Sorted Bigraphs*
Draft!

Lars Birkedal Søren Debois Thomas Hildebrandt

April 4, 2007

Abstract

Building upon work of Milner, Leifer, Jensen and ourselves, we develop a theory of *sorted bigraphical reactive systems*. Concretely, we show how to automatically get a sorting on bigraphs sustaining the behavioural theory of pure bigraphical reactive systems by lifting a *safe sorting* on the underlying reactive system of abstract bigraphs. We back this result by a general construction of a safe sorting, the *Closure sorting*. This sorting restricts a reactive system, and thus by the main result any bigraphical reactive system, to the contexts satisfying any given decomposable predicate. We demonstrate the applicability of the Closure sorting by proving that Milner's Local Bigraphs arise as a closure sorting of pure bigraphs, using part of the usual scoping condition as predicate.

1 Introduction

Bigraphical reactive systems is a framework proposed by Milner and others [1, 2, 3, 4, 5] as a unifying theory of process models and as a tool for reasoning about ubiquitous computing. For process models, it has been shown that Petri-nets [6], CCS [1], various π-calculi [5, 2, 7, 8], mobile ambients [7], and Homer [9] can all be understood as bigraphical reactive systems. Moreover, Milner recently used bigraphs as a vehicle for studying confluence, using the λ-calculus as an example [4, 10]. For ubiquitous computing, bigraphical models were investigated in [11, 12].

A bigraphical reactive system consists of a category of bigraphs and a reaction relation on those bigraphs; we can think of the bigraphs as terms modulo structural congruence. The benefit of working within bigraphical reactive systems comes from their rich behavioural theory [1, 5, 3, 2, 7, 13, 14, 15] which (a) derives a labelled transition system automatically from the reaction relation and (b) guarantees that bisimulation on that transition system is a congruence.

A category of bigraphs is formed according to a single-sorted signature, which defines the kinds of nodes found in bigraphs of that category. Unfortunately, single-sorted signatures are usually insufficient when we define programming languages or

*This work is funded in part by the Danish Research Agency (grant no.: 2059-03-0031) and the IT University of Copenhagen (the LaCoMoCo and BPL projects).

algebraic models: We need to constrain the ways operators can be combined, so we need richer notions of sorting. Indeed, *every one* of [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] uses a richer sorting to fit the framework of bigraphs to the problem at hand[1].

Alas, the behavioural theory of bigraphs applies only to single-sorted (or pure) bigraphs, not to arbitrarily-sorted variations. Hence, *also* every one of [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] must re-develop substantial parts of the behavioural theory for the sorting in question. Worse, although some sortings are easy to construct [1, 3, 7], others require either hard work [11] or ingenuity [5, 4, 10] to achieve conceptually simple effects.

Thus, in this paper, we ask and answer the following two questions:

1. Which sortings sustain the behavioural theory of bigraphs?

2. How do we construct such a sorting for a given problem domain?

Very briefly, we answer Question 1 by showing that any sorting for a reactive system gives rise to a sorting for bigraphs; and that previous work on sortings for reactive systems [16, 7] can be lifted to the present setting of bigraphical reactive systems. We answer 2 by giving a new family of sortings, closure sortings, all of which sustain the behavioural theory. Moreover, we prove that Milner's local bigraphs [10, 4] arise as a sorting of pure bigraphs; we conjecture that most of the sortings used in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] are instances of closure sortings.

In more detail: Question 1. Jensen [7] found a sufficient condition, safety, for a small class of sortings for bigraphs to sustain the behavioural theory. In [16] we lifted that condition to general sortings of reactive systems. However, those results do not apply to bigraphs because the behavioural theory of bigraphs is defined not only in terms of bigraphs themselves, but also in terms of so-called concrete bigraphs.

In the present paper, we show that (a) any sorting F of bigraphs gives rise to a sorting $F^\star$ of the corresponding concrete bigraphs (Proposition 24); and (b) if only the sorting F is safe, then $F^\star$ sustains the behavioural theory of bigraphs (Theorem 26 and 28). In combination, these theorems obviate the need to re-develop the behavioural theory of bigraphs for every new sorting.

In more detail: Question 2. A sorting is defined categorically as a functor from a category (the sorted category) to the pure category [16]. This functor is required to be faithful and surjective on objects: Sortings are only concerned with the *interfaces* between bigraphs and sortings *refine* the basic sorting of the pure category.

As part of the safety condition mentioned above, it is required that if a decomposable context has a sorting, then the sorting can be decomposed correspondingly. In particular, the "has a sorting" predicate P on the pure category is decomposable, i.e., $P(f \circ g)$ implies $P(f)$ and $P(g)$. Thinking in terms of sorted algebra or programming languages this is a very natural condition — a refinement of a sorting should not constrain the way a well-sorted term can be decomposed.

[1] Milner and Leifer anticipated as much [3, p.43]: "... sorting ... is likely to be needed in any significant application."

In [16] we discovered that in general, banning a set of "bad" morphisms always gives rise to a decomposable predicate which is true precisely at the "good" morphisms. This insight gave rise to an answer to Question 2, albeit only for reactive systems: We gave, for any predicate P on the morphisms of a category, a sorting called the Predicate sorting. The Predicate sorting sustains the theory of reactive systems. Using the answer to Question 1 given in this paper, the Predicate sorting can be lifted to a sorting on bigraphs.

In practice, however, the Predicate sorting turns out to provide far more sorts for each bigraph than the sortings found in an ad hoc way for the applications in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. In the present paper, we construct a new family of sortings, the Closure sortings. Like the Predicate sortings, Closure sortings sustain the behavioural theory of both bigraphs and reactive systems. Unlike the Predicate sortings, we conjecture that the Closure sorting re-constructs most of the sortings of the applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. As a particularly spectacular example, we show that Milner's Local bigraphs [10, 4] arise as a Closure sorting, by taking Milner's scoping condition as a predicate on bigraphs.

In summary, the present paper answers questions 1 and 2 above, by (a) lifting the answers given for reactive systems in [16] to the setting of bigraphs and (b) improve the automatic construction of sortings to provide, in most cases, exactly the sortings previously derived by hand. In particular, we show that the closure sorting completely removes the mystery of local bigraphs by revealing that they are just a safely sorted variation of bigraphs derived automatically from pure bigraphs. We take this as a strong indication that the the lifting of safe closure sortings not only answer the two questions, but also automatically provide sortings that are as good as what we could have derived by hand via ingenuity and hard work.

Overview. In Section 2 we revisit Milner and Leifers' classic Reactive Systems [13, 14, 17], a precursor to bigraphs. In Section 3, we recall the definition of a sorting of a reactive system. In Section 4, we give a general condition for a sorting of a reactive system to preserve dynamics up to a predicate (Theorem 12). In Section 5, we give the Closure sorting (Definition 13) and thus a preliminary answer for Question 2 above (Theorem 14); the answer is preliminary in that the Closure sorting is defined only for reactive systems, not for the more general *supported* reactive systems. Thus, in Section 7, we lift that answer to the setting of supported reactive systems, of which bigraphs form a special case, thus delivering the promised answer to Question 2 (Theorem 29). In the process, we show that assorted known results about sortings for reactive systems can also be lifted to supported reactive systems — a result anticipated in [16] — thus delivering the answer to Question 1 (Theorems 26 and 28). Finally, in Section 8, we apply our answers to reveal Local bigraphs as an instance of Closure sorted bigraphs (Theorem 31).

