# Higher-order Contexts via Games and the Int-construction

Lars Birkedal
Mikkel Bundgaard
Søren Debois
Davide Grohmann
Thomas Hildebrandt

Copies may be obtained by contacting:

# Higher-order Contexts via Games and the Int-construction[*]

Lars Birkedal    Mikkel Bundgaard
Søren Debois    Thomas Hildebrandt
PLS Group
IT University of Copenhagen
{birkedal, mikkelbu, debois, hilde}@itu.dk

Davide Grohmann
Dept. of Mathematics and Computer Science
University of Udine
grohmann@dimi.uniud.it

## Abstract

Monoidal categories of acyclic graphs capture the notion of multi-hole context, pervasive in syntax and semantics. Milner's bigraphs is a recent example. We give a method for generalising such categories to monoidal *closed* categories of acyclic graphs. The method combines the Int-construction, lifting traced monoidal categories to compact closed ones; the recent formulation of sortings for reactive systems; and games for multiplicative linear logic. The closed categories obtained by our construction captures a notion of *higher-order* contexts. These encompass extensions to the traditional notion of context found in recent work on Milner's reactive systems and bigraphs. We demonstrate how technical devices employed in these extensions are in fact intrinsic to higher-order contexts. Finally, we use the method to construct higher-order bigraphs, recovering directed bigraphs as a limited instance.

## 1 Introduction

Contexts lie at the very heart of both the syntax and semantics of computations. In concurrency theory, processes are studied by observing their interplay with other processes; their computational context. Sewell's derivation of transitions from reactions [31], Milner's action calculi [23], and Leifer-Milner's seminal reactive systems [20] are all based on the observation, that the semantic notion of

"computational context" is oftentimes simply the syntactic notion of "term context". That is, the meaning of a process is determined by the interplay between its reaction rules and its syntactic contexts. Concretely, labelled transitions are derived from reaction rules by defining that a process $p$ transitions to a process $q$ with label a context $C[-]$ iff there exist a reaction rule $(l, r)$ and a context $D[-]$ such that $C[p] \equiv D[l]$ and $q \equiv D[r]$. Leifer and Milner argued [20] that only minimal contexts $C[-]$ should be considered, and suggested relative pushouts (RPOs) and idem pushouts (IPOs) as notions of minimality.

Based on these ideas, Milner's bigraphs [25, 24] sought to supply a graph-based semantic foundation for both ubiquitous computing and contemporary process calculi, by providing terms and term contexts guaranteed to possess these RPOs. Bigraphs have successfully recovered the semantics of CCS [25], Petri nets [21], $\lambda$-calculus, Mobile Ambients and various $\pi$-calculi [17, 9], business processes [15], and the fusion calculus [14, 13]. However, the labelled transitions derived from the reaction rules are rarely as simple as the existing semantics: the bigraphical transitions systems for the $\lambda$-calculus, Mobile Ambients, and the $\pi$-calculi are all infinitely branching.

The bigraphical semantics of the fusion calculus and of business processes both required attenuation of the usual asymmetry between terms and their contexts: a term cannot effect a context in which it is inserted. For the fusion calculus, *directed* bigraphs [14] admits a succinct representation by allowing not only contexts to identify names in terms, but also terms to identify names in contexts. For business processes [15], *second-order* bigraph contexts enabled the definition of rule schema parametrised by contexts.

Outside the realm of bigraphs, [11] applies second-order term contexts to encodings of the $\lambda$-calculus as reactive systems. The standard observational equivalences for lazy and call-by-value reduction strategies are recovered as weak bisimilarity on reactive systems. In this work, finitely branching transition systems are obtained by the specification of reduction semantics using second-order contexts.

Finally, in [27, 28], higher-order contexts are crucial to the systematic derivation of structural operational semantics for various $\pi$-calculus and mobile ambients.

These papers all use higher-order contexts with a variety of limitations, and for a variety of reasons. Inspired by their apparent similarity and by their demonstrating the usefulness of higher-order contexts, we provide in the present paper a method for constructing general graphical higher-order contexts. Using this method, we generalise bigraphs to higher-order bigraphs. In contrast to (concrete) ordinary and directed bigraphs, however, the category of higher-order contexts does not in general possess RPOs.

The construction combines the Int-construction [18], the recent formulation of sortings [3, 4], and games for multiplicative linear logic [1, 16, 26]. The first step is to embed a symmetric monoidal category of acyclic graphs (i.e., contexts) into a traced symmetric monoidal category of cyclic graphs, on which we can apply the Int-construction to get a higher-order category of cyclic graphs. The next step, which is our key technical contribution, is the extrication of

higher-order *acyclic* graphs from higher-order cyclic graphs. For that, we use a sorting [3, 4] in combination with games from fully complete models of multiplicative linear logic. This approach is a triumph of semantics: Intuitively, the problem is solved the moment we start thinking about graphs as modelling the input/output flow of linear functions. But intuition is not enough; we must know formally what it *means* to be such an input/output flow. The game semantics for linear logic tells us that.

The result is a notion of graphical higher-order contexts, which indeed captures and explains notions of higher-order contexts found in the literature. As a special case, we get a generalisation of bigraphs to higher-order bigraphs, of which directed bigraphs fall out as a special case.

**Overview of the paper:** In Section 2, we introduce our categories of graphs, and outline the following technical development. In Section 3, we recall and expand on notions from game semantics. In Section 4, we construct the above-mentioned refinement. In Section 5, we give our category of higher-order contexts and prove that it is symmetric monoidal closed, that it embeds the original basic category, and that it is, in a sense to be made precise, faithful to the original basic category. In Section 6, we give an alternative characterisation of Milner's abstract bigraphs, then generalise those to the higher-order setting, and relate them to directed bigraphs. In Section 8, we analyse the use of higher-order contexts in [11, 27, 28]. In Section 9, we exemplify the gains in expressivity afforded by higher-order contexts using the reaction semantics of mobile ambients. Finally, in Section 10 we discuss open questions and conclude.

## 2 First-order Trees & Graphs

In this section, we introduce the basic categories of trees and graphs that we shall later generalise to the higher-order case. Afterwards, we give an in-depth overview of the technical development.

Note that our category of graphs is not the category of graphs and graph morphisms known from, e.g., the study of graph rewriting. Rather, it has graphs as morphisms, like Milner's place graphs [25], or like "graphs with interfaces" from borrowed context graph rewriting [12]. We could alternatively have arranged these graphs as categories of cospans [29].

**Definition 1.** A *graph with an interface* is a graph $(V + X, E)$, where $(V + X)$ are the *nodes* and $E \subseteq (V + X)^2$ the *edges*. We call the nodes of $V$ *internal nodes*, the nodes of $X$ the *interface nodes*. We consider such graphs identical if they differ only in the choice of $V$: $(V + X, E)$ and $(V' + X, E)$ are equivalent if there exists a bijection $V \simeq V'$ such that the obvious extension $V + X \simeq V' + X$ is in fact a graph isomorphism. We refer to equivalence classes of graphs with interfaces as *abstract graphs*.

We work with abstract graphs for simplicity; to avoid working with the bicategory of graphs where composition is not strictly associative.

3

**Notation** To avoid littering the presentation with injections of disjoint sums, we adopt the following conventions. (1) When $x \in X$ and there is an obvious inclusion $\alpha : X \hookrightarrow X'$, we use $x$ interchangeably as an element of $X$ and as an element of $X'$. (2) When $R \subseteq X \times X$ and there is an obvious inclusion $\alpha : X' \hookrightarrow X$, we define $R{\upharpoonright}_{X'} = \{(x, x') \mid (\alpha(x), \alpha(x')) \in R\}$. These "obvious inclusions" will invariably be inclusions of sums, e.g., $X + Z \hookrightarrow X + (Y + Z)$. In this case, the first convention dictates that if $y \in Y$, we write also $y \in X + (Y + Z)$ rather than the technically correct but practically noisy $\mathsf{inr}(\mathsf{inl}(y)) \in X + (Y + Z)$. For a relation $R \subseteq X \times X$, when there is an obvious inclusion $X \hookrightarrow Y$ we clearly also have an obvious (pointwise) inclusion $R \subseteq X \times X \hookrightarrow Y \times Y$. In this case, we use $R$ interchangeably as a relation on $X \times X$ and $Y \times Y$.

**Definition 2.** The category $G_0$ of *first-order graphs* has objects finite sets and morphisms $f : X \to Y$ abstract graphs $(V + (X + Y), E)$ such that each $x \in X$ has in-degree 0 and each $y \in Y$ has out-degree 0. Composition is path-composition, eliding the nodes in the interface: $(V + (X + Y), E) \circ (V' + (Y + Z), E') = ((V + V') + (X + Z), E|E')$, where $E|E' \subseteq ((V + V') + (X + Z))^2$ is defined by $E|E' = ((E' \circ E) \cup E \cup E'){\upharpoonright}_{((V+V')+(X+Z))^2}$. The identity $1_X : X \to X$ is the graph $(\emptyset + (X + X), \Delta_X)$ where $\Delta_X$ is the diagonal $\Delta_X = \{(x, x) \mid x \in X\}$.

(Adding the suppressed inclusions should convince you of the utility and harmlessness of the above conventions.)

**Definition 3.** Let $R_0$ be the category of finite sets and relations. Notice that $R_0$ is a sub-category of $G_0$.

Recall that a subcategory $C' \hookrightarrow C$ is *wide* if the inclusion functor is a bijection on objects.

**Definition 4.** The category of *first-order trees* $T_0$ is the wide sub-category of $G_0$ which has morphisms $f : X \to Y$ equivalence classes of graphs $(V + X + Y, E)$ s.t. for every node $v$ of $f$ there exists a unique $a \in X$ s.t. $v$ has a path to $a$; moreover, this path is itself unique.

Connoisseurs of bigraphs will hopefully recognise Milner's abstract place graphs [25, 24] *sans* controls.

**Lemma 1.** $G_0$ *and* $T_0$ *are both* symmetric monoidal *with tensor product graph juxtaposition: unit* $\emptyset$, *object concatenation* $X \otimes Y = X + Y$, *and morphism concatenation*

$$(V + (X + Y), E) \otimes (V' + (X' + Y'), E') =$$
$$((V + V') + ((X + X') + (Y + Y')), E \cup E') \,.$$

$G_0$ *is* traced *with the trace "tying the knot" as follows. Write* $Z^1, Z^2$ *to distinguish the left (first) and right (second) occurrence of $Z$ in* $(V + V') + ((X + Z) + (Y + Z))$. *The trace is* $((V + V') + (X + Y), E^t)$ *where*

$$E^t = ((E \circ b^* \circ E) \cup E){\upharpoonright}_{((V+V')+(X+Y))^2}$$

*where* $b = E{\upharpoonright}_{Z^2 \times Z^1} \circ E{\upharpoonright}_{Z^1 \times Z^2}$, *and* $b^*$ *is the reflexive transitive closure of* $b$.

