is simply a non-deterministic change in the context; the latter is a computation on the context that does not change the context, except for producing an answer to some question. In a location-aware system, a device moving would be a reconfiguration, whereas computing the answer to the question "what devices are currently at the location l" is a query.

We discuss the implementation of this query. (An implementation of the query can be found in Appendix B.) Incidentally, a query such as "find nearest neighbor", which conceptually is only slightly harder, is significantly harder to implement. (Other examples plagued by essentially the same difficulties can be found in [8].)

Consider the following bigraph representing devices (e.g., PDAs) located at locations (e.g., offices, meeting rooms) within a building.

$$l = /w./x./y./z.\mathsf{loc}\,(\mathsf{loc}\,(\mathsf{loc}\,(\mathsf{loc}\,(\mathsf{dev}_w)\mid\mathsf{loc}\,(\mathsf{dev}_x\mid\mathsf{dev}_y)))\mid\mathsf{loc}()\mid\mathsf{loc}\,(\mathsf{dev}_z))$$

Off-hand, finding all devices, say, beneath the root, looks straightforward: We should simply recursively traverse the nesting tree. Unfortunately, such traversal is quite complicated for the following reasons.

- The bigraphical reaction rules do not support recursion directly, so we must encode a runtime stack by means of additional controls.
- Bigraphical reaction rules can be applied in *any* context, but when implementing an operation such as the query we consider now, we need more refined control over when rules can be applied; one may achieve this more refined control by again using additional nodes and controls, essentially implementing what corresponds to a program counter. This still leaves great difficulty in handling concurrent operations, though.
- As a special case of the previous item, it is particularly difficult to express that a reaction rule is intended to apply only in case something is *not* present in the context.

Summing up, the bigraphical rules that model physical action do not in general provide the power to compute directly with a model of that action (because of a lack of control structures). The slogan is "reconfiguring is easy, querying is hard".

In earlier work on evaluating bigraphs as a meta-theory (aim (2) mentioned in the Introduction), reaction rules were used to encode the operational semantics of a calculus or programming language. However, above we attempt to implement a query *directly* as reaction rules. This seemingly innocuous difference will turn out to have major implications for reasoning methods; more on this in Section 6.

We imagine that adding more flexibility to the reaction rules might make it easier to program directly with bigraphs. One possible attempt is to use spatial logics for

bigraphs [5] in combination with sorting, to get control of the contexts in which a particular reaction rule applies.

In the following sections, we propose another way to model context-aware systems in bigraphs, where the reaction rules are not used to program directly with but instead they are used (1) to represent transitions happening in the real world and (2) to encode operational semantics of programming languages, within which one can then implement queries on representations of the real world.

4 Plato-graphical Models of Context-aware Systems

The naive model of the previous section shares an important characteristic with recent proposals of formal models for context-aware computation [3, 7, 25] that comprise agents and contexts only: These models take the agent's ability to determine *what is* the present context as given. We contend that for some systems, it is natural to model not only the actual context but also the agent's representation of the actual context. We shall see that pursuing this idea will partially alleviate the awkwardness seen in the previous Section.

We shall need some notation and definitions.

Notation 1. We write $\mathbf{B} = (\mathcal{K}, \mathcal{R})$ to indicate that $\mathbf{B}$ is a bigraphical reactive system with controls $\mathcal{K}$ and rules $\mathcal{R}$, and write $f \in \mathbf{B}$ to mean that f is bigraph of $\mathbf{B}$.

Definition 1 (Independence). Let $\mathbf{B} = (\mathcal{K}, \mathcal{R})$ and $\mathbf{B}' = (\mathcal{K}', \mathcal{R}')$ be bigraphical reactive systems. Say that $\mathbf{B}$ and $\mathbf{B}'$ are *independent* and write $\mathbf{B} \perp \mathbf{B}'$ iff $\mathcal{K}$ and $\mathcal{K}'$ are disjoint.

Definition 2 (Composite bigraphical reactive systems). Let $\mathbf{B} = (\mathcal{K}, \mathcal{R})$ and $\mathbf{B}' = (\mathcal{K}', \mathcal{R}')$ be bigraphical reactive systems. Define the union $\mathbf{B} \cup \mathbf{B}'$ point-wise, i.e., $\mathbf{B} \cup \mathbf{B}' = (\mathcal{K} \cup \mathcal{K}', \mathcal{R} \cup \mathcal{R}')$, when $\mathcal{K}$ and $\mathcal{K}'$ agree on the arities of the controls in $\mathcal{K} \cap \mathcal{K}'$.

We propose a model of context-aware computing that comprises three bigraphical reactive systems: the context $\mathbf{C}$; its representation or proxy $\mathbf{P}$; and the computational agents $\mathbf{A}$. Drawing on classical work [23] we call such a model *Plato-graphical*.

Definition 3 (Plato-graphical model). A *Plato-graphical model* is a triple $(\mathbf{C}, \mathbf{P}, \mathbf{A})$ of bigraphical reactive systems, such that $\mathcal{M} = \mathbf{C} \cup \mathbf{P} \cup \mathbf{A}$ is itself a bigraphical reactive system and $\mathbf{C} \perp \mathbf{A}$. A *state* of the model is a bigraph of $\mathcal{M}$ on the form $/\vec{x}.(C \parallel P \parallel A)$, where $C \in \mathbf{C}$, $P \in \mathbf{P}$, $A \in \mathbf{A}$, and $\vec{x}$ is some vector of names.

We emphasize the intended difference between $\mathbf{C}$ and $\mathbf{P}$: Whereas an element of $\mathbf{C}$ models a possible context, an element of $\mathbf{P}$ models *a model* of a possible context. The independence condition ensures that agents can only directly observe or manipulate the proxy; not the context itself. (In the parlance of [25], the independence condition ensures separability.) To query or alter the context, agents must use the proxy as a sensor and actuator.

Using bigraphs as our basic formalism gives us two things. First, we can write our own reaction rules. We claim that because of this ability, models become remarkably straightforward and intuitive; hopefully, the reader will agree after seeing our example models in the next section. Second, we automatically get a bisimilarity that is a congruence. Thus, bisimilarity of agents is a very fine equivalence: No state of the context and proxy can distinguish bisimilar agents.

Proposition 1. *Let $\sim$ denote the bisimilarity in $\mathcal{M}$, and let $A, A' \in \mathbf{A}$ with $A \sim A'$. For any $C \in \mathbf{C}$, $P \in \mathbf{P}$, and $\vec{x}$, we have $/\vec{x}.(C \parallel P \parallel A) \sim /\vec{x}.(C \parallel P \parallel A')$.*

To get a less discriminating equivalence we can consider agents under a particular state of the context, or a particular state of the system.

Definition 4. Let $\sim$ denote the bisimilarity in $\mathcal{M}$, and let $A, A' \in \mathbf{A}$, $C \in \mathbf{C}$ and $P \in \mathbf{P}$. We say A and A' are *equivalent w.r.t. P* iff $P \parallel A \sim P \parallel A'$, and we say A and A' are equivalent w.r.t. C, P iff $C \parallel P \parallel A \sim C \parallel P \parallel A'$.

We conjecture that the above forms of derived equivalences will prove useful.

Working within the Plato-graphical model, we are free to emphasize any of its three components, perhaps modeling $\mathbf{P}$ in great detail, but keeping $\mathbf{C}$ and $\mathbf{A}$ abstract.