This paper is an abridged version of the forthcoming technical report [18]; refer to that report for proofs and details.

2 Reactive Systems

In this section, we recall Milner and Leifer's *Reactive Systems* [17, 14, 13]. These systems form the conceptual basis of bigraphical reactive systems. Except for the running example, this section contains no original work.

Let C be a category, and let ϵ be an object of C. We think of morphisms with domain ϵ as *agents* or *processes* and all other morphisms as *contexts*. A *reaction rule* (l, r) is a span of agents; intuitively, l and r are the left- and right-hand sides of rewrite rule. A set $\mathcal{R}$ of reaction rules induces a *reaction relation*, $\longrightarrow\!\!\!\!\!\rightarrow$, obtained by closing reaction rules under contexts:

$$a \longrightarrow\!\!\!\!\!\rightarrow b \quad \text{iff} \quad \exists f \in C, \exists (l, r) \in \mathcal{R}.\ a = f \circ l, b = f \circ r. \tag{1}$$

Altogether, these components constitute a reactive system.

Definition 1 (Reactive system)**.** A *reactive system* over a category C comprises a distinguished object ϵ and a set $\mathcal{R}$ of reaction rules; the reaction rules gives rise to a reaction relation by (1) above.

Example 2. Here is a small process language.

$$P, Q ::= \mathsf{0} \mid \mathsf{a} \mid \mathsf{s} \mid (P|Q) \tag{2}$$

This language has just the nil process $\mathsf{0}$, atomic processes a and s, and parallel composition of processes. As usual, we consider processes up to a structural congruence, namely the smallest one generated by:

$$P|Q \equiv Q|P \qquad (P|Q)|R \equiv P|(Q|R) \qquad P|\mathsf{0} \equiv P. \tag{3}$$

We recognize the free, commutative monoid over $\{\mathsf{a}, \mathsf{s}\}$; that is, a category with a single object, terms up to "$\equiv$" as morphisms, composition $f \circ g = f|g$, and identity 0. (In this case, there is no distinction between agents and contexts: All morphisms are both.) Call this category $\mathcal{C}$.

We intend a to model a normal process and s to model two processes in a synchronized state. Here are the reaction rules:

$$(\mathsf{a}|\mathsf{a},\ \mathsf{s}\quad) \qquad \text{"two processes become synchronized,"} \tag{4a}$$

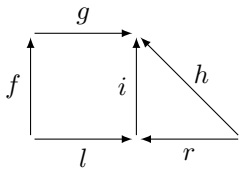
$$(\mathsf{s}\quad,\ \mathsf{a}|\mathsf{a}) \qquad \text{"synchronized processes drop synchronization."} \tag{4b}$$

Here are two reactions, using first rule 4a, then 4b: $\mathsf{a}|\mathsf{a}|\mathsf{s} \longrightarrow\!\!\!\!\!\rightarrow \mathsf{s}|\mathsf{s} \longrightarrow\!\!\!\!\!\rightarrow \mathsf{s}|\mathsf{a}|\mathsf{a}$.

Leifer and Milner gives a method for *deriving* labeled transitions for any reactive system. If the underlying category has *relative-* and *idem* pushouts, then the bisimulation on those labeled transitions is a congruence. (We abbreviate relative- and idem pushouts RPOs and IPOs, respectively. RPOs and IPOs are recalled in Appendix A.) Categories with pushouts have both RPOs and IPOs; the IPOs are precisely the pushouts.

To construct labeled transitions, we take as labels minimal contexts enabling reaction. The notion of IPO captures minimality. (In general, IPOs capture *minimal* contexts enabling reaction, whereas pushouts capture *least* such contexts.)

Definition 3. For a reactive system $(\mathcal{R}, \epsilon)$ over a category C, we define the *transition relation* by $f \xrightarrow{g} h$ iff there exists a context i and a reaction rule $(l, r) \in \mathcal{R}$ s.t. the following diagram commutes, and the square is an IPO.



(5)

Example 4. The elements of the free commutative monoid over $\{\mathsf{a}, \mathsf{s}\}$ are multisets over $\{\mathsf{a}, \mathsf{s}\}$; thus $\mathcal{C}$ is isomorphic to the category of multisets over $\{\mathsf{a}, \mathsf{s}\}$ (here, composition is multiset union). C has pushouts, given by multiset subtraction, and thus RPOs. Intuitively, the pushout of multisets simply adds what is missing: The pushout of a and $\mathsf{a}|\mathsf{a}$ is a and $\mathsf{0}$, the pushout of a and s is s and a. Because IPOs are precisely the pushouts in this category, we find transitions for an agent a by taking the pushout of a and either left-hand side of the two rules.

$$\mathsf{a} \xrightarrow{\mathsf{a}} \mathsf{s} \qquad \text{by rule (4a)} \tag{6}$$

$$\mathsf{a} \xrightarrow{\mathsf{s}} \mathsf{a}|\mathsf{a}|\mathsf{a} \qquad \text{by rule (4b)} \tag{7}$$

Because pushouts in $\mathcal{C}$ adds only what is missing, there is no transition from a with label $\mathsf{a}|\mathsf{a}$. The label can only add what is missing, and the agent a is only one "a" short of the left-hand side $\mathsf{a}|\mathsf{a}$ of rule (4a).

As mentioned, the bisimulation on such derived transition systems is a congruence whenever the underlying category has RPOs [14, 13]. It follows that the bisimulation on the transitions of the above example is a congruence.

Proposition 5 ([13])**.** *Let $(\mathcal{R}, \epsilon)$ be a reactive system on a category $\mathcal{C}$. If $\mathcal{C}$ has RPOs, then the bisimulation on the derived transitions is a congruence.*

3 Sortings

In this section, we recall notion of *sorting* for categories. As in the previous section, this section contains no original work except for the running example. The following notion of sorting is from [16]; an explicit notion of sorting first appears (for bigraphs) in [3], however, binding bigraphs of [2] is also a sorting.

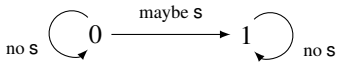
Definition 6 (Sorting)**.** A *sorting* of a category C is a functor $F : X \longrightarrow C$ that is faithful and surjective on objects.

We shall consistently confuse a sorting functor F with its domain: We write $F \longrightarrow C$, and we speak interchangeably of F as a category and as a functor.

Example 7. Suppose we want to restrict our process language such that at most two processes are synchronized at any given time. That is, no process can contain a subterm

s|s. We cannot just stipulate that our category contains no such terms, because the composition of s and s would then be undefined. Instead, we define a sorting $H \longrightarrow \mathcal{C}$ that refines the single homset of $\mathcal{C}$ into three homsets.