Notice that $T_0$ is not closed under this trace. Also notice that this trace makes $R_0$ a traced subcategory of $G_0$, with the trace on $R_0$ one of the standard ones of [18].

An alternative characterisation of $E^t$: For a $z \in Z$, write $z^1, z^2$ to distinguish elements of the left (first) and right (second) occurrence of $z$ in $(V + V') + ((X + Z) + (Y + Z))$. Then $(u, v) \in E^t \subseteq ((V + V') + (X + Y))^2$ iff $(u, v) \in E$ or there exists some sequence $z_{0 \leq i < n}$ over $Z$ s.t. $(u, z_0^2) \in E$, $(z_i^1, z_{i+1}^2) \in E$ for $i < n-1$, and $(z_{n-1}^l, v) \in E$.

**Definition 5.** For a traced monoidal category $C$, write $\mathrm{Int}(C)$ for the Int-construction applied to $C$. Following the nomenclature of [13, 14], we call the category $\mathrm{Int}(G_0)$ *polarised graphs* and $\mathrm{Int}(R_0)$ *polarised relations*.

**Lemma 2.** $\mathrm{Int}(G_0)$ *has objects pairs of finite sets* $A^+, A^-$ *and morphisms* $f : A^+, A^- \to B^+, B^-$ *graphs* $(V + A^+ + B^- + B^+ + A^-, E)$ *with each* $u \in A^+, B^-$ *of in-degree* $0$ *and each* $v \in B^+, A^-$ *of out-degree* $0$. *The composite* $f \circ g$ *of graphs* $(V + A^+ + B^- + B^+ + A^-, E)$ *and* $(V' + B^+ + C^- + B^+ + C^-, E')$ *is* $(V + V' + A^+ + C^- + A^+ + C^-, E \parallel E')$. *Here* $E \parallel E'$ *is given by*

$$E \parallel E' = ((E \cup E') \circ b^* \circ (E \cup E')) \cup (E \cup E') \restriction_{(V + V' + A^+ + A^- + C^+ + C^-)}, \quad (1)$$

*where* $b = E \restriction_{B^- \times B^+} \cup E' \restriction_{B^+ \times B^-}$.

The composition deserves a comment. The relation $b$ relates nodes $u, v$ in the shared interface $B^+, B^-$ if $v$ is reachable from $u$. In the definition of $E \parallel E'$, the subterm $(E \cup E') \circ b \circ (E \cup E')$ says that a $u$ in $V$, $V'$, $A^+$, or $C^-$ is related to anything you can find by taking $u$ across to the shared interface $B^+, B^-$ following $E$ or $E'$, chasing around that interface for a while, then again following $E$ or $E'$ to find some $v$ in $V$, $V'$, $A^-$ or $C^+$. (Sometimes, you will end up in the shared interface instead, but because of the outermost restriction, these "too short trips" will not be visible in the end result.) Finally, the subterm $E \cup E'$ at the far right says that if $E$ send some $v \in A^+, V$ back to $A^-$ without crossing the shared interface, then so does the composite; similarly for $E'$. This is the straightforward generalisation of the composition in $\mathrm{Int}(R_0)$ as described in [18]. Indeed, $\mathrm{Int}(R_0)$ is a traced subcategory of $\mathrm{Int}(G_0)$, and clearly $T_0 \hookrightarrow G_0 \hookrightarrow \mathrm{Int}(G_0)$ by the standard adding of the empty set for the negative components of objects.

**Overview of the technical development.** We are to find a symmetric monoidal closed category which contains $T_0$ as a sub-category. We would like to do that by defining a trace for $T_0$ and then applying the Int-construction of [18]. This approach fails, however, because any potential notion of trace for $T_0$ that we can think of introduces cycles, thus taking us into $G_0$. Perhaps unsurprisingly, it appears you cannot have both a trace and acyclicity.

Thus, we proceed as follows. Using that $G_0$ does have a trace, we use the Int-construction to obtain a compact closed category $\mathrm{Int}(G_0)$. This category has morphisms $f : A^+, A^- \to B^+, B^-$ where $f$ is in fact a morphism $f : A^+ \otimes B^- \to B^+ \otimes A^-$ of $G_0$. Obviously, because $T_0 \hookrightarrow G_0$, some of these morphisms $f$ will

in fact be morphisms of $T_0$. Now, if only we could find a symmetric monoidal closed sub-category $T$ of $\mathrm{Int}(G_0)$ such that $T_0 \hookrightarrow T$ and every morphism of $T$ were in fact a morphism of $T_0$, we would have found our desired category $T$ of higher-order contexts.

Alas, we have been unable to find such a subcategory of $\mathrm{Int}(G_0)$. Instead, we construct $T$ as a a subcategory of a *refinement* of $\mathrm{Int}(G_0)$. Finding such a refinement is an instance of the following more general problem. Given a category $C$ and a property $P$ of the morphisms of $C$, with $P$ not closed under composition, construct a category $X$ which has as objects a refinement of the objects of $C$, and as morphisms the morphisms of $C$ satisfying $P$. That is, there must exist a faithful functor $F : X \to C$, surjective on objects, for which the homsets $C(F(a), F(b))$ are the morphisms satisfying $P$. Some of us studied this problem in the context of bigraphs and reactive systems [3, 4, 10], motivated by its pervasiveness in applications of bigraphs. In that context, such refinements are referred to as *sortings*. The trick is to come up with a refinement of the objects of $C$ which groups morphisms depending on what other morphisms they can be safely composed with.

In the present case, we refine the objects $A^+, A^-$ into the linear types $A$ (over $\otimes, \multimap, I$) such that $A^+$ and $A^-$ are precisely the postive and negative occurrences of atoms of the type $A$. To preserve the property of being acyclic under composition, we read a graph $f : A^+ \otimes B^- \to B^+ \otimes A^-$ as defining a dataflow between the input $A^+, B^-$ and the output $A^-, B^+$. If $f$ defines a valid dataflow for a linear function $A \multimap B$, it is a morphism $f : A \to B$. The property of being a dataflow for a linear function is closed under composition. To make this intuition precise, we turn to games models of linear logic; exactly how we will see in Section 3.

When we read a polarised graph as defining a dataflow, the actual structure of its internal nodes are irrelevant. The only important thing is which nodes in the interface has paths to which other nodes in the interface. Thus, for each graph $f : A^+, A^- \to B^+, B^-$ of $\mathrm{Int}(G_0)$ we consider the relation imposed by $f$ on $A^+ + B^- \times B^+ + A^-$. The latter is a morphism of $\mathrm{Int}(R_0)$, and in fact, we define a functor $p : \mathrm{Int}(G_0) \to \mathrm{Int}(R_0)$.

Because the dataflow defined by a polarised graph is in fact its image under $p$, we can construct the above-mentioned sorting on $\mathrm{Int}(R_0)$, and obtain the sought after category of higher-order trees $T$ as a subcategory of the pullback shown in the following diagram. We call the sorting functor $F$ (for "Flow") which is responsible for picking out the valid data flows in $\mathrm{Int}(R_0)$. We write also $F$ for the category that is the domain of $F$.

$$
\begin{array}{ccc}
T & \longrightarrow & F \\
\downarrow & & \downarrow \\
\mathrm{Int}(G_0) & \xrightarrow{p} & \mathrm{Int}(R_0)
\end{array}
$$

That was the overview of the technical development. We conclude this section by defining the path functor.

**Definition 6.** The *path-functor* $p : \text{Int}(G_0) \to \text{Int}(R_0)$ is the identity on objects, and takes a graph $(V + A^+ + A^- + B^+ + B^-, E)$ to $E^*\!\restriction_{A^+ + A^- + B^+ + B^-}$.

# 3 Games & Types

In this section, we introduce a weakening of *fair games* [16]. We are interested in games only as means of defining valid dataflows, so we could have used instead other game models of $\otimes, \multimap$. Units are needed only to reverse polarity.

We choose fair games for the following property: If $\sigma$ is a total strategy for $A \multimap B$, then $\sigma\!\restriction_A$ and $\sigma\!\restriction_B$ are total strategies for $A$ and $B$, respectively (Proposition 2 below). Our model dispenses with the requirement of [16] that the set of maximal positions of a game includes plays where player begins. We stress that we use these games exclusively to define valid dataflows; we have not investigated their properties as a model of linear logic.

**Definition 7.** Fix a denumerable set $\mathcal{X}$ of literals. The *linear types* are the strings defined by the following grammar.

$$A, B ::= I \mid x \mid A \otimes B \mid A \multimap B, \tag{2}$$

where $x$ ranges over $\mathcal{X}$. An (occurrence of an) *atom* of a type $A$ is a path from the root of the syntax tree for $A$ to a literal. We write $|A|$ for the set of atoms of $A$, and $A^+$ and $A^-$ for the set of positively and negatively occurring atoms, respectively. Formally: $|I| = \emptyset$, $|x| = \{\star\}$, and $|A \otimes B| = |A \multimap B| = |A| + |B|$.

When $A$ contains no literal $x \in \mathcal{X}$ more than once, we do not need to distinguish between literals and atoms. We assume that the set of all possible atoms is a subset of $\mathcal{X}$.

**Notation** For sequences $s, t$, we write the concatenation of $s$ and $t$ simply $st$. If $s$ is a sequence over $X$ and $X' \subseteq X$, we write $s \restriction X'$ for the subsequence of $s$ containing only elements from $X'$. We write $s \restriction X', Y'$ rather than $s \restriction (X' \cup Y')$. When sequences are known to be in fact permutations — which will almost always be the case — we say that $x$ occurs before $y$ in $s$ to mean the obvious: that there exists possibly empty sequences $s', s'', s'''$ s.t. $s = s'xs''ys'''$. We consider sequences ordered under the prefix ordering. Finally, for partial orders $X, Y$, we write $X \sqsubseteq Y$ if there exists a monotone map $X \to Y$.

As mentioned, the following notion of game is a weakening of the notion of *fair game* from [16].

**Definition 8.** A *game* is a triple $(M, \lambda, F)$ of

- a set $M$ of at least two moves;

- a *labelling function* $\lambda : M \to \{\mathsf{P}, \mathsf{O}\}$ telling us whether a move is made by the Player $\mathsf{P}$ or the Opponent $\mathsf{O}$; and

7

- a non-empty anti-chain $F$ of even-length sequences of alternately labelled moves, all beginning with an $\mathsf{O}$-move; the *maximal plays*.

We write $\overline{\lambda}$ for the dual to $\lambda$: $\overline{\lambda}(m) = \mathsf{P}$ if $\lambda(m) = \mathsf{O}$ and $\mathsf{O}$ otherwise. The positions or plays of the game are simply the prefixes of the elements of $F$. A finite, alternately-labelled sequence of moves is thus a valid position iff it can be extended to a maximal play.