Definition 3 above does not impose any restriction on composition of states. For example, assume that we have a Plato-graphical model $\mathcal{M} = (\mathbf{C}, \mathbf{P}, \mathbf{A})$, that c, p and a are controls of $\mathbf{C}$, $\mathbf{P}$ and $\mathbf{A}$, respectively, and that p is *not* a control of $\mathbf{C}$. Then the bigraphs

$$F = \mathsf{c}(-_0 \mid -_1) \parallel \mathsf{p} \parallel \mathsf{a}(-_2) \qquad \text{and} \qquad G = \mathsf{c} \parallel \mathsf{p} \parallel \mathsf{a}$$

are both states of $\mathcal{M}$, but their composite $F \circ G = \mathsf{c}(\mathsf{c} \mid \mathsf{p}) \parallel \mathsf{p} \parallel \mathsf{a}(\mathsf{a})$ is not a state of $\mathcal{M}$. This example implies that bisimilarity of states of a Plato-graphical system may be too fine a relation: Conceivably, when comparing two states s and s', we may wish to take into account only contexts C such that $C \circ s$ and $C \circ s'$ are themselves states, i.e., we might want to outlaw F as a possible context for G. We can achieve this finer control using place-sorting. So, we define a place-sorted Plato-graphical model.

Notation 2. For the interface $\langle m, X, sort_m \rangle$, write $\langle m, sort_m \rangle$ when we do not care about names. Represent $\langle m, sort_m \rangle$ by a vector $m_0, \ldots, m_{n-1}$ of sorts. Denote by $S_{i \leq m}$ such a vector representing the sorted interface $\langle m, X, sort_m \rangle$.

Definition 5 (Sorted Plato-graphical model). Let $\mathcal{M} = \mathbf{C} \cup \mathbf{P} \cup \mathbf{A}$ be a Plato-graphical model with $\mathbf{C} = (\mathcal{K}_{\mathbf{C}}, \mathcal{R}_{\mathbf{C}})$, $\mathbf{P} = (\mathcal{K}_{\mathbf{P}}, \mathcal{R}_{\mathbf{P}})$ and $\mathbf{A} = (\mathcal{K}_{\mathbf{A}}, \mathcal{R}_{\mathbf{A}})$. Define a sorting discipline on $\mathcal{M}$ by taking sorts $\Theta = \{\mathcal{K}_{\mathbf{C}}, \mathcal{K}_{\mathbf{P}}, \mathcal{K}_{\mathbf{A}}\}$ and, for primes, sorting condition

$$\Phi(f : S_{i \leq n} \to S) = ctrl(f) \subseteq S \wedge \forall i \leq n.\ S_i = S \,,$$

lifting to an arbitrary bigraph f' by decomposing f into primes $f' = f_0 \ldots f_{n-1}$ and declaring f' well-sorted iff all the f_i are. Let ϕ be an assignment of Θ-sorts to the rules of $\mathcal{R}_{\mathbf{C}}$, $\mathcal{R}_{\mathbf{P}}$, and $\mathcal{R}_{\mathbf{A}}$, such that every rule is well-sorted under Φ. Define $\mathcal{M}'$ to be $\mathcal{M}$ sorted by (Θ, Φ) (using ϕ to lift the reaction rules). In this case, we call $\mathcal{M}'$ a *sorted Plato-graphical model*, and define the states of $\mathcal{M}'$ to be the well-sorted bigraphs with outer face $\mathcal{K}_{\mathbf{C}}, \mathcal{K}_{\mathbf{P}}, \mathcal{K}_{\mathbf{A}}$.

The condition Φ essentially requires that (1) the controls of a prime (bigraph) are elements of the sort of its outer face, and (2) the sort of the outer face is exactly the sort of each of the sites. Under this sorting discipline and new definition of state, if G is assigned a sort such that it is a state, then F cannot be assigned a sort that makes it composable with G.

Is the bisimilarity in the sorted system $\mathcal{M}'$ a congruence? The sorting discipline of $\mathcal{M}'$ is in general not homomorphic in the sense of Milner [20, Definition 10.7]: we cannot give a sort to controls in $\mathcal{K}_{\mathbf{C}} \cap \mathcal{K}_{\mathbf{P}}$. (If $\mathbf{C}$, $\mathbf{P}$ and $\mathbf{A}$ are pairwise independent, the sorting is homomorphic; however, such a model is pathologic.) Neither is the sorting safe in the sense of Jensen [12, Definition 4.30]; condition (4) cannot be met. Counterexample: Suppose $f : \mathcal{K}_{\mathbf{C}} \to \mathcal{K}_{\mathbf{C}}$ is well-sorted; take $g = f \otimes 1 : \mathcal{K}_{\mathbf{C}} \to \mathcal{K}_{\mathbf{C}}, \mathcal{K}_{\mathbf{A}}$ (recall that $1 : \varepsilon \to \langle 1, \emptyset \rangle$ denotes the barren root). Clearly, $\mathcal{U}(f) = (-_0 \mid -_1) \circ \mathcal{U}(f \otimes 1)$. However, if $\mathcal{K}_{\mathbf{C}} \neq \mathcal{K}_{\mathbf{A}}$ then $(-_0 \mid -_1) : \mathcal{K}_{\mathbf{C}}, \mathcal{K}_{\mathbf{A}} \to \mathcal{K}_{\mathbf{C}}$ is not well-sorted.

Nevertheless, the sorting of Definition 5 does give rise to a bisimilarity that is a congruence; we prove so in Appendix C.

5 Examples

We give two examples of Plato-graphical models.

5.1 A Simple Context-aware Printing System

We model the simple context-aware printing system of [3]. An office-building contains both modern PCL-5e compatible printers and old-fashioned raw-printers. Occasionally, the IT-staff at the building removes or replaces either type of printers. Each printer can process only one job; queueing is done by a central print server. The print server dispatches jobs to raw-printers only if it knows no PCL-printers; if there are PCL-printers, but they are all busy, the job will simply have to wait. This system is context-aware: The type and number of printers physically available determine the meaning of the action "to print". We give a model $\mathbf{B}$ of this system in Figure 1. Looking at the controls of $\mathbf{B}$, it is straightforward to verify that $\mathbf{B}$ is Plato-graphical.

Proposition 2. *The model* $\mathbf{B}$ *of Figure 1 is Plato-graphical.*

We take a detailed look at the model. A state of the context $\mathbf{C}$ consists of nested physical locations loc, within which printers prt are placed. We distinguish between PCL- and raw-printers by putting a token pcl and raw within them, respectively. Each printer has a single port, intended to link the printer to the proxy. Here is a state of the context with a PCL-printer and a raw-printer at adjacent locations; the PCL-printer is idle whereas the raw-printer is busy.

$$C = \mathsf{loc}(\mathsf{loc}(\mathsf{prt}_x(\mathsf{raw} \mid \mathsf{dat}_z)) \mid \mathsf{loc}(/y.\mathsf{prt}_y(\mathsf{pcl})))$$

Context **C**.

Control	Activity	Arity	Comment
loc	active	0	Nested location
prt	passive	1	Physical printer
pcl	atomic	0	Printer-type token
raw	atomic	0	Printer-type token
dat	atomic	1	Binary data for printer

$$\mathsf{loc}(-_0) \longrightarrow \mathsf{loc}(-_0 \mid /x.\mathsf{prt}_x(\mathsf{raw})) \tag{1}$$

$$\mathsf{loc}(-_0) \longrightarrow \mathsf{loc}(-_0 \mid /x.\mathsf{prt}_x(\mathsf{pcl})) \tag{2}$$

$$\mathsf{loc}(-_0 \mid \mathsf{prt}_x(-_1)) \longrightarrow \mathsf{loc}(-_0) \mid x/ \tag{3}$$

$$\mathsf{prt}_x(\mathsf{dat}_z \mid -_0) \longrightarrow \mathsf{prt}_x(-_0) \mid z/ \tag{4}$$

Proxy **P**.