The category H has two objects, 0 and 1. The homset $H(0,0)$ contains morphisms of $\mathcal{C}$ that *do not* contain an s; the homset $H(0,1)$ contains morphisms that has *at most one* s; and the homset $H(1,1)$ contains morphisms that *do not* contain an s, that is, $H(1,1) = H(0,0)$. Here is a sketch of the H category:



Composition is defined as in the original $\mathcal{C}$ category. It is easy to check that this composition is well-defined on our refined homsets. For instance, for $f : 0 \longrightarrow 1$ and $g : 0 \longrightarrow 0$, the composite $f \circ g : 0 \longrightarrow 1$ does indeed contain at most one s.

Usually when we construct a sorting $F \longrightarrow C$, we will want to want to apply proposition 5 to the category F. Hence, we are usually interested in sortings that allow us to infer the existence of RPOs in F from the existence of RPOs in C. The following notion of *transfer* allows such inferences.

Definition 8 (Transfer of RPOs)**.** A sorting $F \longrightarrow C$ *transfers RPOs* iff whenever the image under F of a square s in F has an RPO, then that RPO has an F-preimage that is an RPO for s.

How do we establish that a sorting transfers RPOs? Jensen gives a sufficient condition, *safety*, for making that inference in his Thesis [7]; we generalized that condition in [16] (see also [19]). We recall the definition of safety in Appendix B.

Proposition 9 ([7])**.** *Let $F \longrightarrow C$ be a safe sorting. Then F transfer RPOs, and, if C has RPOs then so does F.*

4 Preservation of Semantics

In Example 7 above, we sort to get rid of unwanted morphisms in $\mathcal{C}$. The unwanted morphisms are those that do not satisfy the predicate "contain at most one s". Altogether, the sorting H of Example 7 is a *realization of a predicate* on the category C. The notion of sorting as a realization of a predicate is a major steps towards answering Question 2, "how do we construct a sorting for a given problem domain": Of the sortings in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], only a few are not realization of predicates.

However, not every sorting that enforces a predicate is equally interesting; we must require also that the sorted category supports the same reactive systems as the original one, at least when we restrict our attention to the "good" morphisms of the original category. In this section, as a first step towards answering Question 2, we give a sufficient condition for such preservation of semantics.

This result generalizes our previous [16], where we proved that a particular sorting preserves semantics. As in that paper, we will consider only predicates that are decomposable, that is, $P(f \circ g)$ implies $P(f)$ and $P(g)$. (This restriction is not a very severe;

see [16].) To formalize the notion "preserve semantics", we need the notion of reactive system reflected along a sorting.

Definition 10 (Reactive System reflected by F at $\bar{\epsilon}$)**.** Let $F \longrightarrow C$ be a sorting, and let $(\mathcal{R}, \epsilon)$ be a reactive system on C. For a preimage $\bar{\epsilon}$ of ϵ, the reactive system *reflected by F at $\bar{\epsilon}$* is $(\bar{\mathcal{R}}, \bar{\epsilon})$, where

$$\bar{\mathcal{R}} = \{(f, g) \mid f, g : \bar{\epsilon} \longrightarrow x \text{ for some } x \text{ and } (F(f), F(g)) \in \mathcal{R}\} \ . \tag{8}$$

We can now say precisely what it means to preserve reactions.

Definition 11 (Preserves P-reactions, transitions)**.** Let $F \longrightarrow C$ be a sorting, let $\mathcal{T}, \mathcal{S}$ be reactive systems on F, C, respectively, and let P be a decomposable predicate on C. *F preserves P-transitions for $\mathcal{T}, \mathcal{S}$* iff whenever f, g, h are morphisms of C with $P(g \circ f)$ and $P(h)$, then

$$f \xrightarrow{\ g\ } h \quad \text{iff} \quad \exists \hat{f}, \hat{g}, \hat{h}.\ \hat{f} \xrightarrow{\ \hat{g}\ } \hat{h} \text{ and } F(\hat{f}) = f, F(\hat{g}) = g, F(\hat{h}) = h. \tag{9}$$

(Similarly for reactions.)

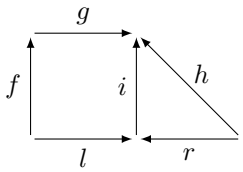
Theorem 12 below give sufficient conditions for a sorting to preserve P-reactions and -transitions, respectively. These conditions depend on the notions of "joint opcartesian lifting" and "weak joint opfibration", which were introduced in [16] and are recalled in Appendix C.

Theorem 12. *Let C be a category, and suppose we have a sorting $F \longrightarrow C$, a decomposible predicate P, and a reactive system $(\mathcal{R}, \epsilon)$ on C. Finally, let $\bar{\epsilon}$ be an F-preimage of ϵ, and consider the F-reflection $(\bar{\mathcal{R}}, \bar{\epsilon})$ of $(\mathcal{R}, \epsilon)$. Then F* preserves P-reactions for $(\bar{\mathcal{R}}, \bar{\epsilon}), (\mathcal{R}, \epsilon)$ *if (1) the image of F is P, (2) F lifts P-agents at $\bar{\epsilon}$, and (3) F is a* weak joint opfibration*. Moreover, F* preserves P-transitions if also *(4) F transfers and preserves RPOs.*

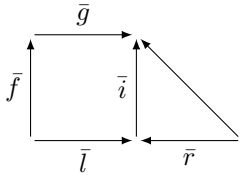
Proof. Part (a). Let f, g be morphisms of C with $P(f)$ and $P(g)$. Suppose first $\hat{f}, \hat{g}$ are preimages of f, g s.t.$\hat{f} \longrightarrow \hat{g}$. Then $\hat{f} = h \circ l$ and $\hat{g} = h \circ r$ for some h and some $(l, r) \in \bar{\mathcal{R}}$, so $(F(l), F(r)) \in \mathcal{R}$ and $f = F(\hat{f}) = F(h) \circ F(l) \longrightarrow F(h) \circ F(r) = F(\hat{g}) = g$.

Suppose instead that $f \longrightarrow g$. Then there exists h, l and r with $(l, r) \in \mathcal{R}$ s.t. $f = h \circ l$ and $g = h \circ r$. By (2) there are lifts of l, r at $\bar{\epsilon}$, so by (3), there is a jointly opcartesian lift $\bar{l}, \bar{r}$ of l, r at $\bar{\epsilon}$. Again by (2), we can lift $h \circ l$ and $h \circ r$ at ϵ; because $\bar{l}, \bar{r}$ jointly opcartesian, there is a lift $\bar{h}$ of h at $\mathrm{cod}(\bar{l}) = \mathrm{cod}(\bar{r})$. Because $(\bar{l}, \bar{r}) \in \bar{\mathcal{R}}$, we have $\bar{h} \circ \bar{l} \longrightarrow \bar{h} \circ \bar{r}$; by definition $f = F(\bar{h} \circ \bar{l})$ and $g = F(\bar{h} \circ \bar{r})$.

Part (b). Let f, g, h be morphisms of C with $P(g \circ f)$ and $P(h)$. Suppose first $f \xrightarrow{\ g\ } h$. Then there exists $(l, r) \in \mathcal{R}$ and a context h s.t. the following diagram commutes and the square is an IPO.