Given games $A, B$, the **tensor game** $A \otimes B$ has moves $M_A + M_B$, labelling function $[\lambda_A, \lambda_B]$, and maximal plays finite alternately-labelled sequences $s$ over $M_A + M_B$ beginning with an $\mathsf{O}$-move such that $s \upharpoonright A \in F_A$ and $s \upharpoonright B \in F_B$. It is easy to see that in such maximal plays, $\mathsf{O}$ decides which sub-game is currently played: if two consecutive moves $mm'$ of a play stems from different components (i.e., one from $A$, one from $B$), then $m'$ is an $\mathsf{O}$-move. The plays of the tensor game are characterised as follows.

**Lemma 3.** *Let $A$ and $B$ be games, and let $s$ be a finite alternately-labelled sequence of moves over $M_A + M_B$ beginning with an $\mathsf{O}$-move. Then $s$ is a play of $A \otimes B$ iff $s \upharpoonright A \in P_A$ and $s \upharpoonright B \in P_B$.*

*Proof.* Suppose $s$ is such a sequence (the other direction is trivial). First note that there exists possibly empty sequences $u, v$ such that $(s \upharpoonright A)u \in F_A$ and $(s \upharpoonright B)v \in F_B$. Assume wlog that the last move of $s$ is in $A$. If it is a $\mathsf{P}$-move, then $suv$ is readily seen to be alternately-labelled; clearly $suv \in F_{A \otimes B}$. If instead it is a $\mathsf{O}$-move, then clearly $u = mu'$ for some $\mathsf{P}$-move $m$ in $A$. The sequence $smvu$ is again readily seen to be alternately-labelled; clearly $suv \in F_{A \otimes B}$. $\qquad\square$

The **linear implication game** $A \multimap B$ has again moves $M_A + M_B$, labelling function $[\overline{\lambda}_A, \lambda_B]$, and maximal plays finite alternately-labelled sequences over $M_A + M_B$ beginning with an $\mathsf{O}$-move such that $s \upharpoonright A \in F_A$ and $s \upharpoonright B \in F_B$. Dually to the tensor game, it is necessarily $\mathsf{P}$ who switches components in the linear implication game. The positions of the linear implication game are characterised as follows.

**Lemma 4.** *Let $A$ and $B$ be games, and let $s$ be a finite alternately-labelled sequence of moves over $M_A + M_B$ beginning with an $\mathsf{O}$-move. Then $s$ is a play of $A \multimap B$ iff (1) $s \upharpoonright A \in P_A$ and $s \upharpoonright B \in P_B$ and (2) if $s \upharpoonright B \in F_B$ then also $s \upharpoonright A \in F_A$.*

*Proof.* Suppose first that $s \in P_{A \multimap B}$; we must prove that $s$ satisfies (1) and (2). For (1) note that because $s$ is a play of $A \multimap B$ there exists some $s \leq f$ such that $f$ is a maximal play of $A \multimap B$. But by definition $f \upharpoonright A \in F_A$ so $s \upharpoonright A \leq f \upharpoonright A \in F_A$; similarly for $s \upharpoonright B$. For (2) assume for a contradiction that $s \upharpoonright B \in F_B$ but $s \upharpoonright A \notin F_A$. Then $s = s'ms''$ where $m$ is the last $B$-move in $s$. Because $s'm \upharpoonright B = s \upharpoonright B \in F_B$ the move $m$ must be a $\mathsf{P}$-move into $s''$ must begin with an $\mathsf{O}$-move in $A$. But only $\mathsf{P}$ can switch sub-game, so $\mathsf{O}$ has no moves at $s'm$. It follows that $s''$ must be empty and $s'm$ must itself be maximal because it cannot be extended. But $s'm \upharpoonright A \notin F_A$; contradiction.

Suppose now that $s$ satisfies (1) and (2). We must show that there exists a $l \in F_{A \multimap B}$ with $s \leq l$. By (1) there exists $u, v$ s.t. $(s \restriction A)u \in F_A$ and $(s \restriction B)v \in F_B$. If $s \restriction B$ then by (2) also $s \restriction A$ so $s$ is itself maximal and we're done. So suppose $s \restriction B \notin F_B$. Assume first $s \restriction A \in F_A$. If $s$ ends with an O-move then $sv$ is easily seen to be alternately-labelled; clearly $sv \in F_{A \multimap B}$. If instead $s$ ends with a P-move, that move must be in $B$; otherwise $s \restriction A$ has odd-length and cannot be maximal. Again $sv$ is easily seen to be alternately-labelled; and clearly $sv \in F_{A \multimap B}$. Assume finally that neither $s \restriction A \in F_A$ nor $s \restriction B \in F_B$. If $s$ ends with an O-move in $B$, $suv$ is readily verified to be alternately-labelled, so $suv \in F_{A \multimap B}$. If $s$ ends with a P-move in $B$ then because maximal plays have even length, $v = mv'$ for some O-move $m$ of $B$ and $v'$ nonempty and beginning with a P-move, whence $smuv'$ can be verified to be alternately-labelled; clearly $smuv' \in F_{A \multimap B}$. If $s$ ends with a O-move in $A$, that is, $s = s'm$, $m \in M_A$, and $\lambda_G(m) = $ O, then $suv$ is readily seen to be alternately-labelled; clearly $suv \in F_{A \multimap B}$. Finally, if $s$ ends with a P-move in $A$, that is, $s = s'm$, $m \in M_A$, and $\lambda_G(m) = $ P, we again find $suv$ alternately-labelled; clearly $suv \in F_{A \multimap B}$. $\qquad\square$

A strategy is what one would expect. Formally, a **P-strategy** for a game $G$ is a non-empty, prefix-closed subset $\sigma \subseteq P_G$ of the plays of $G$, s.t. for any even-length $s \in \sigma$ if $sm \in P_G$ then $sm \in \sigma$. An **O-strategy** is like a P-strategy, except we substitute "odd" for "even". A **total strategy** is one which has an answer to every move the other player might make; that is, a P strategy is total iff for any odd-length non-maximal $s \in \sigma$ there exists a move $m$ such the $sm \in \sigma$. Likewise for O, substituting "even" for "odd". Notice that the intersection of a total P-strategy and a total O-strategy is (the prefix-closure of) a maximum play $G$.

If $A, B, C$ are games, then strategies $\sigma$ for $A \multimap B$ and $\tau$ for $B \multimap C$ can be composed to form a strategy $\tau \circ \sigma$ for $A \multimap C$. Such a composition essentially works as follows. First observe that the game $A \multimap C$ must necessarily start with a move in $C$ (by definition, every play of every game must start with a move by O, so every play of $A$ starts with a move by O; but the linear implication game inverts the labelling function for $A$). By the same token, play in $A \multimap B$ and $B \multimap C$ must begin in $B$ and $C$, respectively. Now, $\tau \circ \sigma$ simply follows $\tau$; when $\tau$ stipulates a move in $B$, we use that move as the beginning of a play in $A \multimap B$ following $\sigma$. From this point on, whenever somebody plays a move in $B$, we feed this move into the other strategy. The following Lemma characterises this notion of composition of strategies at the level of sequences of moves.

**Lemma 5.** *For games $A_1, A_2, A_3$, define $\mathcal{F}(A_1, A_2, A_3)$ to be the set of finite sequences $s$ of moves from $M_{A_1} + M_{A_2} + M_{A_3}$ s.t. for any pair of consecutive moves $mm'$ in $s$, if $m \in M_{A_i}$ and $m' \in M_{A_j}$ then $|i - j| \leq 1$.*

*Let $\sigma$ be a strategy for the game $A \multimap B$ and $\tau$ be a strategy for the game $B \multimap C$. Then $\tau \circ \sigma$ is defined as follows as a strategy for $A \multimap C$.*

$$\tau \circ \sigma = \{s \restriction A, C \mid s \in \mathcal{F}(A, B, C) \land s \restriction A, B \in \sigma \land s \restriction B, C \in \tau\} \qquad (3)$$

*Composition preserves totality of strategies.*

*Proof.* Take $S = \{s \in \mathcal{F}(A, B, C) \mid s \restriction A, B \in \sigma \wedge s \restriction B, C \in \tau\}$. We must prove that $\tau \circ \sigma$ is a non-empty, prefix-closed subset of $P_{A \multimap B}$, and closed under all one-move continuations at even length. $\tau \circ \sigma$ is clearly non-empty and prefix-closed. To see that it is a subset of $P_{A \multimap B}$, suppose $s \in \tau \circ \sigma$; assume $s = \bar{s} \restriction A, C$. By Lemma 4, it is sufficient to prove that $s$ begins with an O-move, that it is alternately-labelled, that $s \restriction A \in P_A$ and $s \restriction C \in P_C$, and that if $s \restriction C \in F_C$ then also $s \restriction A \in F_A$. First, $s$ begins with an O-move. Observe that $\bar{s}$ cannot begin with a move of $B$, because if it did, either $\bar{s} \restriction A, B$ or $\bar{s} \restriction B, C$ would begin with a P-move. It cannot begin with an $A$-move, because then so would $\bar{s} \restriction A, B$. It follows that it must begin with a $C$-move which must be an O-move, because $\bar{s} \restriction B, C \in P_{B \multimap C}$. Next, $s$ is alternately-labelled. Suppose $uv$ adjacent in $s$. If $u, v$ both in, say, $A$, then we have $s \restriction A = (\bar{s} \restriction A, C) \restriction A \in P_A$ whence $u, v$ alternate. So assume wlog $u \in M_A$ and $v \in M_C$. Then there is some subsequence $u, b_1, \ldots, b_n, v$ of $\bar{s}$; because $\bar{s} \restriction A, B \in P_{A \multimap B}$ the switching condition yields $u$ an O-move, similarly because $\bar{s} \restriction B, C \in P_{B \multimap C}$ the switching condition yields $v$ a P-move. Thus, $s$ is alternately-labelled. That $s \restriction A \in P_A$ and $s \restriction C \in P_C$ follows from $s \restriction A = (\bar{s} \restriction A, C) \restriction A \in P_A$; similarly for $C$. Finally, using Lemma 4 part (2) extensively, if $s \restriction C \in F_C$ then $(\bar{s} \restriction B, C) \restriction C \in F_C$ so also $(\bar{s} \restriction B, C) \restriction B \in F_B$; but then also $(\bar{s} \restriction A, B) \restriction B \in F_B$, whence $s \restriction A = (\bar{s} \restriction A, B) \restriction A \in F_A$. Now, we show that $\tau \circ \sigma$ is closed under one-move extentions at even-length positions; suppose $s \in \tau \circ \sigma$ has even length, and suppose wlog that the last move of $s$ is a move of $A$. We must prove that for any move $m$ available to O, $sm \in \tau \circ \sigma$. But this is immediate from the observation that by the switching condition, O has available only moves of $A$. Finally, we prove that composition preserves totality of strategies. Suppose $s \in \tau \circ \sigma$ is odd-length. Assume wlog that the last move $u$ of $s$ is the last move of $A$. If both $\tau$ and $\sigma$ are total then $\sigma$ must prescribe a response in either $A$ or $B$; if it is in $A$, we are done, so suppose it is a move $b_1$ in $B$. Then $\tau$ prescribes a response to $\bar{s}b \restriction B, C$. If this response is in $C$, we are done. Otherwise, we have obtained a $b_2$ in $B$ which we feed back to $\sigma$. And so forth. Because games are finite and the game $B \multimap C$ cannot end with a $B$-move, this process must terminate at which point we have obtained the requisite response. $\square$