Control	Activity	Arity	Comment
prt	passive	1	Physical printer
pcl	atomic	0	Printer-type token
raw	atomic	0	Printer-type token
dat	atomic	1	Binary data for printer
prts	passive	1	Known devices
jobs	passive	0	Pending documents
doc	atomic	1	Document

$$\mathsf{jobs}(\mathsf{doc}_z \mid -_0) \parallel \mathsf{prts}_y(\mathsf{pcl}) \parallel \mathsf{prt}_y(\mathsf{pcl}) \longrightarrow$$
$$\mathsf{jobs}(-_0) \parallel \mathsf{prts}_y(\mathsf{pcl}) \parallel \mathsf{prt}_y(\mathsf{pcl} \mid \mathsf{dat}_z) \tag{5}$$

$$\mathsf{jobs}(\mathsf{doc}_z \mid -_0) \parallel /x.\mathsf{prts}_x(\mathsf{pcl}) \mid \mathsf{prts}_y(\mathsf{raw}) \parallel \mathsf{prt}_y(\mathsf{raw}) \longrightarrow$$
$$\mathsf{jobs}(-_0) \parallel /x.\mathsf{prts}_x(\mathsf{pcl}) \mid \mathsf{prts}_y(\mathsf{raw}) \parallel \mathsf{prt}_y(\mathsf{raw} \mid \mathsf{dat}_z) \tag{6}$$

$$/x.\mathsf{prt}_x(\mathsf{pcl}) \parallel \mathsf{prts}_y(\mathsf{pcl}) \longrightarrow \mathsf{prt}_y(\mathsf{pcl}) \parallel \mathsf{prts}_y(\mathsf{pcl}) \tag{7}$$

$$/x.\mathsf{prt}_x(\mathsf{raw}) \parallel \mathsf{prts}_y(\mathsf{raw}) \longrightarrow \mathsf{prt}_y(\mathsf{raw}) \parallel \mathsf{prts}_y(\mathsf{raw}) \tag{8}$$

Agents **A**.

Control	Activity	Arity	Comment
jobs	passive	0	Pending documents
doc	atomic	1	Document

$$\mathsf{jobs}(-_0) \longrightarrow \mathsf{jobs}(-_0 \mid /z.\mathsf{doc}_z) \tag{9}$$

Fig. 1. Example Plato-graphical model **B**.

Context **C**	Proxy **P**	Agent **A**
$(1): \mathcal{K}_\mathbf{C}$	$(5): \mathcal{K}_\mathbf{A}, \mathcal{K}_\mathbf{P}, \mathcal{K}_\mathbf{C}$	$(9): \mathcal{K}_\mathbf{A}$
$(2): \mathcal{K}_\mathbf{C}$	$(6): \mathcal{K}_\mathbf{A}, \mathcal{K}_\mathbf{P}, \mathcal{K}_\mathbf{C}$	
$(3): \mathcal{K}_\mathbf{C}$	$(7): \mathcal{K}_\mathbf{P}, \mathcal{K}_\mathbf{C}$	
$(4): \mathcal{K}_\mathbf{C}$	$(8): \mathcal{K}_\mathbf{P}, \mathcal{K}_\mathbf{C}$	

Fig. 2. Sorts for the rules of **C**, **P**, and **A**.

Setting C in parallel with some proxy P will allow P access to the raw printer through the shared link x, but not to the PCL-printer, because it is in a closed link. The dynamics of $\mathbf{C}$ allow printers to appear (1, 2), disappear (3), and finish printing (4).

A state of the proxy $\mathbf{P}$ consists of a pool of pending jobs jobs and two tables of printers prts; one contains a token raw, the other a token pcl, indicating what type of printer the table lists. The prts is a table in the sense that its only port is linked to all the printers in the context that the table knows about. Here is an example state of the proxy which knows one raw-printer, knows no PCL-printers and has two pending jobs.

$$P = \mathsf{prts}_x(\mathsf{raw}) \mid /y.\mathsf{prts}_y(\mathsf{pcl}) \mid \mathsf{jobs}(/z.\mathsf{doc}_z \mid /z'.\mathsf{doc}_{z'})$$

Setting C and P above in parallel by $\parallel$, and closing the link x, we get a system $/x.C \parallel P$, where the table $\mathsf{prts}_x(\mathsf{raw})$ and the physical printer $\mathsf{prt}_x(\mathsf{raw} \mid \mathsf{dat})$ are linked. The dynamics of $\mathbf{P}$ state that if there is a job and a known, idle PCL-printer, the proxy may activate this printer (5); that if there is a job, no known PCL-printer, and an idle raw-printer, the context may activate that printer (6); and finally, that the proxy may discover a previously unknown printer (7, 8).

The dynamics of $\mathbf{A}$ allow the agents to spontaneously spool documents (9).

Notice how the two printing rules (5) and (6) do not observe the context directly. Instead, the proxy observes the context (rules (7) and (8)) and records its observations in the tables $\mathsf{prts}_x(\mathsf{raw})$ and $\mathsf{prts}_y(\mathsf{pcl})$; the printing rules (5) and (6) then consults the tables. It is straightforward to determine whether there are no known PCL-printers: simply check if the table of PCL-printers has the form $/y.\mathsf{prts}_y(\mathsf{pcl})$.

As observed in Section 3 and [3], it is generally very difficult, if not impossible, to observe the *absence* of something in the context directly. An interesting but rather natural consequence of the indirect observation is that it becomes asynchronous, i.e., it is possible that a PCL-printer exists but has not yet been observed.

This model $\mathbf{B}$ can be lifted to a sorted one by adding the sorts given in Figure 2; the figure assigns sorts to the outer face of both the redexes and reactums of the indicated rules. It is straightforward to verify that all of the rules are well-sorted.

Proposition 3. *The model* $\mathbf{B}$ *with the sorting assignment of Figure 2 is a sorted Plato-graphical model.*

5.2 A Location-aware Printing System

Suppose we extend the printing system with location-awareness, by stipulating that a print job is not printed until the printer and the device submitting the job are co-located. To model this extended system, we introduce a new control dev for devices (PCs or PDAs) with one port and change doc to include an extra port so we can link submitted jobs to the devices submitting them. The linking is reflected in the following modified rule (9) for spooling print jobs:

$$\mathsf{loc}(\mathsf{dev}_x \mid -_0) \parallel \mathsf{jobs}(-_1) \longrightarrow \mathsf{loc}(\mathsf{dev}_x \mid -_0) \parallel \mathsf{jobs}(-_1 \mid /z.\mathsf{doc}_{z,x}) \qquad (9')$$

We must also modify rules (5) and (6) to insist that the device and printer are co-located. Rule (5) becomes

$$\mathsf{jobs}(\mathsf{doc}_{z,x} \mid -_0) \parallel \mathsf{prts}_y(\mathsf{pcl}) \parallel \mathsf{loc}(\mathsf{dev}_x \mid \mathsf{prt}_y(\mathsf{pcl})) \longrightarrow$$
$$\mathsf{jobs}(-_0) \parallel \mathsf{prts}_y(\mathsf{pcl}) \parallel \mathsf{loc}(\mathsf{dev}_x \mid \mathsf{prt}_y(\mathsf{pcl} \mid \mathsf{dat}_z)). \tag{$5'$}$$

(We suppress the new Rule (6').)