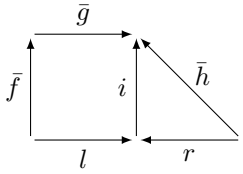


We find $P(i \circ l)$ because $P(g \circ f)$, and $P(i \circ r)$ because $P(h)$. By (3), F is a weak opfibration, so by (2), we have opcartesian lifts $\bar{f}$, $\overline{g \circ f}$, $\bar{l}$, $\bar{r}$, $\overline{i \circ l}$ and $\overline{i \circ r}$ at $\bar{\epsilon}$. Because $\bar{f}$ is opcartesian, we find a lift of $\bar{g}$ at the codomain of $\bar{f}$; we may assume this lift opcartesian. By (3), we may assume $\bar{l}, \bar{r}$ jointly opcartesian and $\overline{i \circ l}, \overline{i \circ r}$ a cospan, so there exists a lift $\bar{i}$ of i at $\mathrm{cod}(\bar{l}) = \mathrm{cod}(\bar{r})$. Again by (3), we may assume $\bar{g}, \bar{i}$ jointly opcartesian. Altogether, we have erected the following diagram.



By (4) and [19, Theorem 8], we have constructed an IPO. Obviously $(F(\bar{l}), F(\bar{r})) \in \mathcal{R}$, so we have a transition $\bar{f} \xrightarrow{\bar{g}} \bar{i} \circ \bar{r}$; clearly $F(\bar{i} \circ \bar{r}) = h$.

Suppose instead $\hat{f} \xrightarrow{\hat{g}} \hat{h}$. Then there exists i, l, r s.t. $(l, r) \in \bar{\mathcal{R}}$, the following diagram commutes, and the square is an IPO.



Clearly, $(F(l), F(r)) \in \mathcal{R}$), so by (4) and [19, Theorem 8], the image of the square is an IPO, and there is a transition $a \xrightarrow{L} b$. By (1), $P(f)$, $P(g)$, $P(h)$ and $P(g \circ f)$, so we have the desired transition $f \xrightarrow{g} h$. □

5 Closure Sorting

In this section, we take a second step towards answering Question 2: We define, for each decomposable predicate P, a *Closure sorting* realizing P. Every closure transfer RPOs and preserves P-reactions and -transitions. We define Closure sortings in terms of categories and reactive systems, but in Section 7, we lift them to bigraphs.

Suppose we want to construct a sorting realizing a decomposable predicate P on the morphisms of a category C. The basic problem here is that we may have morphisms f and g which satisfy P individually but not when composed:

$$x \xrightarrow{g} y \xrightarrow{f} z \quad , \qquad P(f),\ P(g),\ \neg P(f \circ g)\ . \tag{10}$$

At each preimage of y, we must choose whether to admit f or g. We make this choice explicit in the closure sorting by taking as pre-images for an object x pairs (F, G) of sets of morphisms such that every $g \in G$ can be composed with every $f \in F$, that is,

such that $P(g \circ f)$. This approach leads to rather a lot of objects in the sorted category, so we further insist that (F, G) be maximal in the sense that adding morphisms to that is either F or G would violate P by composition.

Define $g \perp f$ iff $P(g \circ f)$. Then define, for any $c \in C$, operators $\triangle$ and $\triangledown$ by

$$\begin{aligned} \triangle F &= \{g : c \longrightarrow x \mid g \perp F, \text{ any } x \in C\} \\ \triangledown G &= \{f : y \longrightarrow c \mid G \perp f, \text{ any } y \in C\} , \end{aligned} \tag{11}$$

where, e.g., $g \perp F$ is lifted pointwise. If you think of the composition $g \circ f$ as "g is put on top of f", then $\triangle F$ points towards the morphisms that can safely be put on top of F, and $\triangledown G$ points towards the morphisms that can safely by put below G[2]. We can now define that $(F, G)_c$ is *maximal* iff $\triangle F = G$ and $\triangledown G = F$.

Definition 13 (Closure sorting $\mathfrak{C}(P)$)**.** Let C be a category, and let P be a decomposable predicate on C. The *Closure sorting* $\mathfrak{C}(P) \longrightarrow C$ has *objects* $(F, G)_c$ where c is an object of C and F, G are sets of morphisms of C s.t. every $f \in F$ and $g \in G$ has $\mathrm{cod}(f) = c = \mathrm{dom}(g)$ and $P(g \circ f)$. Moreover, $(F, G)_c$ must be maximal. $\mathfrak{C}(P)$ has *morphisms* $k : (F, G)_c \longrightarrow (H, J)_d$ where

$$f \in F \implies k \circ f \in H \qquad \text{and} \qquad j \in J \implies j \circ k \in G \, . \tag{12}$$

It is fairly easy to establish that the $\triangle$ and $\triangledown$ form a Galois connection: $\triangle F \supseteq G$ iff $F \subseteq \triangledown G$. Thus $\triangle\triangledown$ and $\triangledown\triangle$ are indeed closure operators: $\triangle\triangledown\triangle F = \triangle F$ and $\triangledown\triangle\triangledown G = \triangledown G$. (Hence the name "Closure sortings".) We can use these closure operators to "fill up" a pair $(F, G)_c$ that is not maximal, taking either $(\triangledown G, \triangle\triangledown G)_c$ or $(\triangledown\triangle F, \triangle F)_c$. This realization is crucial in establishing that every closure sorting satisfies the premises of Proposition 9 and Theorem 12.

Lemma 13.1. *Let P be a decomposable predicate on C. Then $\mathfrak{C}(P)$ (1) is safe, (2) is a weak joint opfibration, and (3) lifts P-agents at every object c of C.*

It follows that every closure sorting sustain the behavioural theory of reactive systems in general and preserves semantics of individual reactive systems:

Theorem 14. *Let C be a category with RPOs, and let P be a predicate on C. Then $\mathfrak{C}(P)$ has RPOs, transfers RPOs, and preserves P-reactions and -transitions.*

Example 15. The sorting $H \longrightarrow C$ of Example 7 is indeed a closure sorting for the predicate "contains at most one s"; we recover the objects 0 and 1 as are $0 = (\triangledown\triangle\{\mathsf{0}\}, \triangle\{\mathsf{0}\})$ and $1 = (\triangledown\{\mathsf{0}\}, \triangle\triangledown\{\mathsf{0}\})$. By Theorem 14, H has RPOs and preserves both reactions and transitions within that predicate.

The closure sorting gives an answer for Question 2, "how do we construct a sorting given some problem domain". However, as formulated here, this answer applies only to reactive systems and not to bigraphs, which require the more general setting of supported reactive systems. We shall now lift the Closure sorting to this more general setting. First, we revisit S-categories and introduce supported reactive systems.

[2]The operators $\triangle$ and $\triangledown$ are closely related to the BiLog [20] adjuncts to composition.

6 Supported Reactive Systems

In this section, we generalize Milner and Jensen's Wide Reactive Systems [1, 5, 7] to *supported reactive systems*. Supported reactive systems encompass bigraphs; in the next section, we use them to give our final answer to Question 1, "what sortings sustain the behavioural theory of bigraphs". Milner and Leifer invented the notion of support because the category of bigraphs does not possess RPOs. They overcome this deficiency by enriching the category of bigraphs with *location of subterms* [1, 3, 2]; The notion of support and S-categories formalize this enriching.