So far, we have closely followed [16]. We now move to our specific application. First, the **atomic game** is the game $(\{!, ?\}, [? \mapsto \mathsf{O}, ! \mapsto \mathsf{P}], \{?!\})$. We intend atomic games to model interaction points in interfaces. For instance, the game on the object $\{a\}$ will be the atomic game. The play

$$
\begin{array}{ll}
\mathrm{a} & \\
? & \mathrm{O} \\
! & \mathrm{P}
\end{array}
$$

should intuitively be understood as the opponent (a graph with domain $\{a\}$) asking for input on $a$; the player (a graph with codomain $\{a\}$) then providing that input. Next, the **unit game** is simply the atomic game. We now have unit, atomic, tensor, and linear implication games; we associate a game with each type of Definition 7 in the obvious way. From this point on, we shall in

general conflate types and games, e.g., we shall treat $a \multimap b$ interchangeably as a type and as a game. It is instructive to consider the possible maximal plays of $a \multimap b$ and $a \otimes b$. There are only the following three.

| a | $\multimap$ | b | | a | $\otimes$ | b | | a | $\otimes$ | b | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ? | | | | ? | | ? | | | O |
| ? | | | | | | ! | | ! | | | P |
| ! | | | | ? | | | | | | ? | O |
| | | ! | | ! | | | | | | ! | P |

For the remainder of this section, we establish various properties of these games that we will need later on. Most importantly, the above-mentioned Proposition 2.

**Definition 9.** Games $A$ and $B$ are *isomorphic games* iff there exists a bijection $\gamma : M_A \to M_B$ of the atoms of $A$ and $B$ that commutes with the labelling functions and preserves and reflects maximal plays.

This notion of isomorphism is the same one you would get from the obvious category of games and game homomorphisms. Observe that $\gamma$ extends (pointwise) to a bijection on strategies.

**Lemma 6.** *Let $A$ be a type.*

1. *Each maximal play of $A$ includes every move of $A$.*

2. *There exists a total $\mathsf{P}$- and a total $\mathsf{O}$-strategy for $A$.*

3. *Suppose $A$ contains no occurences of the unit $I$. Then $A$ has moves $M_A \simeq \Sigma_{a \in |A|} M_a$ where each $M_a$ is the moves of the atomic game.*

4. *Suppose $A$ contains no occurences of the unit $I$, and let $s$ be an alternately-labelled sequence over $M_A \simeq \Sigma_{a \in |A|} M_a$. Then $s \in F_A$ iff for each $a \in |A|$ we have $s \restriction M_a \in F_a$.*

*Proof.* Trivial induction. $\square$

Both (3), (4) above and the following Proposition generalise straightforwardly to include also types with units. Part (3) justifies the following notational convention. For a type $A$, we write the moves of the sub-game associated with the atom $a \in |A|$ simply $!_a, ?_a$.

The following Proposition will be useful in proving that various categories are symmetric monoidal closed. For instance, it is helpful in establishing bijection between homsets $A \otimes B \to C$ and $B \to A \multimap C$.

**Proposition 1.** *Let $A$ and $B$ be types containing no occurrences of the unit $I$. If there is a bijection $\gamma : |A| \simeq |B|$ which preserves polarities, then $A$ and $B$ are isomorphic games.*

*Proof.* Extend $\gamma$ to a bijection $\hat{\gamma} : M_A \simeq \Sigma_{a \in |A|} M_a \simeq \Sigma_{b \in |B|} M_b \simeq M_b$ by taking $\hat{\gamma}(!_a) = !_{\gamma(a)}$ and $\hat{\gamma}(?_a) = ?_{\gamma(a)}$. We must show $\hat{\gamma}$ commutes with the labelling functions and preserves and respects maximal plays. For the labelling functions, observe that $\lambda_A(!_a) = \mathsf{O}$ iff $a$ occurs positively in $A$ iff $\gamma(a)$ occurs positively in $B$ iff $\lambda_B(!_{\gamma(a)}) = \mathsf{O}$ iff $\lambda_B(\hat{\gamma}(!_a)) = \mathsf{O}$. The remaining three cases are similar. For the maximum plays, we find using Lemma 6 that $s \in F_A$ iff for each $a \in |A|$ we have $s \upharpoonright M_a \in F_a$ iff for each $a \in |A|$ we have $\hat{\gamma}(s) \upharpoonright M_{\gamma(a)} \in F_{\gamma(a)}$ iff $\hat{\gamma}(s) \in F_B$. $\qquad\square$

The following Proposition is crucial to elevating the intuition about graphs-as-models-of-dataflow to a proof that composition of our higher-order contexts really are acyclic (proof of Lemma 7).

**Proposition 2.** *Let $\sigma$ be a total $\mathsf{P}$-strategy for a game $G$, and let $A$ be a sub-game of $G$. If $A$ occurs positively in $G$ then $\sigma \upharpoonright A$ (where $\sigma \upharpoonright A = \{s \upharpoonright A \mid s \in \sigma\}$) is a total $\mathsf{P}$-strategy for $A$; if instead $A$ occurs negatively in $G$ then $\sigma \upharpoonright A$ is a total $\mathsf{O}$-strategy for $A$.*

*Proof.* First note that if $s$ is a play of $G$ then $s \upharpoonright A$ is a play of $A$ (induction on $G$). It follows that every $s \in \sigma \upharpoonright A$ is indeed a play of $A$. Thus, we need to verify that $\sigma \upharpoonright A$ is non-empty, prefix-closed, includes all possible moves at even-length (odd-length) positions, and that it is complete. It is non-empty because $\sigma$ is total. If $t'$ is a prefix of $t \in \sigma \upharpoonright A$ then $t$ is a subsequence of some $s \in \sigma$; but $\sigma$ is prefix-closed, so there must exist a prefix $s'$ of $s$ s.t. $s' \upharpoonright A = t'$.

Before we move on, we make the following observations (both seen by easy induction). (1) When $A$ occurs positively in $G$ then the polarity of the moves of $A$ is the same in $A$ and $G$ (i.e., $\lambda_A = (\lambda_G \upharpoonright M_A)$), and when $A$ occurs negatively in $G$, the polarities are opposite (i.e., $\overline{\lambda}_A = (\lambda_G \upharpoonright M_A)$). (2) In any play $s$ of $G$, the first $A$-move $m$ in $s$ has $\lambda_G(m) = \mathsf{O}$ if $A$ occurs positively in $G$, $\lambda_G(m) = \mathsf{P}$ otherwise.

Now suppose that $A$ is a sub-formula of $G$ and consider some sequence $t = t_1, \ldots, t_n \in \sigma \upharpoonright A$. Necessarily $t = s \upharpoonright A$ for some $s \in \sigma$; we may assume that $t_n$ is in fact also the last move of $s$. Suppose first that $A$ occurs positively in $G$ and $n$ is even. By the above observations $\lambda_A(t_1) = \lambda_G(t_1) = \mathsf{O}$; because $n$ even and $t$ a play of $A$ we find $\lambda_A(t_n) = \mathsf{P}$, so also $\lambda_G(t_n) = \lambda_A(t_n) = \mathsf{P}$. But then $s$ has even length, so because $\sigma$ is a strategy, for every $m \in M_A \subseteq M_G$ s.t. $sm \in P_G$ also $sm \in \sigma$, whence $tm \in \sigma \upharpoonright A$. If instead $n$ is odd, we must find $a \in M_A$ s.t. $t_1, \ldots, t_n, a \in \sigma \upharpoonright A$. Because $\sigma$ is total there exists $s'$ s.t. $ss' \in F_G$. By Lemma 6 $ss'$ includes every move of $G$; thus, there exists a prefix $s''a$ of $s'$ where $s''$ contains no move of $A$. Hence, $t_1, \ldots, t_n, a \in \sigma \upharpoonright A$.

Suppose now instead that $A$ occurs negatively in $G$ and that $n$ is odd. This time we find $\lambda_G(t_1) = \mathsf{P}$ so $\lambda_A(t_1) = \overline{\lambda}_G(t_1) = \mathsf{O}$; because $n$ is odd $\lambda_A(t_n) = \mathsf{O}$ so $\lambda_G(t_n) = \overline{\lambda}_A(t_n) = \mathsf{P}$. Again we have that $s$ has even length, so we find again that because $\sigma$ is a strategy, for every $m \in M_A \subseteq M_G$ s.t. $sm \in P_G$ also $sm \in \sigma$, whence $tm \in \sigma \upharpoonright A$. Finally, if $n$ is even, we observe again that by totality of $\sigma$ there exists $s'$ s.t. $ss' \in F_G$ whence $s'$ has some prefix containing exactly one move $a \in M_A$, so $t_1, \ldots, t_n, a \in \sigma \upharpoonright A$. $\qquad\square$

# 4 Higher-order Total Functions

We want to view the relations in $\text{Int}(G_0)$ as descriptions of dataflows. But which relations $R \subseteq A^+ + B^- \times B^+ + A^-$ describe valid dataflows? The category of graph contexts we come from is $T_0$ of trees. The dataflows corresponding to paths in the graphs must corresponds to paths in trees. This means that the relation $R$ must be *deterministic*, as a tree contains at most one path between any two nodes, and it must be *total* because every node in a tree has a path to the root. Thus, we must refine $R$ into a category of *total functions*.

The sorting $F$ refines the objects $A^+, A^-$ of $\text{Int}(G_0)$ into types $A$ with positive and negative (occurrences of) atoms $A^+, A^-$, respectively. In $F$, there is a relation $R : A \to B$ iff $R \subseteq A^+ + B^- \times B^+ + A^-$ describes a valid dataflow for a function $A \multimap B$. We use our games to formalise the phrase "describes a valid dataflow for a function".

First notice that for a type $A$, any play of $A$ induces a linear order on the atoms $|A|$ of $A$: the order in which !-moves are played. A strategy $\sigma$ thus defines a partial order on $|A|$ by taking $a \le b$ iff $!_a$ is played before $!_b$ in every maximal play of $\sigma$.

**Definition 10.** Let $A$ be a type and let $\sigma$ be a strategy for $A$. We define a partial order $\le_\sigma$ on $|A|$ by $a \le_\sigma b$ iff for every maximal $s \in \sigma$ we find $!_a$ occuring before $!_b$.