Modifying the system once again, instead of insisting that device and printer have to be actually co-located, we just require the print job to end at a printer close to the device. The print server will need to query the proxy for the printer nearest a given device. We saw in Section 3 that implementing such queries is awkward, so we will need to use the proxy. In the preceding Section, we did so directly in bigraphs; this time around, we transfer the expressive convenience of a general-purpose programming language to bigraphs for ease of implementation. We use bigraphs directly for modeling the actual context **C**, whereas we will exploit bigraphs as a meta-calculus for modeling the proxy **P**.

In detail, the whole model is $\mathbf{B} = \mathbf{C} \cup \mathbf{P} \cup \mathbf{A}$, with $\mathbf{P} = \mathbf{S} \cup \mathbf{L}$. Here **C** is intended to be a bigraphical model of the "real world", the proxy **P** is comprised of a location sensor **S** and a location model **L** and **A** is the location-based application (the "computational agent").

A state C of **C** could look like this:

$$C = \mathsf{loc}(\mathsf{loc}(\mathsf{loc}(\mathsf{loc}(\mathsf{dev}_w) \mid \mathsf{loc}(\mathsf{dev}_x \mid \mathsf{dev}_y))) \mid \mathsf{loc} \mid \mathsf{loc}(\mathsf{dev}_z))$$

Changes in the real world are modeled by reaction rules that reconfigure such states. If we want to model, say, that a devices may move from one location to another, we include the reaction rule

$$\mathsf{loc}(\mathsf{dev}_x \mid -_0) \parallel \mathsf{loc}(-_1) \longrightarrow \mathsf{loc}(-_0) \parallel \mathsf{loc}(\mathsf{dev}_x \mid -_1). \tag{10}$$

To implement the proxy, encode as a BRS a programming language $\mathcal{L}$ with data structures, communication primitives, and concurrency, e.g., Pict [22] or CML [24]. (We return to this assumption below.) That is, define a translation from terms of $\mathcal{L}$ to bigraphs, and add reaction rules encoding the operational semantics of $\mathcal{L}$. *Then implement the location model, the sensor, and the agents in $\mathcal{L}$ and use the encoding to transfer that model to bigraphs*. In particular, a state of the location model **L** will have a data structure *representing* the current state of **C**. If $\mathcal{L}$ is an even half-way decent programming language, it should be straightforward to implement queries such as one of Section 3 or the "find closest printer" we need above.

The sensor informs the location model about changes in **C**. We extend the above rule (10) moving a device to

$$(\mathsf{loc}(\mathsf{dev}_x \mid -_0) \parallel \mathsf{loc}(-_1)) \mid S \mid L \longrightarrow (\mathsf{loc}(-_0) \parallel \mathsf{loc}(\mathsf{dev}_x \mid -_1)) \mid S' \mid L, \tag{$10'$}$$

where S' is an $\mathcal{L}$-encoding of "send a notification to **L** that device x has moved". Upon receiving the notification, **L** updates its representation of the world. Agents of **A** can in turn query **L** when they need location information.

6 Discussion

We consider the following questions.

1. What languages $\mathcal{L}$ can we encode?
2. How closely may Plato-graphical model real systems?
3. What challenges have we found for bigraphical models?
4. What uses do we envision for Plato-graphical models?
5. How do we reason about Plato-graphical models?

Ad. 1. As mentioned, there exist bigraphical encodings of various π-calculi [12, 14, 13] and of the λ-calculus [18, 19]. Using ideas of the latter encodings, we have encoded Mini-ML (call-by-value λ-calculus with pairs and lists) in local bigraphs [18]. Based on our experiences with this encoding, we find it palatable to encode CML or Pict[1].

Ad. 2. The model closely reflects how some actual location-aware systems work, for instance the one running at the ITU. Here, a sensor system (made by Ekahau) computes every two seconds the physical location of every device on the WLAN. The sensor system informs a location model about updates to locations; location-aware services then interact with the location model. In our sketched plato-graphical model, the location model **L** may lag behind the actual **C**, if **L**'s representation of **C** does not reflect some recent reconfiguration of **C**. But that also happens in the real system at the ITU – when a location-aware service asks the location model for the whereabouts of a device, it obtains not the position of the device, but the position of the device the last time the sensor checked. In the mean time, the device may have moved.

Ad. 3. When modeling the physical world, we have made use of both the place and link graphs, the place graph modeling the location hierarchy of a building. As argued in [2], DAGs or graphs are more natural models of location. Thus, systems such as the ones we have considered here suggest generalizing the place graphs part of bigraphs, or consider ways to encode DAGs or general graphs naturally as place graphs.

Ad. 4. Given an implementation of bigraphical reactive systems, one could *simulate* the behaviour of a location-aware system, and thus allow for experimentation with different designs of location-aware and context-aware systems. Likewise, one could experiment with different choices for the $\mathcal{L}$ language of Section 5.2. Such simulation suggests further extensions of the bigraphical model: In actual context-aware systems, one is generally interested in timing aspects (e.g., the sensor samples only every two seconds), continuous space (e.g., the sensor really produces continuous data), and probabilistic models (e.g., to accurately simulate sensors and sensor failure).

---

[1] We are presently working on implementing an interpreter for bigraphical reactive systems; such an interpreter will make it easier to experiment with these and other encodings.

Ad. 5. What about using Plato-graphical models for *formal reasoning* about context-aware systems? One use of formal models is to prove an abstract specification model equivalent to a concrete implementation model. In π-calculus, we come with π-terms i, s, one for the implementation and one for the specification. The terms i and s are themselves the models; we take (π-) bisimilarity as equivalence, so to prove i and s equivalent, we merely prove them bisimilar. We can play the same game within any BRS: Simply come up with a bigraph I (the implementation model) and a bigraph S (the specification model), and prove them bisimilar within the labelled transition system of the BRS. Because that bisimulation is a congruence, such reasoning should be tractable.

Unfortunately, bisimulation within a single BRS is the wrong equivalence for plato-graphical models. Suppose we want a specification model $\mathcal{M}$ with an abstract view of the context, and an implementation model $\mathcal{M}'$ with a detailed view of the context. We express this by having $\mathcal{M}$ and $\mathcal{M}'$ differ only in their context sub-BRSs, that is,

$$\mathcal{M} = \mathbf{C} \cup \mathbf{P} \cup \mathbf{A} \qquad \mathcal{M}' = \mathbf{C}' \cup \mathbf{P} \cup \mathbf{A}.$$

The trouble is that because $\mathbf{C}$ and $\mathbf{C}'$ may have different controls and reaction rules, bisimulation between their respective labelled transition systems is meaningless! What we need is a notion of equivalence of BRSs, not just equivalence of bigraphs of a single BRS. At the time of writing, we know of no such equivalence[2]. Thus, our investigation of bigraphical models for context-aware systems suggests that equivalence of BRSs is a key notion currently missing. One possible direction would be to try recover from the notion of WRS-functor [15] — functors that preserve reaction rules — a notion of a BRS implementing another BRS.

7   Conclusion & Future Work

We have initiated an evaluation of the use of bigraphical reactive systems for models of context-aware computing in ubiquitous systems. We found that BRSs, in their current form, are not suitable for directly modeling context queries, but on the other hand suitable for modeling reconfigurations of the actual context.