Example 16. Informally, we add support to our original process language of Example 2 by adding an identity (a number) to each atom a and s One supported variant of the term $\mathsf{a}|\mathsf{a}$ is $\mathsf{a}_0|\mathsf{a}_1$, another is $\mathsf{a}_1|\mathsf{a}_2$. With support, we can differentiate between subterms: a_0 and a_1 are two different subterms of $\mathsf{a}_0|\mathsf{a}_1$. Moreover, we get a notion of agents sharing a subterm: $\mathsf{a}_0|\mathsf{a}_1$ and $\mathsf{a}_1|\mathsf{a}_2$ share the subterm a_1 and contain distinct subterms a_0 and a_2, respectively.

Obviously, an identity should occur at most once in a term; if we admit $\mathsf{a}_0|\mathsf{a}_0$ as a term, we cannot use the identity 0 to locate a subterm. But what then is the result of the composition $\mathsf{a}_0 \circ \mathsf{a}_0$? Milner and Leifer proposes that it be undefined, and thus arrive at *supported pre-categories* or *s-categories* [14, 1]. These are like categories, except composition is defined only on morphisms with disjoint support sets.

Definition 17 (S-category)**.** An S-category S comprises a set of objects; a set of morphisms $S(a,b)$ for every a, b; an identity $1_a \in S(a,a)$ for every a; a *partial composition operation* $\circ : S(a,b) \times S(b,c) \rightharpoonup S(a,c)$ for every a, b, c; and a *support operation* $|a|$ which assigns to each morphism f a finite set, the *support* of a. Composition with identities is always defined and $1_b \circ f = f = f \circ 1_a$. Composition is associative: Either $f \circ (g \circ h) = (f \circ g) \circ h$ or both sides are undefined. Finally, S must satisfy that

1. $f \circ g$ is defined iff $|f| \cap |g| = \emptyset$; in this case, $|f \circ g| = |f| \cup |g|$; and
2. for any $f : a \longrightarrow b$ and any injective function ϕ with $|f| \subseteq \mathrm{dom}(\phi)$, there is a morphism $\phi \cdot f : a \longrightarrow b$, the *support translation* of f by ϕ, subject to 6 additional conditions found in Appendix D.

Morphisms $f, g : a \longrightarrow b$ are *support equivalent*, $f \doteq g$, iff there is a support translation ϕ with $\phi \cdot f = g$; support equivalence is an equivalence relation. Finally, an S-functor is like a functor, except it must respect support-equivalence.

The (unlisted) requirements of the support translation capture our intuition that the support supplies identity to the sub-components of a morphism; that it does not matter precisely what support one chooses; and, crucially, that support translation does not yield a morphism that is different in any substantial way.

Example 18. We construct a supported variant of the category $\mathcal{C}$ from example 2. The S-category $\mathcal{S}$ has a single object and as morphisms subsets of $\{\mathsf{a}_i \mid i \in \mathbb{N}\} \cup \{\mathsf{s}_i \mid i \in \mathbb{N}\}$; in keeping with the process calculus intuition of Example 2, we write such a

morphism as, e.g., $\mathsf{a}_2|\mathsf{a}_3|\mathsf{s}_5$ rather than $\{\mathsf{a}_2, \mathsf{a}_3, \mathsf{s}_5\}$. The support $|f|$ of a morphism is the set of indices in f, e.g., the support of $\mathsf{a}_2|\mathsf{a}_3|\mathsf{s}_5$ is $\{2, 3, 5\}$. Composition $f \circ g$ is defined when $|f| \cap |g| = \emptyset$, in which case $f \circ g = f \cup g$.

For a S-category S, we obtain a category $[S]$ by taking the quotient over "$\doteq$".

Definition 19 (Support quotient)**.** Let S be an S-category. The *support quotient* $[S]$ of S is the has the same objects as S and support equivalence classes of S-morphisms as morphisms:

$$[S](a, b) = \{[f] \mid f \in S(a, b)\}.$$

(We denote by $[f]$ the support equivalence class represented by f.) The support of any morphism in $[S]$ is empty. Finally, the assignment $f \mapsto [f]$ defines an S-functor $\eta_S : S \longrightarrow [S]$.

S-categories and S-functors form a subcategory SCAT. Any category C can be considered a S-category if we add empty support to every morphism of C; thus, we have an inclusion $I : \text{CAT} \longrightarrow \text{SCAT}$ to be this inclusion. The support quotient operation $[-]$ induces a functor $[-] : \text{SCAT} \longrightarrow \text{CAT}$, taking an S-functor $M : S \longrightarrow U$ to the functor $[M] : [S] \longrightarrow [U]$: Simply take $[M](a) = M(a)$ and $[M](f) = [M(f)]$. The support quotient functor $[-]$ is left-adjoint to I, with unit η:

$$\text{SCAT} \underset{I}{\overset{[-]}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \text{CAT}$$

Example 20. The category $\mathcal{C}$ of example 2 is the quotient of the S-category $\mathcal{S}$ of example 18, that is, $\eta_{\mathcal{S}} : \mathcal{S} \longrightarrow [S] \simeq I(\mathcal{C})$.

Our definitions of sortings and reactive systems lifts straightforwardly to S-categories. The one difference is that we require that reaction rules of a reactive system are closed under support equivalence.

An S-category S supplies its quotient category $[S]$ with information about location of subparts. With this extra information, we derive more finely nuanced transition system for S, by taking as reactions and transitions in $[S]$ the images of reactions and transitions of a reactive system on S, respectively.

Definition 21 (Supported reactive system)**.** A *supported reactive system over* $[S]$ is a reactive system $(\mathcal{R}, \epsilon)$ on S; it has *supported reactions* and *supported transitions* in $[S]$ given by

$$\begin{aligned} \eta_S(f) &\longrightarrow \eta_S(g) \qquad \text{iff} \qquad f \longrightarrow g \\ \eta_S(f) &\xrightarrow{\eta_S(g)} \eta_S(h) \qquad \text{iff} \qquad f \xrightarrow{g} h, \end{aligned} \tag{13}$$

where f, g, h are morphisms of $\mathcal{S}$.

Example 22. We construct a supported reactive system on $\mathcal{S}$, analogous to the one we constructed on $\mathcal{C}$ in Example 2. We shall see how the supported transitions we get here differ from the transitions of Example 2. As rules, we take the pairs

$$(\mathsf{a}_i|\mathsf{a}_j, \mathsf{s}_k) \text{ for } i \neq j \qquad \text{and} \qquad (\mathsf{s}_i, \mathsf{a}_j|\mathsf{a}_k) \text{ for } j \neq k\ . \tag{14}$$

To compare with Example 2, we find the supported transitions of the agent a (in $[\mathcal{S}] \simeq I(\mathcal{C})$) has. To find these, we look to the S-category $\mathcal{S}$ and find transitions

$$\begin{array}{rll} \mathsf{a}_i \xrightarrow{\mathsf{a}_j} \mathsf{s}_k & & (i \neq j) \\ \mathsf{a}_i \xrightarrow{\mathsf{a}_j|\mathsf{a}_k} \mathsf{a}_i|\mathsf{s}_l & & i \notin \{j,k,l\},\ j \neq k \\ \mathsf{a}_i \xrightarrow{\mathsf{s}_j} \mathsf{a}_i|\mathsf{a}_k|\mathsf{a}_l & & j \neq i,\ i,k,l \text{ distinct.} \end{array} \tag{15}$$

We then take the quotient to find supported transitions $\mathsf{a} \xrightarrow{\mathsf{a}} \mathsf{s}$, $\mathsf{a} \xrightarrow{\mathsf{a}|\mathsf{a}} \mathsf{a}|\mathsf{s}$, and $\mathsf{a} \xrightarrow{\mathsf{s}} \mathsf{a}$. Contrast these supported transitions with the transitions of Example 4, where there were no transition $\mathsf{a} \xrightarrow{\mathsf{a}|\mathsf{a}} \mathsf{a}|\mathsf{s}$. Because of support, we get finer labels: The context can choose to interact with only a part of the agent. Because of the support quotient, we do not distinguish precisely where the context and agent interacts.