Similarly, a partial function $f : X \to Y$ induces (by its reflexive closure) a partial order on $X + Y$. In the sequel, we shall consider both strategies and such functions partial orderings when doing so is convenient.

A total function $f : A^+ + B^- \to B^+ + A^-$ will be a morphism $f : A \to B$ of $\text{dom}(F)$ iff there exists a total strategy $\sigma$ for $A \multimap B$ s.t. $f \sqsubseteq \sigma$. Intuitively, if $f$ takes some $x \in A^+ + B^-$ to a $f(x) \in B^+ + A^-$, it means that $f$ when read as a dataflow takes an input on $x$ and propagates that input to an output on $f(x)$. The condition that there exists some $\sigma$ with $f \sqsubseteq \sigma$ means that input on $f$ is always available to $f$ before output must be delivered on $f(x)$. We define $\text{dom}(F)$ and $F$ formally below; first, we prove that composition preserves totality.

**Lemma 7.** *Let $A, B, C$ be types, let $f : A^+ + B^- \to B^+ + A^-$ and $g : B^+ + C^- \to C^+ + B^-$ be total functions, and let $\sigma, \tau$ be total strategies for $A \multimap B$ and $B \multimap C$, respectively, with $f \sqsubseteq \sigma$ and $g \sqsubseteq \tau$. Then (1) $g \circ f$, where "$\circ$" is composition in $\text{Int}(R_0)$, is a total function, and (2) $g \circ f \sqsubseteq \tau \circ \sigma$.*

*Proof.* (1) Suppose for a contradiction that it is not. Then there exists a sequence $b_1, \ldots, b_n \in |B|$ s.t. $f(b_1) = b_2, \ldots, g(b_n) = b_1$ or $g(b_1) = b_2, \ldots, f(b_n) = b_1$. Assume the former (the latter case is analogous). Because $f \sqsubseteq \sigma$ and $g \sqsubseteq \tau$, we have

$$b_1 \le_\sigma b_2 \le_\tau b_3 \le_\sigma \cdots \le_\tau b_{n-1} \le_\sigma b_n \le_\tau b_1 \ .$$

But by Proposition 2, $\sigma \restriction B$ is a total P-strategy for $B$ and $\tau \restriction B$ is a total O-strategy for $B$, whence their intersection defines a non-empty set of maximal

plays of $B$. Each of these plays is a linear order on $|B|$ which respects both $\leq_\sigma$ and $\leq_\tau$; contradiction.

(2) Consider $x, y \in |A| + |B| + |C|$ s.t. $(g \circ f)(x) = y$. Recall the definition (3) of $\tau \circ \sigma$. Suppose $s$ is a maximal play in $\tau \circ \sigma$. There must be some $s' \in \mathcal{F}(A, B, C)$ with $s' \upharpoonright A, B \in \sigma$, with $s' \upharpoonright B, C \in \tau$ and with $s = s' \upharpoonright A, C$. If $f(x) = y$ then necessarily $x, y \in |A|$; because $f \sqsubseteq \sigma$ we have $x \leq_\sigma y$. The moves $!_x, !_y$ both occur in $s$, hence also in $s'$ and in $s \upharpoonright A, B$. But $s \upharpoonright A, B$ is a play of $A \multimap B$, hence a prefix of a a maximal such play, so because $x \leq_\sigma y$ we must have $!_x$ occuring before $!_y$ in $s$. The case $g(x) = y$ is symmetric. Suppose neither $f(x) = y$ nor $g(x) = y$. Assume $x \in |A|$ and $y \in |C|$ (the symmetric case is, well, symmetric). There must exist some sequence $b_1, \ldots, b_n \in |B|$ s.t.

$$f(x) = b_1, g(b_1) = b_2, \ldots, f(b_{n-1}) = b_n, g(b_n) = y . \tag{4}$$

We must prove that $!_x, !_y$ both occur in $s$ and in that order. They occur by maximality of $s$. By (4), $g \sqsubseteq \tau$ and $s' \upharpoonright B, C \in \tau$, we must have $!_{b_n}, y$ occurring in that order in $s' \upharpoonright B, C$; and hence in $s$. Similarly by (4), $f \sqsubseteq \sigma$ and $s \upharpoonright A, B \in \sigma$, we must have $!_{b_{n-1}}, !_{b_n}$ occurring in that order in $s' \upharpoonright A, B$; and hence in $s$. And so forth. Altogether, we've established that $!_x, !_{b_1}, \ldots, !_{b_n}, !_y$ all occur in $s$ in that order, whence $!_x, !_y$ both occur and in that order in $s \upharpoonright A, C$. □

We give the category $F$ at the domain of the sorting.

**Definition 11.** The category $F$ of *higher-order total functions* (valid data flows) has objects $A$ as given by Definition 7; it has morphisms $f : A \to B$ whenever $f : A^+, A^- \to B^+, B^-$ is a total function of $\text{Int}(R_0)$, and there exists a total strategy $\sigma$ for $A \multimap B$ s.t. $f \sqsubseteq \sigma$; and it has composition as in $\text{Int}(R_0)$. The *sorting functor* takes $A$ to $A^+, A^- \in \text{Int}(R_0)$; and $f : A \to B$ to $f : A^+, A^- \to B^+, B^- \in \text{Int}(R_0)$. NB: We write $F$ for both the category and the sorting functor.

As an example, here are two total functions $f : \{a\}^+, \emptyset \to \{x\}^+, \{y\}^-$ and $g : \{x\}^+, \{y\}^- \to \emptyset, \emptyset$ of $\text{Int}(R_0)$; defined by $f(a) = x, f(y) = x$; and $g(x) = y$.
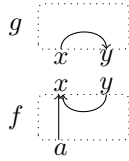


Clearly, the composite $(g \circ f)(a) = \bot$ in $\text{Int}(R_0)$. In $F$, however, we have refined the object $\{x\}^+, \{y\}^-$ into (among others) $(x \multimap y) \multimap I$ and $y \multimap x$. Only $f$ is a valid dataflow for $(y \multimap x)$, whereas only $g$ is a valid dataflow for $(x \multimap y) \multimap I$. To see this, we must prove that $f$ admits a total strategy for $a \multimap (y \multimap x)$, but not for $a \multimap ((x \multimap y) \multimap I)$; and similarly for $g$. Let us look at the sub-games on the object in the middle.

```
y    —o   x        (x    —o   y)    —o   I
          ?                              ?    O
?                          ?                  P
!                  ?                          O
     !             !                          P
                         !                    O
                              !    P
```

We see that both games have only one maximal play, hence only one total strategy. The unique total strategy for $y \multimap x$ must play $!_y$ before $!_x$. So that type can be the codomain of $f$, which respects that ordering, but not the domain of $g$, which does not. Conversely for the unique total strategy for $(x \multimap y) \multimap I$.

# 5 Higher-order Contexts

**Definition 12.** Let $F^*$ be the pullback of $F$ along $p$.



The category $T \hookrightarrow F^*$ of *higher-order contexts* is the wide subcategory of $F^*$ which includes a graph $(V + A^+ + B^- + B^+ + A^-, E)$ iff for each $v \in V$ there exists a unique $x \in B^+ + A^-$ s.t. $v$ has a path to $x$; and this path is itself unique.

**Proposition 3.** *The category $T$ has objects linear types as in Definition 7; it has morphisms $f : A \to B$ graphs $(V + A^+ + B^- + B^+ + A^-, E)$ such that*

1. *each $u \in A^+, B^-$ has in-degree 0 and each $v \in B^+, A^-$ has out-degree 0;*

2. *for each $u \in A^+, B^-, V$ there exists a unique $v \in B^+, A^-$ s.t. there is a path from $u$ to $v$; moreover, this path is unique;*

3. *there exists a total strategy $\sigma$ for $A \multimap B$ s.t. if $u \in A^+ + B^-$, $v \in B^+ + A^-$, and $v$ reachable from $u$, then in every maximal play of $\sigma$, $!_u$ occurs before $!_v$.*

*Composition is as in* $\mathrm{Int}(G_0)$.

We now return to the requirements from Section 2: $T$ must embed $T_0$, its morphisms must have underlying morphisms in $T_0$, and it must be symmetric monoidal closed.

**Theorem 1.** *The following holds.*

1. *The category $T_0$ of trees is a full subcategory of the category $T$ of higher-order contexts, by the assignment $(V + (A + B), E) : A \to B \hookrightarrow (V + A + \emptyset + \emptyset + B, E) : \bigotimes_{a \in A} a \to \bigotimes_{b \in B} b$.*

2. *Every morphism $f : A \multimap B$ of $T$ is in fact a morphism $f : |A|^+ \otimes |B|^- \to |B|^+ \otimes |A|^-$ of $T_0$.*

3. *All of $F^*$, $F$, and $T$ are symmetric monoidal closed.*

*Proof.* 1. The inclusion (11) is the standard inclusion $T \hookrightarrow G$ [18]. By Lemma 3, it is sufficient to prove that there exists a total strategy for $\bigotimes_{a \in A} a \multimap \bigotimes_{b \in B} b$ which plays any $!_a$ before any $!_b$. By Lemma 6, P has both a total strategy $\sigma$ for $\bigotimes_{a \in A} a$ and a total strategy $\tau$ for $\bigotimes_{b \in B} b$. We stipulate the following total strategy for P. By the switching condition, we can answer O's initial move (which must be some ? in $\bigotimes_{b \in B} b$) by following $\sigma$ until every move of $\bigotimes_{a \in A} a$ has been played; only then we switch to following $\tau$, which will eventually lead us having played all moves in $\bigotimes_{a \in A} a \multimap \bigotimes_{b \in B} b$. This total strategy clearly sees every move of $\bigotimes_{a \in A} a$ played before it allows any !-move of $\bigotimes_{b \in B} b$.

2. Immediate from Proposition 3.

3. Let $A, B$ be objects of $F^*$. Their exponential is $A \multimap B$. We define their evaluation map $\eta : A \otimes (A \multimap B) \to B$ as follows. We must give a graph $(V + |A| + |A| + |B| + |B|, E)$; we will give a graph with no internal nodes, so $V$ will be empty. The graph $\eta$ is the graph underlying the morphism $1_A \otimes 1_B$ (up to isomorphisms on the disjoint sums); by Proposition 1 the games $A \otimes B \multimap A \otimes B$ and $A \otimes (A \multimap B) \multimap B$ theare isomorphic, whence the duplicated copycat strategy witnessing $\eta : 1_A \otimes 1_B$ being a morphism also witnesses $\eta : A \otimes (A \multimap B) \to B$ being a morphism. If $f : A \otimes B \to C$ is some graph $(V + (|A| + |B|) + |C|, E)$, its curried counterpart $\Delta(f) : A \to B \multimap C$ is the same graph up to the isomorphism $V + (|A| + |B|) + |C| \simeq V + |A| + (|B| + |C|)$; again by Proposition 1, the total strategy witnessing that the graph $f$ is a morphism also witnesses that $\Delta(f)$ is. Clearly $\Delta(f)$ is the unique graph such that $f = \eta_{A,B} \circ (\Delta(f) \otimes 1_B)$. Thus $F^*$ is symmetric monoidal closed. Observing that $F$ is actually a subcategory of $F^*$, that is, $F \hookrightarrow T \hookrightarrow F^*$; that every $\eta_{A,B}$ is in fact a morphism of $F$; and that both $F$ and $T$ are closed under $\Delta$, we see that also $F, T$ are symmetric monoidal closed. $\square$

Part (2) of this Theorem justifies the name "higher-order contexts" for $T$: each morphism in it is built from trees and trees only. The sorting $F^* \to G$ has in a sense cut away the morphisms of $G_0$ that are not also morphisms of $T_0$.