In response, we proposed Plato-graphical models, where both state and dynamics are logically divided in three parts: the actual context, the observed context (or proxy), and the computational agents, respectively. The computational agents and the actual context are separated, and interact only through the proxy. This separation into different BRSs makes it possible to encode different parts of the system using domain-specific languages. Moreover, we have shown how the context-aware printing system of [3] can be modeled straightforwardly in the Plato-graphical model.

---

[2] The reader may suggest that we just define a common language for modeling both the abstract and detailed view, and define a translation from this language into a single BRS. However, in this case we are no longer modeling a ubiquitous system directly in bigraphs (aim 1 of the Introduction), but using bigraphs as a meta-calculus (aim 2 of the Introduction).

Further, we have argued that Plato-graphical models are useful for simulating context-aware systems, and we are currently working on an implementation of BRSs at ITU to allow such experimentation. Only through such experimentation will it be clear how useful Plato-graphical models really are. For simulation purposes it will be important to extend bigraphs with timing aspects, continuous space, and probabilities.

Finally, we have pointed out that establishing a notion of equivalence between BRSs, as opposed to bisimilarity within a BRS, is important future work.

8   Acknowledgments

References

1. Mike Addlesee, Rupert W. Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, 2001.

2. Christian Becker and Frank Dürr. On location models for ubiquitous computing. *Personal and Ubiquitous Computing*, 9:20–31, 2005. Springer.

3. Pietro Braione and Gian Pietro Picco. On calculi for context-aware coordination. In *Proceedings of COORDINATION'04*, volume 2949 of *LNCS*, pages 38–54. Springer, 2004.

4. Mikkel Bundgaard and Thomas Hildebrandt. Bigraphical semantics of higher-order mobile embedded resources with local names. In *Proceedings of GT-VC'05*, 2005. Accepted for publication.

5. Giovanni Conforti, Damiano Macedonio, and Vladimiro Sassone. Spatial Logics for Bigraphs. In *Proceedings of ICALP'05*, volume 3580 of *LNCS*, pages 766–778. Springer, 2005. ISBN 3-540-27580-0.

6. Troels C. Damgaard and Lars Birkedal. Axiomatizing binding bigraphs (revised). Technical Report TR-2005-71, IT University of Copenhagen, 2005.

7. Rocco De Nicola, Daniele Gorla, and Rosario Pugliese. Basic observables for a calculus for global computing. In *Proceedings of ICALP'05*, volume 3580 of *LNCS*, pages 1226–1238. Springer, 2005.

8. Søren Debois and Troels C. Damgaard. Bigraphs by Example. Technical Report TR-2005-61, IT University of Copenhagen, March 2005.

9. Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *Workshop on The What, Who, Where, When, and How of Context-Awareness*, 2000. Part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000).

10. Matthew Hennessy. Context-awareness: Models and analysis. Talk at 2nd UK-UbiNet Workshop, slides at `www.cogs.susx.ac.uk/users/matthewh/talks.html`, May 2004.

11. Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *Proceedings of Pervasive'02*, volume 2414 of *LNCS*, pages 167–180. Springer, 2002.

12. Ole Høgh Jensen. *Mobile Processes in Bigraphs*. PhD thesis, University of Aalborg, 2005. Forthcoming.

13. Ole Høgh Jensen and Robin Milner. Bigraphs and Transitions. In *Proceedings of POPL'03*, pages 38–49. ACM Press, 2003. ISBN 1-58113-628-5.

14. Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, University of Cambridge – Computer Laboratory, February 2004. ISSN 1476-2986.

15. James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *Proceedings of CONCUR'00*, pages 243–258. Springer, 2000.

16. Robin Milner. Axioms for bigraphical structure. Technical Report UCAM-CL-TR-581, University of Cambridge – Computer Laboratory, February 2004. ISSN 1476-2986.

17. Robin Milner. Bigraphs for Petri Nets. In *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, volume 3098 of *LNCS*, pages 686–701. Springer, 2004.

18. Robin Milner. Bigraphs whose names have multiple locality. Technical Report UCAM-CL-TR-603, University of Cambridge – Computer Laboratory, September 2004. ISSN 1476-2986.

19. Robin Milner. Bigraphs: A tutorial. Slides, April 2005. Available at `http://www.cl.cam.ac.uk/users/rm135/bigraphs-tutorial.pdf`.

20. Robin Milner. Pure bigraphs. Technical Report UCAM-CL-TR-614, University of Cambridge – Computer Laboratory, January 2005. ISSN 1476-2986.

21. Robin Milner. Pure bigraphs: Structure and dynamics. *Information and Computation*, 2005. To appear.

22. Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*, pages 455–494. MIT Press, 2000.

23. Plato. The republic, book vii, 360 B.C. Translation by Benjamin Jowett.

24. John H. Reppy. *Concurrent Programming in ML*. Cambridge University Press, 1999.

25. Gruia-Catalin Roman, Christine Julien, and Jamie Payton. A formal treatment of context-awareness. In *Proceedings of FASE'04*, volume 2984 of *LNCS*, pages 12–36, 2004.

26. Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994.

27. Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to context than location. *Computers & Graphics Journal*, 23(6):893–902, December 1999.

28. Mark Weiser. Hot topics – ubiquitous computing. *IEEE Computer*, 26(10):71–72, October 1993.

A Bigraphs

We recite the identical relevant definitions of [14] and a few from [21].

Definition 6 (pure signature). A *(pure) signature* $\mathcal{K}$ is a set whose elements are called *controls*. For each control K it provides a finite ordinal $ar(K)$, an *arity*; it also determines which controls are *atomic*, and which of the non-atomic controls are *active*. Controls which are not active (including the atomic controls) are called *passive*.

Definition 7 (prime interface). An *interface* $I = \langle m, X \rangle$ consists of a finite ordinal m called a *width*, a finite set X called a *name set*. An interface is *prime* if it has width 1.

Definition 8 (prime bigraph). A *prime* bigraph $P : m \to \langle X \rangle$ has no inner names and a prime outer face.

Definition 9 (place graph). A *place graph* $A = (V, ctrl, prnt) : m \to n$ has an *inner width m* and an *outer width n*, both finite ordinals; a finite set V of nodes with a control map $ctrl : V \to \mathcal{K}$; and a *parent map prnt* $: m \uplus V \to V \uplus n$. The parent map is *acyclic*, i.e. $prnt^k(v) \neq v$ for all $k > 0$ and $v \in V$. An *atomic* node – i.e. one whose control is atomic – may not be a parent. We write $w >_A w'$, or just $w > w'$, to mean $w = prnt^k(w')$ for some $k > 0$.

The widths m and n index the *sites* and *roots* of A respectively. The sites and nodes – i.e. the domain of *prnt* – are called *places*.

Definition 10 (precategory of place graphs). The precategory of place graphs $´\text{PLG}$ has finite ordinals as objects and place graphs as arrows. The composition $A_1 \circ A_0 : m_0 \to m_2$ of two place graphs $A_i = (V_i, ctrl_i, prnt_i) : m_i \to m_{i+1}$ $(i = 0, 1)$ is defined when the two node sets are disjoint; then $A_1 \circ A_0 \stackrel{\text{def}}{=} (V, ctrl, prnt)$ where $V = V_0 \uplus V_1$, $ctrl = ctrl_0 \uplus ctrl_1$, and $prnt = (\mathsf{Id}_{V_0} \uplus prnt_1) \circ (prnt_0 \uplus \mathsf{Id}_{V_1})$. The identity place graph at m is $\mathrm{id}_m \stackrel{\text{def}}{=} (\emptyset, \emptyset_{\mathcal{K}}, \mathsf{Id}_m) : m \to m$.

