

Sortings for reactive systems

Lars Birkedal **Søren Debois** Thomas Hildebrandt

Bigraphical Programming Languages Group
IT University of Copenhagen

CONCUR, Bonn, August 2006

Reactive systems

Definition

A reactive system over a category \mathbf{B} consists of

- a **distinguished object** of that category, say ϵ ;
- a set of **reaction rules**, say R , consisting of pairs (l, r) of morphisms $l, r : \epsilon \rightarrow X$.

We think of the latter as rewrite rules. The left-hand side $l : \epsilon \rightarrow X$ is rewritten to the right-hand side $r : \epsilon \rightarrow X$.

The reaction rules generate a **reaction relation** by closing under all contexts. That is, for all contexts C ,

$$C \circ l \longrightarrow C \circ r.$$

Reactive systems

Example category

Here is a toy process language. Let A be a set of actions.

$$P, Q ::= 0$$
$$\quad \quad \quad | a \quad \quad a \in A$$
$$\quad \quad \quad | P|Q$$

Quotient by a structural congruence.

$$P|Q \equiv Q|P \quad (P|Q)|R \equiv P|(Q|R) \quad P|0 \equiv P$$

We recognize the free commutative monoid over A . We take this monoid as our category **B**. (Single object, terms as morphisms, $f \circ g = f|g$.)

Reactive systems

Example semantics

The single object of **B** is the distinguished ϵ .

Processes $A = \{b, c, m\}$, where b is a normal process, c is a critical process, and m is a mutex. The reaction rules R are

$(b m, c \quad)$	“acquire the mutex and become critical.”
$(c \quad, b m)$	“become normal and release the mutex.”

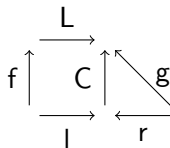
Example reactions:

$$b|b|m \longrightarrow b|c \longrightarrow b|b|m.$$

Reactive systems

Transitions

A transition is the **least context enabling reaction**. That is, there is $f \xrightarrow{L} g$ iff there is a C and a rule (l, r) s.t. the following diagram commute and the square is “least”:



L, C pushout of f, l would be “least”. Our example category has pushouts, but some categories we are interested in do not, whence we require instead that L, C is an IPO for f, l .

The key selling point of reactive systems is the following theorem.

Theorem (Milner & Leifer)

For any reactive system over \mathbf{B} , if \mathbf{B} has RPOs (or pushouts), bisimulation is a congruence.

Downside: Maybe we don't like the labels we get in a reactive system.

Reactive systems

Example transitions

The elements of the free commutative monoid are actually multisets; the composition multiset union. Pushouts are given by multi-set subtraction.

$$\begin{array}{ccc} & Y - X & \\ & \longrightarrow & \\ X & \uparrow & \uparrow & X - Y \\ & \longleftarrow & \\ & Y & \end{array}$$

Transitions in our calculus adds what's missing from reaction rules.

$$b \xrightarrow{m|} c \quad \text{using } (b|m, c)$$

$$b \xrightarrow{c|} b|b|m \quad \text{using } (c, b|m)$$

$$c|b \xrightarrow{m|} c|c \quad \text{using } (b|m, c)$$

(Maybe we don't like the latter transition; the context shouldn't be able to arbitrarily release mutexes.)

Definition

A **sorting** of a reactive system over \mathbf{B} is a functor $p : \mathbf{E} \rightarrow \mathbf{B}$ that is faithful and surjective on objects.

Intuition: A preimage of a morphism f is a **sort** or **type** for f .

Because p is faithful, each homset in \mathbf{E} is a subset of its p -image.

Because p is surjective on objects, every homset in \mathbf{B} has a preimage in \mathbf{E} .

Altogether, p **refines** the homsets of \mathbf{B} .

Sorting

Example

Suppose that we do not want to allow terms containing more than one mutex, that is, containing a subterm $m|m$.

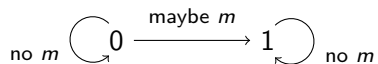
We can't just say "consider \mathbf{B} but omit terms containing $m|m$ as a subterm" because composition is no longer well-defined. What would the composite of m with m be? $m|m$?

Instead, we use sorting to distinguish morphisms with and without an m .

Sorting

Example (cont'd)

Here is our **E** category:



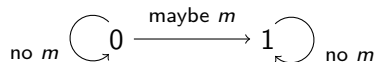
Morphisms $f : 0 \rightarrow 0$ cannot contain an m , e.g., $c : 0 \rightarrow 0$ or $c|b : 0 \rightarrow 0$.

Morphisms $f : 0 \rightarrow 1$ may contain an m , e.g., $m|b : 0 \rightarrow 1$, but also $c|b : 0 \rightarrow 1$ and $0 : 0 \rightarrow 1$.