We conclude the development of $T$ by remarking that, up to isos in $G$, category $T_0$ is the pullback of $T \to G$ along the inclusion of $G_0$ into $G$.

$$
\begin{array}{ccc}
T_0 & \hookrightarrow & T \\
\uparrow & \lrcorner & \downarrow \\
\downarrow & & \downarrow \\
G_0 & \hookrightarrow & G
\end{array}
$$

We remark that $T$ does not possess RPOs: a counterexample can be found in Appendix A. We discuss RPOs and the derivation of transitions in Section 10.

# 6  Higher-order Bigraphs

In this section we give an alternative characterisation of Milner's bigraphs [25, 24] and construct higher-order bigraphs. We proceed as in the preceding section: First, we define a traced category of cyclic bigraphs $C_0$ and recover bigraphs $B_0$ as a subcategory $B_0 \hookrightarrow C_0$. Then, we characterise $\mathrm{Int}(C_0)$, define a path-functor $p : \mathrm{Int}(C_0) \to \mathrm{Int}(R_0) \times \mathrm{Int}(R_0)$, and a sorting $F \times F \to \mathrm{Int}(R_0) \times \mathrm{Int}(R_0)$. Finally, we obtain higher-order bigraphs $H$ as a sub-category of the pullback $(F \times F)^*$ of the bigraph path-functor $q$ and that sorting.

$$
\begin{array}{ccc}
H \hookrightarrow (F \times F)^* & \longrightarrow & F \times F \\
\downarrow & & \downarrow \\
\mathrm{Int}(C_0) & \xrightarrow{\quad q \quad} & \mathrm{Int}(R_0) \times \mathrm{Int}(R_0)
\end{array}
$$

You may think that we are about to labouriously reconstruct the entire development of the previous sections. In fact, we are; only not "labouriously", as we will be able to characterise bigraphs in terms of the category $T_0$. Once we have that characterisation, there is really little left to be done. First, Milner's notion of signature.

**Definition 13.** A signature $\Sigma = (K, \alpha)$ is a set $K$ of controls and a map $\alpha : K \to \mathbb{N}$.

The category $T_0 \hookrightarrow G_0$ is essentially Milner's abstract place graphs. We now define $P_0$ intended to be likewise essentially abstract link graphs ($P_0$ for pre–link-graphs).

**Definition 14.** The category $P_0$ of *pre–link-graphs* is the wide subcategory of $G_0$ that has morphisms equivalence classes of graphs $(V + (X + Y), E)$ where $V$ can be partitioned $V = P \cup H$ into sets of ports $P$ and hyperedges $H$ s.t. each $p \in P$ has in-degree 0 and out-degree 1, and each $h \in H$ has out-degree 0.

Intuitively, $E$ devolves to become the link map. $P_0$ is a subcategory because composition in $G_0$ preserves $V$ and because in each graph $(V + (X + Y), E)$ of $G_0$, each $x \in X$ has out-degree 1. By the same reasoning, we have the following Lemma.

**Lemma 8.** $P_0$ *is symmetric monoidal and traced, with the same tensor and trace as* $G_0$.

We now define bigraphs $B_0$ and their generalisation $C_0$ to possibly cyclic bigraphs. The distinction between ordinary and cyclic place graphs comes down to whether the place component is chosen from $G_0$ (cyclic) or $T_0$ (ordinary).

**Definition 15.** A pre–cyclic-bigraph comprises:

1. (Place component) A graph $(V + (X + Y), E)$ which is (a representative of) a morphism of $G_0$.

2. (Link component) A graph $((P \cup H) + (X' + Y'), E')$ which is (a representative of) a morphism of $P_0$.

3. (Glue) A labelling function $\ell : V \to K$ and a glueing relation $R \subseteq V \times \mathbb{N} \times P$. The relation must be functional in both $V$ and $P$, and must satisfy that for each $v \in V$ and each $0 \le i < \alpha(\ell(v))$ there exists $p \in P$ with $(v, i, p) \in R$.

Intuitively, $(v, i, p) \in R$ signifies that $p$ is the $i$'th port of $v$. Two pre-bigraphs are equivalent iff their place and link components are (Definition 1) and the involved bijections commutes with the labelling function $\ell$ and the glueing relation $R$. The category $C_0$ of *cyclic bigraphs* has the same objects as $G_0 \times P_0$; morphisms equivalence classes of pre–cyclic-bigraphs; and composition as in $G_0 \times P_0$ with the obvious disjoint sum of labelling functions and glueing relations. The category $B_0$ of *bigraphs* is the subcategory $B_0 \hookrightarrow C_0$ in which the place component of each morphism is in fact a morphism of $T_0$.

**Proposition 4.** *For a given signature $\Sigma$, the category $B_0$ is equivalent to Milner's abstract bigraphs over $\Sigma$.*

*Proof.* Write $A$ for Milner's category of abstract bigraphs over $\Sigma$. We construct a functor $F : B_0 \to A$ which is full, faithful, and surjective. For objects we define $F(X, Y) = (|X|, Y)$. For morphisms, let $(f, g, \ell, R) : X \to Y$ be a representative. First, the place graph image. Suppose the place component $f$ is $f = (V + (X + Y), E)$. We take $F^p(f, g, \ell, R) = (V, E, \ell)$, that is, we keep the nodes $V$, use $E$ as the parent map "as is" up to the isomorphism

$$E \subseteq (V + X) \times (V + Y) \simeq |X| \uplus V \times V \uplus |Y| \tag{5}$$

where, e.g., $|X|$ is considered an ordinal; then $\ell$ is in fact a control map. Because $f$ is a forest, $E$ is an acyclic function. Suppose next that the link component $g$ is $g = (P \cup H + (X' + Y'), E')$. We take $F^l(f, g, \ell, R) = (V, H, \ell, E')$, that is, we keep the nodes (from the place component), the edges of the link graph is the hyper-edges $H$ of the link component, $\ell$ is again a control map, and again $E'$ is in fact the link map up to the following isomorphism.

$$E' \subseteq (P + X') \times (H + Y') \simeq X' \uplus \Sigma_{v \in V} \alpha(\ell(v)) \times H \uplus Y' \tag{6}$$

This isomorphism is justified by the observation that $P \simeq \Sigma_{v \in V} \alpha(\ell(v))$. Clearly $(f, g, \ell, R)$ is equivalent to $(f', g', \ell', R')$ if and only if $F(f, g, \ell, R)$ is support

equivalent to $F(f', g', \ell', R')$. We leave the verification that $F$ commutes with composition as an exercise to the reader. Thus $F$ is a faithful functor. To see that it is also full, it is sufficient to note that $T_0$ (Definition 4) and $P_0$ (Definition 14) admit edge-relations sufficiently flexible for any place and link graph, respectively. But this is immediate by the above comments justifying (5) and (6). Finally, $F$ is obviously surjective on objects. $\qquad\square$

Notice the forgetful functor $u : C_0 \to G_0 \times P_0$. Using that functor, the following Lemma is almost immediate.

**Lemma 9.** $C_0$ *is traced symmetric monoidal, with the same tensor and trace as in $G_0 \times P_0$.*

*Proof.* It is straightforward to verify the symmetric monoidal structure. For the trace, it is sufficient to notice that the constituents $V$, $P$, and $H$ of the labelling function and the glueing relation are invariant under the traces of $G_0$ and $P_0$. $\qquad\square$

**Definition 16.** Define $\mathrm{Int}(C_0)$ and $\mathrm{Int}(u)$ as follows.

$$
\begin{array}{ccc}
\mathrm{Int}(C_0) & \xrightarrow{\quad\mathrm{Int}(u)\quad} & \mathrm{Int}(G_0 \times P_0) \\
\Big\uparrow & & \Big\uparrow \\
C_0 & \xrightarrow{\quad u \quad} & G_0 \times P_0
\end{array}
$$

Define the (lifted) path-functor $q : \mathrm{Int}(C_0) \to \mathrm{Int}(R_0) \times \mathrm{Int}(R_0)$ as follows.

$$
q : \mathrm{Int}(C_0) \xrightarrow{\mathrm{Int}(u)} \mathrm{Int}(G_0 \times P_0) \simeq \mathrm{Int}(G_0) \times \mathrm{Int}(P_0)
$$
$$
\hookrightarrow G \times G \xrightarrow{p \times p} \mathrm{Int}(R_0) \times \mathrm{Int}(R_0)
$$

Finally, define $(F \times F)^*$ as the pullback of $F \times F$ and $q$.

$$
\begin{array}{ccc}
(F \times F)^* & \longrightarrow & F \times F \\
\Big\downarrow & & \Big\downarrow \\
\mathrm{Int}(C_0) & \xrightarrow{\quad q \quad} & \mathrm{Int}(R_0) \times \mathrm{Int}(R_0)
\end{array}
$$

**Lemma 10.** *The category $(F \times F)^*$ has object pairs of types. It has morphisms $f : (A, B) \to (C, D)$ comprising the following constituents.*

1. *(Place component) A (representative of a) morphism $f : A \to B$ of $F^*$.*

19

2. *(Link component) A (representative of a) morphism $g : C \to D$ of $F^*$ s.t. $g : C^+ \otimes D^- \to D^+ \otimes C^-$ is a morphism of $P_0$.*

3. *(Glue) A labelling function $\ell : V_f \to K$ and a glueing relation $R \subseteq V_f \times \mathbb{N} \times P_g$ which is functional in both $V_f$ and $P_g$, and satisfies that for each $v \in V_f$ and each $0 \leq i < \alpha(\ell(v))$ there exists $p \in P_g$ with $(v, i, p) \in R$.*

*Composition is as in $\mathrm{Int}(C_0)$.*

*Proof.* Immediate from Definition 15 above, Definition 14, and Lemma 3. □

**Definition 17** (Higher-order bigraphs, $H$)**.** $H$ is the wide subcategory of $(F \times F)^*$ in which the place components are all (representatives of) morphisms of $T$.