Definition 11 (tensor product, $´\text{PLG}$). The *tensor product* $\otimes$ in $´\text{PLG}$ is defined as follows: On objects, we take $m \otimes n = m + n$. For two place graphs $A_i : m_i \to n_i$ $(i = 0, 1)$ we take $A_0 \otimes A_1 : m_0 + m_1 \to n_0 + n_1$ to be defined when A_0 and A_1 have disjoint node sets; for the parent map, we first adjust the sites and roots of A_1 by adding them to m_0 and n_0 respectively, then take the union of the two parent maps.

Definition 12 (barren,sibling,active,passive). A node or root is *barren* it is has no children. Two places are *siblings* if they have the same parent. A site s of A is *active* if $ctrl(v)$ is active whenever $v > s$; otherwise s is *passive*. If s is active (resp. passive) in A, we also say that A is *active* (resp. *passive*) *at s*.

Definition 13 (hard place graphs). A *hard* place graph is one in which no root or non-atomic node is barren. They form a sub-precategory denoted by $´\text{PLG}_h$.

Presuppose a denumerable set χ of global names.

Definition 14 (link graph). A *link graph* $A = (V, E, ctrl, link) : X \to Y$ has finite sets X of *inner names*, Y of *(outer) names*, V of *nodes* and E of *edges*. It also has a function $ctrl : V \to \mathcal{K}$ called the *control map*, and a function $link : X \uplus P \to E \uplus Y$ called the *link map*, where $P \stackrel{\text{def}}{=} \sum_{v \in V} ar(ctrl(v))$ is the set of *ports* of A.

We shall call the inner names X and ports P the *points* of A, and the edges E and outer names Y its *links*.

Definition 15 (precategory of link graphs). The precategory ´LIG has name sets as objects and link graphs as arrows. The composition $A_1 \circ A_0 : X_0 \to X_2$ of two link graphs $A_i = (V_i, E_i, ctrl_i, link_i) : X_i \to X_{i+1}$ $(i = 0, 1)$ is defined when their node sets and edge sets are disjoint; then $A_1 \circ A_0 \stackrel{\text{def}}{=} (V, E, ctrl, link)$ where $V = V_0 \uplus V_1, ctrl = ctrl_0 \uplus ctrl_1, E = E_0 \uplus E_1$ and $link = (\text{Id}_{E_0} \uplus link_1) \circ (link_0 \uplus \text{Id}_{P_1})$. The identity link graph at X is $\text{id}_X = (\emptyset, \emptyset, \emptyset_\mathcal{K}, \text{Id}_X) : X \to X$.

Definition 16 (tensor product, ´LIG). The *tensor product* $\otimes$ in ´LIG is defined as follows: On objects, $X \otimes Y$ is simply the union of sets required to be disjoint. For two link graphs $A_i : X_i \to Y_i$ $(i = 0, 1)$ we take $A_0 \otimes A_1 : X_0 \otimes X_1 \to Y_0 \otimes Y_1$ to be defined when the interface products are defined and when A_0 and A_1 have disjoint node sets and edge sets; then we take the union of their link maps.

Definition 17 (parallel product). The *parallel product* $\mid$ in ´LIG is defined as follows: On objects, $X \mid Y \stackrel{\text{def}}{=} X \cup Y$. On link graphs $A_i : X_i \to Y_i$ $(i = 0, 1)$ we define $A_0 \mid A_1 : X_0 \otimes X_1 \to Y_0 \mid Y_1$ whenever X_0 and X_1 are disjoint, by taking the union of link maps.

Definition 18 (concrete pure bigraph). A *(concrete) pure bigraph* over the signature $\mathcal{K}$ takes the form $G = (V, E, ctrl, G^P, G^L) : I \to J$ where $I = \langle m, X \rangle$ and $J = \langle n, Y \rangle$ are its *inner* and *outer faces*, each combining a *width* (a finite ordinal) with a finite set of global names drawn from χ. Its first two components V and E are finite sets of *nodes* and *edges* respectively. The third component $ctrl : V \to \mathcal{K}$, a *control map*, assigns a control to each node. The remaining two are: $G^P = (V, ctrl, prnt) : m \to n$, $G^L = (V, E, ctrl, link) : X \to Y$.

A place graph can be combined with a link graph iff they have the same node set and control map.

Definition 19 (Tensor product). The *tensor product* of two bigraph interfaces is defined by $\langle m, X \rangle \otimes \langle n, Y \rangle \stackrel{\text{def}}{=} \langle m + n, X \uplus Y \rangle$ when X and Y are disjoint. The *tensor product* of two bigraphs $G_i : I_i \to J_i (i = 0, 1)$ is defined by $G_0 \otimes G_1 \stackrel{\text{def}}{=} \langle G_0^P \otimes G_1^P, G_0^L \otimes G_1^L \rangle : I_o \otimes I_1 \to J_0 \otimes J_1$ when the interfaces exist and the node sets are disjoint. This combination is well-formed, since its constituents share the same node set.

Definition 20 (precategory of pure concrete bigraphs). The precategory ´BIG$(\mathcal{K})$ of pure concrete bigraphs over a signature $\mathcal{K}$ has pairs $I = \langle m, X \rangle$ as objects (*interfaces*) and bigraphs $G = (V, E, ctrl_G, G^P, G^L) : I \to J$ as arrows (*contexts*). We call I the *inner face* of G, and I the *outer face*. If $H : J \to K$ is another bigraph with node set disjoint from V, then their composition is defined directly in terms of the compositions of the constituents as follows: $H \circ G \stackrel{\text{def}}{=} \langle H^P \circ G^P, H^L \circ G^L \rangle : I \to K$. The identities are $\langle \text{id}_m, \text{id}_X \rangle : I \to I$, where $I = \langle m, X \rangle$.

The subprecategory ´BIG$_h$ consists of *hard* bigraphs, those with place graphs in ´PLG$_h$.

Definition 21 (tensor product, ´BIG). The *tensor product* of two bigraph interfaces is defined by $\langle m, X \rangle \otimes \langle n, Y \rangle \stackrel{\text{def}}{=} \langle m+n, X \cup Y \rangle$ when X and Y are disjoint. The *tensor product* of two bigraphs $G_i : I_i \to J_i$ $(i = 0, 1)$ is defined by $G_0 \otimes G_1 \stackrel{\text{def}}{=} \langle G_0^{\mathsf{P}} \otimes G_1^{\mathsf{P}}, G_0^{\mathsf{L}} \otimes G_1^{\mathsf{L}} \rangle : I_0 \otimes I_1 \to J_0 \parallel J_1$ when the interfaces exist and the node sets are disjoint. This combination is well-formed, since its constituents share the same node set.

Definition 22 (parallel product, ´BIG). The *parallel product* of two bigraphs is defined on interfaces by $\langle m, X \rangle \parallel \langle n, Y \rangle \stackrel{\text{def}}{=} \langle m+n, X \cup Y \rangle$, and on bigraphs by $G_0 \parallel G_1 \stackrel{\text{def}}{=} \langle G_0^P \otimes G_1^P, G_0^L \mid G_1^L \rangle : I_o \otimes I_1 \to J_0 \parallel J_1$ when the interfaces exist and the node sets are disjoint.