Milner proved that bisimulation on supported transitions is a congruence when S has RPOs. (He proved it for Wide Reactive Systems, but the proof carries over.)

Proposition 23 ([1])**.** *Let $(\mathcal{R}, \epsilon)$ be a supported reactive system on $[S]$, and let S have RPOs. Then bisimulation on the supported transitions is a congruence.*

It is easy to see that $\mathcal{S}$ of Example 22 has RPOs; hence, bisimulation on the supported transitions given there is a congruence.

7 Sortings for Supported Reactive Systems

Finally, we can answer Questions 1 and 2 of the introduction. Our answer to Question 1, "what sortings sustain the theory of bigraphs", will be that *all sortings that we know to sustain the theory of reactive systems by virtue of [16, 7, 1] can be lifted to sortings that sustain the theory of* supported *reactive systems.* Our answer to Question 2, "how do we construct a sorting for a given problem domain", comes out as a special case: We lift the Closure sorting of Section 5. By the answer to Question 1, this lifted Closure sorting sustains the theory of supported reactive systems. We mentioned that the Closure sorting yields most of the sortings we have seen in the literature so far; we allow ourselves the hope that it will yield most of the sortings we will need in the future.

First, we will need pullbacks along sortings of categories.

Lemma 23.1. SCAT *has pullbacks along (ordinary) functors.*

Here is how we lift sortings. Suppose we have an S-category S that has RPOs, and a sorting $F \longrightarrow [S]$ (see Figure 1-(i)). We take the pullback $F^\star$ of F along η_S (see Figure 1-(ii)); $F^\star \longrightarrow S$ is easily seen to be an S-sorting. We are not done yet, however, because we want $F^\star$ to be support for F. Fortunately, Proposition 24 below states that F is indeed isomorphic to $[F^\star]$ (see Figure 1-(iii)).

Proposition 24. *Let S be an S-category, and let $F \longrightarrow [S]$ be a sorting on $[S]$. Then $F^\star$ is a sorting $F^\star \longrightarrow S$, and F is isomorphic to $[F^\star]$.*

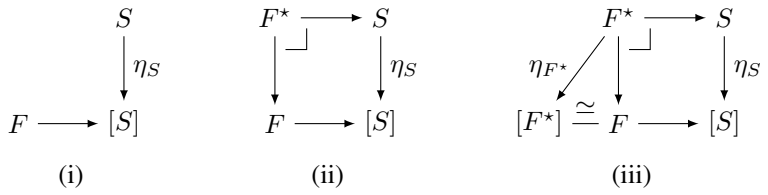


Figure 1: Lifting a sorting on $[S]$ to a sorting on S

We want $F^\star$ to transfer RPOs: By proposition 23, any supported reactive system on F will then have a bisimulation on its supported transitions that is a congruence. Luckily, the pullback $F^\star$ preserves Jensen's safety condition. (The notion of safety and Proposition 9 are readily verified to hold in the setting of S-categories.)

Proposition 25. *Safety in* SCAT *is preserved by pullback along pure functors.*

This is nearly magic. $F \longrightarrow [S]$ transfer RPOs even though there might be no RPOs to transfer; however, the transfer-property lifts across the pullback to $F^\star \longrightarrow S$ where there *are* RPOs to transfer.

Theorem 26. *Let S be an S-category with RPOs, and let $F \longrightarrow [S]$ be a sorting on $[S]$. If F is safe, then $F^\star$ has RPOs.*

In the case of (non-supported) reactive systems, we have Theorem 12 to helps us preserve transitions under a predicate. We obtain an analogous result for supported reactive systems in Theorem 28 below. First, we note that we can reflect supported reactive systems along pullbacks.

Lemma 26.1. *Let S be an S-category, let $(\mathcal{R}, \epsilon)$ be a reactive system on S, and let $F \longrightarrow [S]$ be a sorting on $[S]$. Reflecting $(\mathcal{R}, \epsilon)$ along the pullback $F^\star$ of F along η_S yields a supported reactive system $(\mathcal{R}^\star, \epsilon^\star)$ over F.*

Now we can define what it means to preserve supported P-transitions.

Definition 27 (Preserves support P-reactions, -transitions)**.** Let $F \longrightarrow [S]$ be a sorting, let P be a decomposable predicate on $[S]$, and suppose we have supported reactive systems on $[S]$ and F. We say that *F preserves supported P-reactions (-transitions)* iff for any f, g, h in $[S]$ with $P(g \circ f)$ and $P(h)$,

$$f \xrightarrow{g} h \quad \text{iff} \quad \exists \hat{f}, \hat{g}, \hat{h}.\ \hat{f} \xrightarrow{\hat{g}} \hat{h} \text{ and } F(\hat{f}) = f, F(\hat{g}) = g, F(\hat{h}) = h. \qquad (16)$$

Theorem 28. *Let P be a decomposable predicate on $[S]$, let $F \longrightarrow [S]$ be a sorting, and let $(\mathcal{R}, \epsilon)$ be a supported reactive system on $[S]$, and consider an F-reflection of $(\mathcal{R}, \epsilon)$ at $\bar{\epsilon}$. Then (a) F respects supported reaction semantics at $\bar{\epsilon}$ in F if (1) the image of F is P, (2) F lifts P-agents at $\bar{\epsilon}$, and (3) F is a weak joint opfibration. Moreover, (b) F respects supported transition semantics if also (4) F transfer RPOs and S has RPOs.*

Together, Theorems 26 and 28 answer Question 1, "which sortings sustain the behavioural theory of bigraphs." (Theorem 26 also defines the admittedly rather vague phrase, "sustain the behavioural theory of bigraphs".)

Finally, because the closure sorting $\mathfrak{C}(P)$ is very well-behaved in the case of ordinary categories and reactive systems (see Lemma 13.1 and Theorem 14), we invoke Theorems 26 and 28 to see that it is indeed well-behaved in the case of S-categories and supported reactive systems.

Theorem 29. *Let S be an S-category with RPOs, and let P be a decomposable predicate on $[S]$. Then (a), $\mathfrak{C}(P)^\star$ has RPOs; (b), the bisimulation on the supported transitions of any supported reactive system on $\mathfrak{C}(P)$ is a congruence; (c), $\mathfrak{C}(P)$ respects supported P-reactions and P-transitions.*

This Theorem answers Question 2, "How do we construct a sorting for a given problem domain."