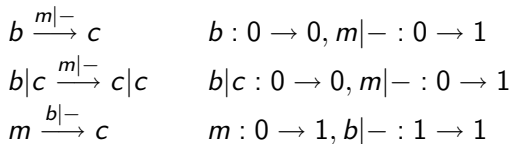
Morphisms $f : 1 \rightarrow 1$ cannot contain an m , e.g., $c : 1 \rightarrow 1$.

Sorting

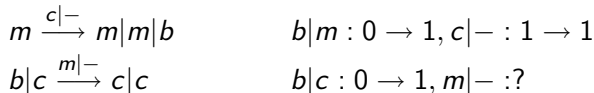
Example transitions



Assuming appropriate lifting of the reaction rules, here are some example transitions.



And here are some non-transitions.



Sorting

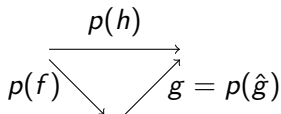
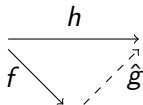
Preservation of RPOs

Given a sorting $p : \mathbf{E} \rightarrow \mathbf{B}$, does \mathbf{E} possess RPOs? (Recall that our \mathbf{B} has RPOs because it has pushouts.)

Ole Jensen has an answer in his thesis (although that answer is intended only for bigraphs and formulated in a specific instance of sorting in the setting of supported precategories.)

We have reformulated his answer in terms of **opcartesian liftings** and a generalization (weakening) of the concept of **opfibrations**.

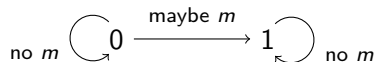
For a functor $p : \mathbf{E} \rightarrow \mathbf{B}$, a morphism f is **opcartesian** iff for all h (see diagrams).



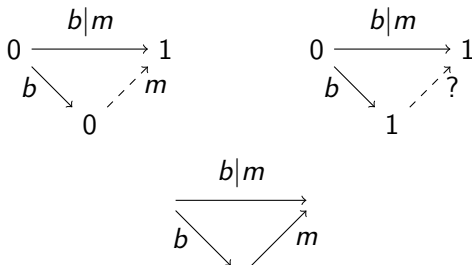
Sorting

Opcartesians (cont'd)

Recall our example sorting **E**.



Consider the agent b in **B**. Does it have an opcartesian lifting at 0?



Yes; $b : 0 \rightarrow 0$. Opcartesians are analogous to principal types (most general/most flexible types).

Definition (Weak opfibration)

A functor $p : \mathbf{E} \rightarrow \mathbf{B}$ is a **weak opfibration** iff whenever $f : B \rightarrow B'$ has a lift at E , it has in fact an opcartesian lift at E .

Intuition: Every typeable agent has a principal type.

Incidentally, this definition is the most we can weaken the standard definition of opfibration while maintaining the property that any f in \mathbf{E} can be written as $\phi \circ \bar{f}$ for a vertical ϕ and an opcartesian \bar{f} .

A sorting that is a weak opfibration (and satisfies minor additional requirements) transfers RPOs.

We saw in the example that we can use sorting to “get rid of unwanted morphisms”, in that case those with subterms $m|m$.

Suppose the “unwanted” morphisms are identified by a predicate P on the morphisms of \mathbf{B} . Three questions arise.

- 1 How do we go from a predicate to a sorting?
- 2 How do we ensure preservation of RPOs?
- 3 How do we ensure preservation of intended semantics?

Sorts and predicates

Answer

Call a predicate P **decomposable** iff $P(f \circ g) \implies P(f), P(g)$.

We see immediately that the predicate “has no subterm $m|m$ ” is decomposable.

Theorem

For any decomposable predicate P , there is a sorting with image P preserving RPOs and semantics.

“Preserving semantics” means that, under the obvious lift of rewrite rules,

- 1 the image of a reaction in **E** is a reaction in **B**;
- 2 a reaction in **B** satisfying P is the image of a reaction in **E**;

and similarly for transitions.

Sorts and predicates

Expressivity of decomposable predicates

Theorem

A predicate P is decomposable iff there exists a set Φ of unwanted morphism s.t. $P(f)$ precisely if whenever $f = f' \circ \psi \circ f''$, then $\psi \notin \Phi$.

In words, a predicate P is decomposable iff for some set Φ , P characterizes the morphisms that are not factored by a morphism of Φ .

For our example, the set Φ is the singleton $m|m$.

Here is a non-composable predicate: “if f contains an m , it must also contain a b ”. This predicate holds for $m|b$, but not for m .

We have defined **sortings** (or typings) for Leifer & Milner's reactive systems. Building on Jensen's work, we have found, for any decomposable predicate P , a sorting which respects P , maintains semantics, and maintains congruence properties.

In the near future, we hope to lift this work to bigraphs, which are nearly, but not quite, reactive systems.

Thank you!

Questions?