Refer to [14] for the definition of ´BBG, ´BBG$_\mathsf{h}$, and $\simeq$.

Definition 23 (bigraphical reactive system). A *bigraphical reactive system (BRS)* over $\mathcal{K}$ consists of ´BBG$(\mathcal{K})$ equipped with a set ´Reacts of reaction rules closed under support equivalence ($\simeq$). We denote it – and similarly for ´BBG$_\mathsf{h}(\mathcal{K})$ – by ´BBG$(\mathcal{K}, \text{´Reacts})$.

Refer to [21] for the definition of *s-category*.

Definition 24 (place-sorted bigraphs). An interface $\langle m, X \rangle$ is Θ-*(place-)sorted* if it is enriched by ascribing a sort to each place $i \in m$. If I is place-sorted we denote its underlying unsorted interface by $\mathcal{U}(I)$.

We denote by ´BIG$_\mathsf{h}(\mathcal{K}, \Theta)$ the s-category in which the objects are place-sorted interfaces, and each arrow $G : I \to J$ is a bigraph $G : \mathcal{U}(I) \to \mathcal{U}(J)$. The identities, composition and tensor product are as in ´BIG$_\mathsf{h}(\mathcal{K})$, but with sorted interfaces.

Definition 25 (place-sorting). A *place-sorting* is a triple $\Sigma = (\mathcal{K}, \Theta, \Phi)$ where Φ is a condition on the place graphs of Θ-sorted bigraphs over $\mathcal{K}$. The condition Φ must be satisfied by the identities and preserved by composition and tensor product.

A bigraph in ´BIG$_\mathsf{h}(\mathcal{K}, \Theta)$ is Σ*-(place)sorted* if it satisfies Φ. The Σ-sorted bigraphs form a sub-s-category of ´BIG$_\mathsf{h}(\mathcal{K}, \Theta)$ denoted by ´BIG$_\mathsf{h}(\Sigma)$. Further, if ´$\mathcal{R}$ is a set of Σ-sorted reaction rules then ´BIG$_\mathsf{h}(\Sigma, \text{´}\mathcal{R})$ is a Σ-sorted BRS.

B Encoding of "find all devices"

Consider the following simple bigraph representing a building consisting of locations (e.g., rooms) and devices (e.g., PDAs) in these locations. (We have omitted the outer names on the locations, and also sites.)

$$l = \mathsf{loc}(\mathsf{loc}(\mathsf{loc}(\mathsf{loc}(\mathsf{dev}_1) \mid \mathsf{loc}(\mathsf{dev}_2 \mid \mathsf{dev}_3))) \mid \mathsf{loc}() \mid \mathsf{loc}(\mathsf{dev}_4))$$

Consider how to implement a query to return all the devices in the building by means of bigraphical reaction rules. Observe that we have chosen to represent all locations via the same control loc, rather than using different controls office, building, etc., for

different locations – this is to avoid having to write reaction rules for every combination of location controls.

Now, assume that a query occurs by some process introducing a node with control f into the system (in a unique[3] in node), and that no other queries impose themselves while we calculate the answer to this one. The termination condition (observable by the "input/output process") is when in is empty (and nodes with control f′ appear in the node with unique control out). We can not handle concurrent queries so that is why we wish to detect termination (so that we can begin the next query).

The idea is to do a depth-first search/collection while keeping track of where we have already looked by placing these subtrees into "searched-nodes" (s). Controls:

Control	Activity	Arity	Comment
in	passive	0	Input node
f	atomic	0	Controls find-all query
f′	atomic	1	Answer node
loc	active	1	Nested location
out	passive	0	Output node
g	atomic	0	Dummy, just to keep in non-empty
dev	atomic	1	Device, has link to id
s	passive	0	Collects searched nodes

And now, for the rules. Initialization; move f into the *top* location (enclosing all the others) to indicate "the point of control" in the structure, and add g to indicate that we are not done with the query:

$$\mathsf{in}(\mathsf{f}) \parallel \mathsf{loc}_{top}(\mathsf{loc}_x(-_0)) \parallel \mathsf{out}() \longrightarrow \mathsf{in}(\mathsf{g}) \parallel \mathsf{loc}_{top}(\mathsf{loc}_x(\mathsf{f} \mid \mathsf{s}() \mid -_0)) \parallel \mathsf{out}()$$

If a device is found, add it to s, and add a representative for it to out:

$$\mathsf{loc}_x(\mathsf{f} \mid -_0 \mid \mathsf{dev}_y \mid \mathsf{s}(-_1)) \parallel \mathsf{out}(-_2)$$
$$\longrightarrow /y.\mathsf{loc}_x(\mathsf{f} \mid -_0 \mid \mathsf{s}(-_1 \mid \mathsf{dev}_y)) \parallel \mathsf{out}(-_2 \mid \mathsf{f}'_y)$$

Notice that the label (context) of this transition will include the bigraph top/x. This rule can be used as long as there are devices in the current location being searched. When done with this location f is moved up, since we assume that a location can only contain either devices or other locations. (The query would have been significantly harder without this assumption.) So, if a location containing location(s) is being searched:

$$\mathsf{loc}_x(\mathsf{f} \mid \mathsf{loc}_y(-_1) \mid \mathsf{s}(-_2) \mid -_0) \longrightarrow \mathsf{loc}_x(\mathsf{loc}_y(\mathsf{f} \mid \mathsf{s}() \mid -_1) \mid \mathsf{s}(-_2) \mid -_0)$$

Then, search deeper. A new s-node is created when going down, this is a trick to do "on-line garbage collection" when climbing back up the tree. (The query has to leave

---

[3] Certain properties can only be ensured by invariants of the reactive system, e.g. uniqueness of controls.

the bigraph as it was initially.) When a leaf is reached (an empty location) move f up, merging s-nodes. This is more clever than doing a clean-up traversal because there is no construction in reaction rules that can express "not", i.e. one can not write a rule saying "clean up until there are no more s-nodes in the tree". The "climbing" rule:

$$\mathsf{loc}_x(\mathsf{loc}_y(\mathsf{f} \mid \mathsf{s}(-_2)) \mid \mathsf{s}(-_1) \mid -_0) \longrightarrow \mathsf{loc}_x(\mathsf{f} \mid \mathsf{s}(\mathsf{loc}_y(-_2) \mid -_1) \mid -_0)$$

Notice how the presented rules use sites to make themselves general. To clean up when we have traversed the whole tree (and there is exactly one s):

$$\mathsf{in}(g) \parallel \mathsf{loc}_{top}(\mathsf{loc}_x(\mathsf{f} \mid \mathsf{s}(-_0))) \longrightarrow \mathsf{in}() \parallel \mathsf{loc}_{top}(\mathsf{loc}_x(-_0))$$

At this point out will have all representatives, in is emptied to indicate termination. (The reader is encouraged to try out the rules on the example location model. It is easiest doing it using the graphical bigraph notation.)

C   Rigid control-sortings and RPOs

For a bigraph b (sorted or otherwise), we write b^* for the function that takes each place (site or root) or node of b to its uniquely determined root. In this appendix we will generally omit writing down the link-graph part of interfaces when we do not need them.