8 Local Bigraphs is a Closure Sorting

For the finale, we prove that Milner's Local bigraphs [4, 10] arise as a closure-sorting of pure bigraphs [1, 5]. (We have so far called "pure bigraphs" simply "bigraphs".)

We do not reiterate the definition of pure and local bigraphs here. Refer to one of [1, 5] for the definition of pure bigraphs; to one of [20, 11] for intuition and examples of pure bigraphs; and to [10, 4] for the definition of local bigraphs. We will however need to recall the notion of signature [4, 10, 1, 5].

A *signature* comprises a set of control symbols and a map assigning a non-negative arity to each symbol. We write $\mathbf{B}(\Theta)$ for the category of abstract bigraphs over the signature Θ. A *binding signature* further partitions each arity n into binding ports $0 \ldots k$ and pure ports $k+1 \ldots n-1$. We write $\mathbf{L}(\Sigma)$ for the category of abstract local bigraphs over a binding signature Σ.

Write $U(\Sigma)$ for the pure signature obtained from the binding signature Σ by forgetting the partitioning of arities. Recall that U induces a forgetful functor from local to pure bigraphs; we write this functor $U : \mathbf{B}(\Sigma) \longrightarrow \mathbf{B}(U(\Sigma))$.

We derive a predicate from the Scope rule of [10, 4] simply by disregarding the parts of that scope rule that mentions local interfaces.

Definition 30 (Scope)**.** Let Σ be a binding signature. The *Scope predicate for* Σ is a predicate $\mathbf{P}_\Sigma$ on the morphisms of $\mathbf{B}(U(\Sigma))$, that is, on the pure bigraphs over $U(\Sigma)$. For a bigraph f of $\mathbf{B}(U(\Sigma))$, we define $\mathbf{P}_\Sigma(f)$ iff whenever a binding port p on a node n is in a link, then any other port p' in that link is on a node that has n as an ancestor.

This predicate is easily seen to be decomposable, so we can form the closure sorting $\mathfrak{C}(\mathbf{P}_\Sigma) \longrightarrow \mathbf{B}(U(\Sigma))$. This closure sorting is indeed isomorphic to the abstract category $\mathbf{P}_\Sigma$ of local bigraphs over Σ.

Theorem 31. *Let Σ be a binding signature. Then the category $\mathbf{L}(\Sigma)$ is isomorphic to $\mathfrak{C}(\mathbf{P}_\Sigma)$, the domain of the closure sorting $\mathfrak{C}(\mathbf{P}_\Sigma) \longrightarrow \mathbf{B}(U(\Sigma))$.*

The proof follows shortly. First, notice that we derive both abstract and concrete local bigraphs *automatically* from the predicate P_Σ: The abstract local bigraphs are given by the closure sorting $\mathfrak{C}(P_\Sigma)$; the concrete local bigraphs are given by the pull-back $\mathfrak{C}(P_\Sigma)^\star$.

It follows that the behavioural theory local bigraphs sustain the behavioural theory of pure bigraphs. Milner proved as much by hand in [10, 4]; now, we get the same result from Theorem 29.

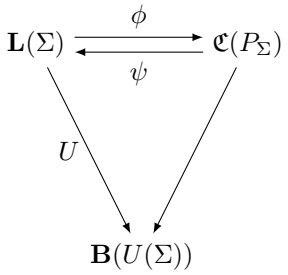
Corollary 32. *Let Σ be a binding signature. For any supported reactive system on $\mathbf{L}(\Sigma)$, bisimilarity on the supported transitions is a congruence.*

However, we also *expand* on Milner's results, because Theorem 29 *also* says that local bigraphs preserves supported P_Σ-reactions and -transitions.

Corollary 33. *Let Σ be a binding signature. Then the sorting $\mathbf{L}(\Sigma) \longrightarrow \mathbf{B}(U(\Sigma))$ respects supported P_Σ-reactions and -transitions.*

8.1 Proof of Theorem 31

We define functors $\phi : \mathbf{L}(\Sigma) \longrightarrow \mathfrak{C}(P_\Sigma)$ and $\psi : \mathfrak{C}(P_\Sigma) \longrightarrow \mathbf{L}(\Sigma)$, and we prove that the following diagram commutes; in particular $\phi \circ \psi = 1 = \psi \circ \phi$.



First, define ϕ on objects, i.e., local interfaces by

$$\phi(\vec{x}) = (\{U(f) | f : \vec{y} \longrightarrow \vec{x}\}, \{U(g) | g : \vec{x} \longrightarrow \vec{z}\})_{U(\vec{x})} \ .$$

Define ϕ on morphisms (local bigraphs) by

$$\phi(f : \vec{x} \longrightarrow \vec{y}) = U(f) : \phi(\vec{x}) \longrightarrow \phi(\vec{y})$$

9 Conclusion

First, we have extended the theory of sortings for reactive systems with a new construction of a sorting for decomposable predicates, the Closure sorting. Second, we have generalized the theory of sortings for reactive systems to supported reactive systems, which encompass bigraphical reactive systems. Third, we have proved that the resulting general theory of sortings for bigraphical reactive systems sustains the behavioural

theory for bigraphical reactive systems qua preservation of RPOs and preservation of semantics. Finally, we proved that local bigraphs arise naturally as a Closure sorting obtained from the scope condition. Besides alleviating the need for redeveloping the behavioural theory for local bigraphs, it supports local bigraphs as the natural extension of bigraphs with local names. We conjecture that the remaining sortings we have seen for bigraphs in the literature so far can also be obtained as Closure sortings.

As an alternative to S-categories, Leifer suggested bicategories [17]; Sassone and Sobocinsky subsequently lifted Leifer and Milner's original work on reactive systems [13] to 2-categories of cospans [21, 22] and significantly developed the idea. The idea of automatically derived transition systems originates with Sewell [15], and have lately been developed for graph transformation systems [23, 24]. Braione studied a more intentional notion of sorting for reactive systems in [25].

We see two main avenues of future work. One is to investigate sortings in other frameworks, in particular within graph rewriting and the 2-categorical approach mentioned above. Another is to investigate the algebraic properties of sortings and if the Closure sorting is somehow universal among sortings that capture a decomposable predicate and respect the behavioural theory.