**Theorem 2.** *$H$ is symmetric monoidal closed, $B_0$ is a subcategory of $H$, and every morphism of $H$ is in fact a morphism of $B_0$: every $f : A \multimap B$ of $H$ is a morphism $f : |A|^+ \otimes |B|^- \to |B|^+ \otimes |A|^-$ of $B_0$.*

*Proof sketch.* $F^* \times F^*$ symmetric monoidal closed follows from Theorem 1, part (3). Looking at the proof that theorem, we see that the evaluation maps are all in the wide subcategory $F \times F \hookrightarrow F^* \times F^*$. $H$ also contains $F \times F$ as a subcategory (with the empty labelling function and the empty glueing relation); clearly, the extra glueing structure in $H$ relative to $F^* \times F^*$ is irrelevant for the symmetric monoidal closed structure.

$B_0 \hookrightarrow H$ is immediate from $T_0 \hookrightarrow G$, $P_0 \hookrightarrow G$ and Theorem 1, part (1). □

# 7 Directed bigraphs

In this section, we will show that directed bigraphs can be characterised as a subcategory of higher-order bigraphs. First, we recall briefly the intuitive idea behind directed bigraphs: Names are not resources on their own, but only denote, or give access to, (abstract) resources (represented by "edges"). Names are exposed on the interfaces of systems, much like "channels". If a name on an interface is associated to a resource of the system, then the system is offering the resource to the outside; on the converse, a name not associated to any resource can be seen as a formal parameter of the system, that is the system is asking for an external resource through that name. Thus, we can discern a "resource request flow" which starts from control ports, goes through names on interfaces, and eventually terminates in edges.

**Definition 18.** The category $D$ of *directed bigraphs* is the subcategory $D \hookrightarrow H$ where all objects (pair of types) are in the following form:

$$(\bigotimes_{x \in X} x, ((\bigotimes_{y \in Y} y) \multimap I) \otimes \bigotimes_{z \in Z} z) \ .$$

Moreover, no morphism with such a domain links any $z$ to any $y$.

Notice that such kind of objects forces the place graph component to be "ordinary", i.e., it belongs to $T_0$ (in the sense of Theorem 1). Instead, the (directed) link graph has a higher-order like structure, but the inputs from the contexts cannot be mapped back as outputs for the contexts or vice versa. In other words, the "resource request flow" has always a fixed direction (i.e., ascending or descending) and it cannot be inverted.

**Proposition 5.** *For a given signature $\Sigma$, the category $D$ is equivalent to Grohmann-Miculan's abstract directed bigraphs [14, 13] over $\Sigma$.*

*Proof.* Write $DB$ for Grohmann-Miculan's category of abstract directed bigraphs over $\Sigma$. We construct a functor $F : D \to DB$ which is full, faithful, and surjective. For objects we define

$$F((\bigotimes_{x \in X} x, ((\bigotimes_{y \in Y} y) \multimap I) \otimes \bigotimes_{z \in Z} z)) = (|X|, (Y, Z)) \ .$$

For morphisms, let $(f, g, \ell, R)$ be a representative. First, the place graph image. For the fact that it belongs to $T_0$, the proof proceeds as in Proposition 4. Suppose next that the link component $g$ is $g = (P \cup H + (Y^+ + Z^- + Z^+ + Y^-), E')$. We take $F^l(f, g, \ell, R) = (V, H, \ell, E')$, that is, we keep the nodes (from the place component), the edges of the link graph is the hyper-edges $H$ of the link component, $\ell$ is the control map, and $E'$ is in fact the link map up to the following isomorphism.

$$E' \subseteq (P + Y^+ + Z^-) \times (H + Y^- + Z^+) \simeq$$
$$Y^+ \uplus Z^- \uplus \Sigma_{v \in V} \alpha(\ell(v)) \times H \uplus Y^- \uplus Z^+ \quad (7)$$

This isomorphism is justified by the observation that $P \simeq \Sigma_{v \in V} \alpha(\ell(v))$ and by the fact that mapping names in $Y$ to names in $Z$ is forbidden by hypothesis and the vice versa is forbidden by the considered types. Clearly $(f, g, \ell, R)$ equivalent to $(f', g', \ell', R')$ iff $F(f, g, \ell, R)$ support equivalent to $F(f', g', \ell', R')$. We leave the verification that $F$ commutes with composition as an exercise to the reader. Thus $F$ is a faithful functor. It is also full for the same arguments as in Proposition 4 and by the equation (7). Finally, $F$ is obviously surjective on objects. $\square$

# 8 Applications of Higher-order Contexts

In this section, we briefly analyse recent appearances of higher-order contexts in the literature. Presently, we trade rigour for readability; in Section 9, we shall apply the conclusions of this conceptual analysis rigorously when we give reduction semantics to a fragment of mobile ambients using the graphical higher-order model we present in subsequent sections.

In [11], Di Gianantonio, Honsell, and Lenisa investigate various encodings of the $\lambda$-calculus as a reactive system. As mentioned above, to recover the

standard observational equivalences as weak bisimilarity on reactive systems they crucially rely on the transition systems on these reactive systems being finitely branching. Such finiteness is at odds with the beta reduction rule

$$(\lambda x.M)N \rightarrow_\beta M\{N/x\}, \tag{8}$$

which is parametric in $M, N$. When we take seriously the paradigm that meaning (transitions) is found in the interaction between reaction rules in context, the term $(\lambda x.x)$ should have infinitely many transitions, namely for each of the infinitely many $N$ that yields a valid instance $(\lambda x.x)N$ of the above reduction rule. This difficulty is cleverly overcome by (among other things) extending the contexts considered to encompass also *second-order contexts*. Di Gianantonio, Honsell, and Lenisa then stipulate that each side of the $\beta$-reduction rule (8) above should be read not as a schema of infinitely many terms, but instead as a *single* function from contexts to contexts

$$(\lambda x.M)-_N \rightarrow_\beta M\{-_N/x\}$$

such that, say, $(\lambda x.M)-_N$ takes terms $N$ to the term $(\lambda x.M)N$. Under this notion of context, there is a least context enabling $\beta$-reduction for $(\lambda x.x)$: it is the second-order context $[]-_N$.

In [27, 28], Rathke and Sobociński give a systematic derivation of the transition semantics for the $\pi$-calculus and mobile ambients, respectively, starting from reduction semantics. In order to get an inductive transition semantics, they introduce higher-order contexts: for, e.g., ambients, they stipulate the following basic transition.

$$\mathsf{in}\ n.P \xrightarrow{\mathsf{in}\ n} \lambda XxY.n[x[P \mid X] \mid Y] \tag{9}$$

Notice how the right-hand side fits into a higher-order context, which can put in a suitable name $x$ and suitable ambients $X$ and $Y$. This way, Rathke and Sobociński avoid having to keep the missing parts $X, Y, x$ as part of the label $\mathsf{in}\ n$ by leaving their instantiation to the eventual insertion of the right-hand side into a (higher-order) context. (Note that on some level, this is essentially the same trick as the one employed by Di Gianantonio, Honsell, and Lenisa.)

This device is crucial in obtaining compositionality, as the semantics of larger terms can be obtained simply by application of the meaning of smaller ones. For instance, here is the rule defining the interplay of entry and parallel composition.

$$\frac{P \xrightarrow{\mathsf{in}\ n} T}{P \mid Q \xrightarrow{\mathsf{in}\ n} \lambda X.T(Q \mid X)} \tag{10}$$

Here, the application in $\lambda X.T(Q \mid X)$ allows the insertion of $Q$ *under* the ambient $x[\ldots]$ present somewhere in $T$ (see (9) above).

Rathke and Sobociński do not explicitly acknowledge their category of terms and contexts as a higher-order one, and view their use of $\beta$-reduction on the metalevel as a mere technical device [28, p. 467]. We would suggest that this ingenious use of lambdas is in fact more fundamental: They are working in a

22

category of higher-order contexts, and the reason that they arrive at a compositional semantics is that they can define the semantics of composited entities *by composition in that category.* In (10) above, the right-hand side $\lambda X.T(Q \mid X)$ is in fact the insertion of $T$ into the (higher-order) context $\lambda X.[](Q \mid X)$!

## 9    Mobile Ambients

In this section, we encode a fragment of Mobile Ambients (MA) as a reactive system on higher-order bigraphs $H$. We use only the place component and the labelling function, so you can alternatively think of the model as one on $T$, only with labelled graphs. While our fragment is too small to be of independent interest, it is large enough to demonstrate key benefits of higher-order contexts: we get ground reaction rules, and we can define reaction rules parameterised by contexts. We consider the following fragment of MA.

$$
\begin{array}{rcl}
P & ::= & \mathbf{0} \;\big|\; P \mid Q \;\big|\; n[\,P\,] \;\big|\; M.P \\
M & ::= & \mathtt{in}\ n \;\big|\; \mathtt{out}\ n \;\big|\; \mathtt{open}\ n
\end{array}
$$

In this fragment we have the standard process constructors: the inactive process, parallel composition, a named ambient, and prefixing with a capability. As usual, we will take terms up to a structural congruence, namely the monoid laws for $\mid, \mathbf{0}$. We will let $E$ range over evaluation contexts, defined as any context where the hole is not under a prefix, and we will write $E(P)$ for the insertion of the process $P$ into the evaluation context $E$.

We model each term constructor with a corresponding control in the signature (label), and model parallel composition by common parents. For simplicity of the encoding, we consider, e.g., $n[\,-\,]$ and $m[\,-\,]$ to be distinct term constructors ([17] demonstrates how to construct an encoding where constructors are parametric in names). An MA term or context is thus modelled as a graph, which is somewhat akin to the abstract syntax tree of that term or context.

The standard reduction semantics for MA is obtained by closing the following rules under evaluation contexts.

$$
\begin{array}{c}
n[\,\mathtt{in}\ m.P \mid Q\,] \mid m[\,R\,] \rightarrow_{\mathrm{MA}} m[\,n[\,P \mid Q\,] \mid R\,] \\
n[\,m[\,\mathtt{out}\ n.P \mid Q\,] \mid R\,] \rightarrow_{\mathrm{MA}} m[\,P \mid Q\,] \mid n[\,R\,] \\
\mathtt{open}\ n.P \mid n[\,Q\,] \rightarrow_{\mathrm{MA}} P \mid Q
\end{array}
\tag{11}
$$

The first reaction rule express that a "$\mathtt{in}\ m$"-capability will instruct the surrounding ambient to enter a sibling ambient named $m$. The "$\mathtt{out}\ n$"-capability instructs the surrounding ambient to exit its parent ambient named $n$. And the "$\mathtt{open}\ n$"-capability provides a way of dissolving the locality of an ambient named $n$ located at the same level.    Observe that these rules are parametric (non-ground), i.e., to get an instance of the first rule, you have to plug in processes for $P, Q$, and $R$.