Definition 26 (Rigid control-sorting). Let $\mathcal{K}$ be a set of controls. A sorting $\mathcal{S} = (\mathcal{K}, \Theta, \Phi)$ is a *rigid control-sorting* if $\Theta \subseteq \mathcal{P}(\mathcal{K})$ and there exists a predicate ϕ, such that

$$\Phi\big((m, s_m) \xrightarrow{f} (n, s_n)\big) \quad \text{iff} \quad \begin{cases} (i) & s_m(i) = s_n(f^*(i)) & \text{for } i < m, \\ (ii) & \phi(ctrl_f(v), s_n(f^*(v))) \text{ for } v \text{ node in } f. \end{cases}$$

In the sequel, we assume a fixed set of controls $\mathcal{K}$, rigid control-sorting $\mathcal{S} = (\mathcal{K}, \Theta, \Phi)$, a sorted signature $\Sigma^{\mathcal{S}}$ and a corresponding unsorted signature $\mathcal{U}(\Sigma^{\mathcal{S}}) = \Sigma$; following [12], we write $\mathbf{BIG}(\Sigma)$ for the precategory of concrete bigraph over Σ and $\mathbf{BIG}(\Sigma^{\mathcal{S}})$ for the corresponding precategory of sorted concrete bigraphs, and we write $\mathcal{U}$ for the forgetful functor from $\mathbf{BIG}(\Sigma^{\mathcal{S}})$ to $\mathbf{BIG}(\Sigma)$; recall that this functor is faithful.

In Theorem 1 we state that $\mathbf{BIG}(\Sigma^{\mathcal{S}})$ has RPOs; it follows that the standard bisimulation on $\mathbf{BIG}(\Sigma^{\mathcal{S}})$ is a congruence. To establish Theorem 1, we will need some lemmas to make precise just how closely $\mathbf{BIG}(\Sigma^{\mathcal{S}})$ mimics $\mathbf{BIG}(\Sigma)$.

Lemma 1. *If $\mathcal{U}(a) = p \circ q$, then there exists unique b, c s.t. $\mathcal{U}(b) = p$, $\mathcal{U}(c) = q$ and $a = b \circ c$.*

Proof. For existence, suppose $a : (m, s_m) \longrightarrow (n, s_n)$ and $\mathrm{cod}(q) = \mathrm{dom}(p) = l$. Define

$$s_l(i) \stackrel{\mathrm{def}}{=} s_n(p^*(i)). \tag{11}$$

20

We claim that $c = (m, s_m) \xrightarrow{q} (l, s_l)$ and $b = (l, s_l) \xrightarrow{p} (n, s_n)$ are well-sorted. Consider c. Condition (i) of Definition 26 is satisfied by (11), Condition (ii) is satisfied because the nodes of c is a subset of the nodes of a. Now consider b. For $i < n$, we find

$$s_m(i) = s_n(a^*(i)) = s_n(p^*(q^*(i))) = s_l(q^*(i)),$$

satisfying Condition (i). Next, for v a node of q, we find

$$\phi(ctrl_q(v), s_l(q^*(v))) = \phi(ctrl_a(v), s_n(p^*(q^*(v)))) = \phi(ctrl_a(v), s_n(a^*(v))).$$

But $\phi(ctrl_a(v), s_n(a^*(v)))$ is satisfied by well-sortedness of a; thus Condition (ii) is satisfied.

For uniqueness, it is sufficient to prove that s_l is the only sorting making b and c well-sorted. Suppose s_l' is an alternate such sorting. If there is $i < l$ s.t. $s_l'(i) \neq s_l(i) = s_n(p^*(i))$, then $(l, s_l') \xrightarrow{p} (n, s_n)$ is not well-sorted: contradiction. Thus $s_l' = s_l$.

Lemma 2. *If a, b is a cospan and $\mathcal{U}(a) = \mathcal{U}(b)$, then $a = b$.*

Proof. Because $\mathcal{U}(a) = \mathcal{U}(b)$, a and b must have the same inner width, m:

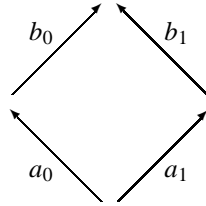$$(m, s_m) \xrightarrow{\quad a \quad} (n, s_n) \xleftarrow{\quad b \quad} (m, s_m').$$

Suppose for a contradiction that there is $i < m$ s.t. $s_m'(i) \neq s_m(i)$. Then

$$s_n(a^*(i)) = s_m(i) \neq s_m'(i) = s_n(b^*(i)),$$
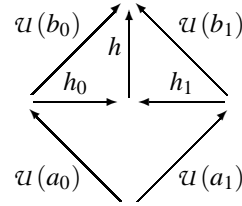
but that cannot be, because $a^*(i) = b^*(i)$ follows from $\mathcal{U}(a) = \mathcal{U}(b)$: contradiction. $\square$

Theorem 1. $\mathbf{BIG}(\Sigma^S)$ *has RPOs.*
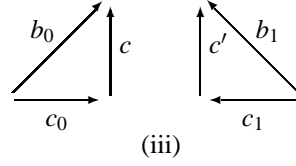
Proof. Consider the square (i) below.

(i) (ii)

Apply $\mathcal{U}$ to get a similar square in $\mathbf{BIG}(\Sigma)$, and erect an RPO there, altogether obtaining the diagram (ii). By Lemma 1, there are c_0 and c factoring b_0 s.t. $\mathcal{U}(c_0) = h_0$ and $\mathcal{U}(c) = h$; symmetrically, there are also c_1 and c' factoring b_1 s.t. $\mathcal{U}(c_1) = h_1$ and $\mathcal{U}(c') = h$. (See diagram (iii) below.)

21

(iii)

But b_0, b_1 is a cospan, so also c, c' is a cospan; thus $c = c'$ by Lemma 2, and we have a candidate RPO c_0, c_1, c.

Suppose d_0, d_1, d is an alternate candidate RPO. We must find unique e s.t. $c = d \circ e$. Because $\mathcal{U}(c_0), \mathcal{U}(c_1), \mathcal{U}(c)$ is an RPO, we find unique p s.t. $\mathcal{U}(c) = \mathcal{U}(d) \circ p$. By Lemma 1, there are unique d', e s.t. $\mathcal{U}(d') = \mathcal{U}(d)$, $\mathcal{U}(e) = p$ and $c = d' \circ e$. But then d, d' is cospan, so by Lemma 2, $d = d'$. Thus, we have found e s.t. $c = d \circ e$. For uniqueness, suppose there is e' with $c = d \circ e'$. Then

$$h = \mathcal{U}(c) = \mathcal{U}(d) \circ \mathcal{U}(e') = \mathcal{U}(d) \circ p$$

but then $\mathcal{U}(e') = p = \mathcal{U}(e)$ by uniqueness of p; but e', e is also a cospan, so by Lemma 2, $e = e'$.

Corollary 1. *Bisimulation on the standard transition-system of $\mathbf{BIG}(\Sigma^S)$ is a congruence.*

Proof. By [12, Theorem 3.16], possession of RPOs is a sufficient prerequisite for the desiderata.

We can now prove that the sorting of Definition 5 gives a congruential bisimulation.

Theorem 2. *Let S be the sorting given in Definition 5. Then the bisimulation over the standard transitions of $\mathbf{BIG}(\Sigma^S)$ is a congruence.*

Proof. By Corollary 1, it is sufficient to show that S is a rigid control sorting. Take $\phi(k, K) = k \in K$. Clearly, $\Phi\big((m, s_m) \xrightarrow{f} (n, s_n)\big)$ is equivalent to $i < m \implies s_m(i) = s_n(f^*(i))$ and $v \in f \implies \phi(ctrl_f(v), s_n(f^*(v)))$.