References

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

[2] Jensen, O.H., Milner, R.: Bigraphs and transitions. In: POPL '03. (2003) 38–49

[3] Milner, R., Leifer, J.J.: Transition systems, link graphs and Petri nets. Technical Report 598, U. of Cambridge Computer Laboratory (2004)

[4] Milner, R.: Bigraphs whose names have multiple locality. Technical Report 603, U. of Cambridge Computer Laboratory (2004)

[5] Jensen, O.H., Milner, R.: Bigraphs and mobile processes (revised). Technical Report 580, U. of Cambridge Computer Laboratory (2004)

[6] Milner, R.: Bigraphs for petri nets. In: Lectures on Concurrency and Petri Nets. Volume 3098 of LNCS. (2003) 686–701

[7] Jensen, O.H.: Mobile Processes in Bigraphs. PhD thesis, U. of Aalborg (2006)

[8] Bundgaard, M., Sassone, V.: Typed polyadic pi-calculus in bigraphs. In: PPDP '06. (2006) 1–12

[9] Bundgaard, M., Hildebrandt, T.: Bigraphical semantics of higher-order mobile embedded resources with local names. In: GT-VC '05. Volume 154 of ENTCS. (2006) 7–29

[10] Milner, R.: Local bigraphs and confluence: Two conjectures. In: EXPRESS '06 (Preliminary proceedings). (2006) 42–50

[11] Birkedal, L., Debois, S., Elsborg, E., Hildebrandt, T., Niss, H.: Bigraphical Models of Context-aware Systems. In: FOSSACS '06. Volume 3921 of LNCS. (2006)

[12] Elsborg, E.: Bigraphical Location Models. Technical Report 94, IT U. of Copenhagen (September 2006)

[13] Leifer, J.J., Milner, R.: Deriving bisimulation congruences for reactive systems. In: CONCUR '00. (2000) 243–258

[14] Leifer, J.J.: Operational Congruences for Reactive Systems. PhD thesis, U. of Cambridge Computer Laboratory and Trinity College (2001)

[15] Sewell, P.: From rewrite rules to bisimulation congruences. Theoretical Computer Science **274**(1–2) (2002) 183–230

[16] Birkedal, L., Debois, S., Hildebrandt, T.: Sortings for reactive systems. In: CONCUR '06. Volume 4137 of LNCS., Springer (2006) 248–262

[17] Leifer, J.J.: Synthesising labelled transitions and operational congruences in reactive systems, part 2. Technical Report RR-4395, INRIA (2002)

[18] Birkedal, L., Debois, S., Hildebrandt, T.: Sorted bigraphs. Technical report, IT U. of Copenhagen (2007) To appear. Draft at `http://itu.dk/people/debois/tr07.pdf`.

[19] Birkedal, L., Debois, S., Hildebrandt, T.: Sortings for reactive systems. Technical Report 84, IT U. of Copenhagen (2006)

[20] Conforti, G., Macedonio, D., Sassone, V.: Spatial logics for bigraphs. In: ICALP '05. Volume 3580 of LNCS. (2005) 766–778

[21] Sassone, V., Sobocinski, P.: Reactive systems over cospans. In: LICS '05, IEEE (2005) 311–320

[22] Sassone, V., Sobocinski, P.: Deriving bisimulation congruences: 2-categories vs. precategories. In: FOSSACS '03. Volume 2620 of LNCS. (2003) 409–424

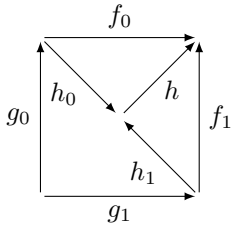
[23] Bonchi, F., Gadducci, F., König, B.: Process bisimulation via a graphical encoding. In: ICGT '06. Volume 4178 of LNCS. (2006) 168–183

[24] Ehrig, H., König, B.: Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. Mathematical Structures in Computer Science **16**(6) (2006) 1133–1163

[25] Braione, P., Picco, G.P.: On calculi for context-aware coordination. In: COORD '04. Volume 2949 of LNCS. (2004) 38–54

A Relative Pushouts

Definition 34 (Relative pushout)**.** Consider the following diagram.



(17)

Suppose the outer square commutes. The triple (h_0, h_1, h) is an RPO for g_0, g_1 to f_0, f_1 iff the entire diagram commutes and (h_0, h_1, h) is universal, that is, if (h'_0, h'_1, h') has $h'_0 \circ g_0 = h'_1 \circ g_1$ and $f_i = h' \circ h'_i$, then there exists a unique k s.t. $h = h' \circ k$ and $h'_i = k \circ h_i$. If $(f_0, f_1, 1)$ is an RPO for g_0, g_1 to f_0, f_1, we say that (f_0, f_1) is an *idem pushout* (IPO) for g_0, g_1.

Intuitively, if (h_i, h) is an RPO for g_i to f_i, then h is the common part of the contexts f_i. The universality condition says that h is as big as possible: If h' is an alternative common part, then it must factor h, and there are thus commonalities in the f_i captured by h but not by h'. With this intuition, if f_i is an IPO for g_i, the f_i are minimal contexts making up for the differences between the g_i.

B Safety

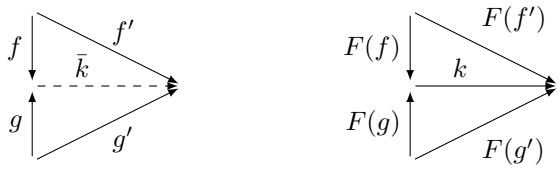
Definition 35 (Safety [7, 16])**.** A sorting $F \longrightarrow C$ is *safe* iff

1. F is a *weak opfibration*, that is, any morphism f of F can be written $\varphi \circ \hat{f}$, where φ is a vertical and $\hat{f}$ is opcartesian.
2. F *reflects prefixes*, that is, whenever $F(f) = g \circ h$, then there is $\bar{h}$ s.t. $F(\bar{h}) = h$ and $\bar{h}$ and f have the same domain.
3. F has *vertical pushouts*, that is, the fibres of F has pushouts, and each of those pushouts are also pushouts in F.

C Jointly Opcartesians and Weak Joint Opfibrations

Definition 36 (Jointly opcartesian)**.** Let $F \longrightarrow C$ be a functor. A cospan f, g in C is said to be *jointly opcartesian* iff whenever f', g' is a cospan, f, f' is a span, and g, g' is a span (see the diagram below, left side) with $F(f') = k \circ F(f)$ and $F(g') = k \circ F(g)$ (see the diagram below, right side), then there exists a unique lift $\bar{k}$ of k s.t. $f' = \bar{k} \circ f$

and $g' = \bar{k} \circ f'$.



Example 37. In the sorting $F \longrightarrow C$ of example 7, the cospan a, a has a jointly opcartesian lift $\mathsf{a} : 0 \longrightarrow 0 \longleftarrow 0 : \mathsf{a}$; the cospan s, a has a jointly opcartesian lift $\mathsf{s} : 0 \longrightarrow 1 \longleftarrow 0 : \mathsf{a}$.

Definition 38 (Weak Joint Opfibration [16])**.** A functor $F : X \longrightarrow C$ is a *weak joint opfibration* iff whenever $F(f), F(g)$ form a cospan in C, then there exist a jointly opcartesian pair $\hat{f}, \hat{g}$ with $F(\hat{f}) = F(f)$ and $F(\hat{g}) = F(g)$.

D Support Translation

Here are the 6 conditions missing from Definition 17.

$$\phi \cdot 1_A = 1_A \tag{18}$$
$$\phi \cdot (g \circ f) = (\phi \cdot g) \circ (\phi \cdot f) \tag{19}$$
$$1_{|f|} \cdot f = f \tag{20}$$
$$(\phi \circ \phi') \cdot f = \phi \cdot (\phi' \cdot f) \tag{21}$$
$$\phi \cdot f = (\phi \downarrow |f|) \cdot f \tag{22}$$
$$|\phi \cdot f| = \phi(|f|) \tag{23}$$