It will be convenient to have a term language for the graphs of $H$. It is known that a symmetric monoidal closed category has linear $\lambda$-calculus as an

internal language [2, 22], but for the present example, we will just use the following suggestive notation. For brevity, we consider only the third rule. We give reaction rules as pairs of morphisms $f, g : I \to X$, where $f$ is the redex, and $g$ is the reactum.

$$\lambda P{:}a, Q{:}b.\ \texttt{open}\ n.P \mid n[\,Q\,] \longrightarrow \lambda P{:}a, Q{:}b.\ P \mid Q : c$$

The redex denotes the graph with nodes $\texttt{open}\ n$ and $n[\ ]$ containing interface nodes $P, Q$, respectively, to be instantiated by a context[1]. The redex and reactum in the reaction rule are both morphisms of $I \to ((a \otimes b) \multimap c)$ as witnessed by, e.g., the total strategy with the maximal play $?_c?_b!_b?_a!_a?_I!_I!_c$. The rule is ground $(I)$; and when the context provides two graphs $(a \otimes b)$, thus filling in the parameters, we will get a ground graph $(c)$. In contrast, the obvious reading of the original parametric rule of (11) is a first-order morphism $a \otimes b \to c$.

If we derive labelled transitions for the ambient $\texttt{open}\ n.\mathbf{0}$ by finding contexts enabling reaction using the standard paramteric first-order reaction rules, we would find infinitely many transitions of the following form.

$$\texttt{open}\ n.\mathbf{0} \xrightarrow{n[\,R\,]} \mathbf{0} \mid R$$

Using instead the higher-order reaction rules, we get higher-order labels. For instance, the ambient $\texttt{open}\ n.\mathbf{0}$ has the following transition[2].

$$\texttt{open}\ n.\mathbf{0} \xrightarrow{\lambda Q{:}b.n[\,Q\,]} \lambda Q : b.\ \mathbf{0} \mid Q$$

This use of higher-order reaction rules was introduced in [11], where it was instrumental in obtaining a finitely branching transition system.

To illustrate reaction rules parameterised by contexts, we extend MA with primitives $\texttt{send}\ n.P$ and $\texttt{recv}.Q$ for communication over location boundaries. These primitives are idealised versions of similar ones found in higher-order process calculi with locations [8, 30].

$$\texttt{send}\ n.P \mid n[\,E(\texttt{recv}.Q)\,] \to_{\text{MA}} n[\,E(P \mid Q)\,] \tag{12}$$

Here the process $P$ is communicated to a sub-process inside the ambient named $n$. This sub-process can reside arbitrary deeply inside the ambient, provided it is in an active context. Note that we abstract not only over processes $P$ and $Q$, but also over a context $E$. We can model this kind of communication using a higher-order context, as illustrated in Fig. 1 and denoted by the following expression.

$$\lambda Q{:}b, E{:}c \multimap d, P{:}a.\ \texttt{send}\ n.P \mid n[\,E\,(\texttt{recv}.Q)\,] : e$$
$$\longrightarrow \lambda Q{:}b, E{:}c \multimap d, P{:}a.\ n[\,E\,(P \mid Q)\,] : e$$

---

[1]Higher-order evaluation contexts can be defined from first-order evaluation contexts are using Theorem 1, part 2, or Theorem 2, part 2.

[2]There are many more transitions; we leave for future work (see next section) to determine whether, and in what sense, this one is minimal.
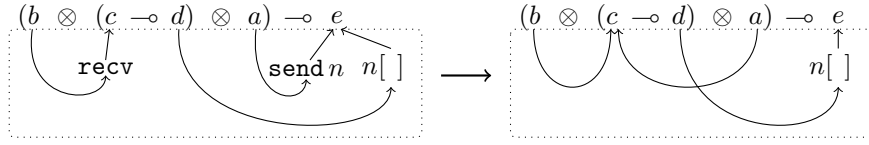
$$(b \otimes (c \multimap d) \otimes a) \multimap e \qquad\longrightarrow\qquad (b \otimes (c \multimap d) \otimes a) \multimap e$$

(redex: `recv`, `send` $n$, $n[\ ]$) $\longrightarrow$ (reactum: $n[\ ]$)

Figure 1: The reaction rule (12) as higher-order graphs.

These are morphisms of $I \to (b \otimes (c \multimap d) \otimes a) \multimap e$; it is straightforward to find a total strategy witnessing both redex and reactum being morphisms. Here is one.

$$I \quad \multimap \quad (b \quad \otimes \quad (c \quad \multimap \quad d) \quad \otimes \quad a) \quad \multimap \quad e$$

| $I$ | $b$ | $c$ | $d$ | $a$ | $e$ | | |
|---|---|---|---|---|---|---|---|
| | | | | | ? | O |
| | | | | ? | | P |
| | | | | ! | | O |
| | ? | | | | | P |
| | ! | | | | | O |
| | | | ? | | | P |
| | | ? | | | | O |
| | | ! | | | | P |
| | | | ! | | | O |
| ? | | | | | | P |
| ! | | | | | | O |
| | | | | ! | | P |

We relate the term and graph representations of this reaction rule. In the redex, the contents of $Q$ (the atom $b$) goes inside the node `recv`, which in turn is inside the hole of $E$ (the atom $c$). The node `recv` disappears in the reactum. More interestingly, in the redex, the contents of $P$ (the atom $a$) is inside the node `send` $n$ which in turn is inside the root of the rule (the atom $e$). In the reactum, the contents of $P$ will, in parallel with the contents of $Q$, be placed inside the hole of $E$.

This concludes our examples of higher-order reaction rules. We have seen how they admit both the replacement of parametric reaction rules with ground higher-order rules, yielding in turn higher-order labels [11], as well as the definition of reaction rules parameterised by contexts [8, 15].

## 10  Conclusion

We have given a method for generalising monoidal categories of acyclic graphs to monoidal closed such categories. We have applied this method to Milner's bigraphs, obtaining higher-order bigraphs. Finally, we have exemplified the benefits of higher-order contexts with the reaction semantics for mobile ambients.

The obvious next step is to use higher-order contexts for deriving transitions and bisimulation equivalences. Because of [27, 28], we are optimistic that reactive systems on closed categories may yield good equivalences.

However, as mentioned in Section 5, our category of higher-order contexts does not in general possess RPOs. Thus, there is a challenge in finding the right notion of minimal higher-order context. We see three potential lines of attack. (1) Find sub-categories of $T$ or $H$ which do possess RPOs. We are encouraged by the example of directed bigraphs [14, 13], which is such a subcategory, and of second-order contexts of [11], which possesses RPOs but is comprised by *ordered* labelled trees (that is, terms). Ideally, such a subcategory would be symmetric monoidal closed, a property enjoyed by neither of these examples. (2) Dispense with the restriction to minimal contexts and consider instead *saturated semantics* [5, 6, 7]. In saturated semantics one considers all contexts, not just the minimal ones. Saturated semantics have recovered certain semantics of $\pi$-calculi and Prolog and is thus known to provide good congruences in the first-order case. However, the only known tractable characterisation of saturated semantics requires the category in question to possess RPO-like structure. Thus, this line of attack requires either that we find sub-categories with sufficient structure, or that we find an alternative characterisation of saturated semantics. (3) Look for a notion of minimal context particular to the higher-order setting. This line of work would attempt to generalise the techniques employed in [27, 28], and, to a lesser extent, [19]. The latter work considers (possibly open) terms and parametric rewrite rules, generalising the IPO-notion of minimal context to encompass also a most general parameter. However, since the parameter and context are treated as two different entities in the labels, the labels can contain redundancy. We hope to avoid this problem: in our setting, the context provides both.

Derivations of transitions aside, the present work begets several other questions; we mention a few. Suppose our initial graphical model (presently $T_0$ or $B_0$) is known to have a term language (this is the situation for bigraphs). Is there a canonical extension of this term language to the higher-order extension? That is, is the method or construction given in the present paper in some sense a *free* construction? Is the sorting $F$ a closure sorting [4]? Can we capture the notion of a valid dataflow using games other than fair games, in particular, using the games of [26]? (It should be possible, but the present development falls apart: fair games are apparently the only ones admitting Proposition 2.) Finally, how does the present work apply to graph-rewriting formalisms other than bigraphs [29, 12]?

# References

[1] S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.

[2] P. N. Benton, G. M. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In M. Bezem and J. F. Groote, editors, *TLCA*, volume 664 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 1993.

[3] L. Birkedal, S. Debois, and T. Hildebrandt. Sortings for reactive systems. In *Proc. CONCUR'06*, volume 4137 of *LNCS*, pages 248–262. Springer, 2006.

[4] L. Birkedal, S. Debois, and T. Hildebrandt. On the construction of sorted reactive systems. In *Proc. CONCUR'08*, volume 5201 of *LNCS*, pages 218–232. Springer, 2008.

[5] F. Bonchi, B. König, and U. Montanari. Saturated semantics for reactive systems. In *Proc. LICS'6*, pages 69–80. IEEE Computer Society, 2006.

[6] F. Bonchi and U. Montanari. Coalgebraic models for reactive systems. In *Proc. CONCUR'07*, volume 4703 of *LNCS*, pages 364–379. Springer, 2007.

[7] F. Bonchi and U. Montanari. Symbolic semantics revisited. In *Proc. FoSSaCS'08*, volume 4962 of *LNCS*, pages 395–412. Springer, 2008.

[8] M. Bundgaard. *Semantics of Higher-Order Mobile Embedded Resources and Local Names*. PhD thesis, IT University of Copenhagen, 2007.

[9] M. Bundgaard and V. Sassone. Typed polyadic pi-calculus in bigraphs. In *Proc. PPDP'06*, pages 1–12. ACM Press, 2006.

[10] S. Debois. *Sortings and Bigraphs*. PhD thesis, IT University of Copenhagen, 2008.

[11] P. Di Gianantonio, F. Honsell, and M. Lenisa. RPO, second-order contexts, and lambda-calculus. In *Proc. FoSSaCS'08*, volume 4962 of *LNCS*, pages 334–349. Springer, 2008.

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

[13] D. Grohmann and M. Miculan. Directed bigraphs. In *Proc. MFPS'07*, volume 173 of *ENTCS*, pages 121–137. Elsevier, 2007.

[14] D. Grohmann and M. Miculan. Reactive systems over directed bigraphs. In *Proc. CONCUR'07*, volume 4703 of *LNCS*, pages 380–394. Springer, 2007.

[15] T. Hildebrandt, H. Niss, and M. Olsen. Formalising business process execution with bigraphs and Reactive XML. In *Proc. COORDINATION'06*, volume 4038 of *LNCS*, pages 113–129. Springer, 2006.

[16] J. M. E. Hyland and C.-H. L. Ong. Fair games and full completeness for multiplicative linear logic without the mix rule. Unpublished notes, 1993.

[17] O. H. Jensen. Mobile processes in bigraphs. Monograph available at `http://www.cl.cam.ac.uk/~rm135/Jensen-monograph.html`, 2006.

[18] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[19] B. Klin, V. Sassone, and P. Sobociński. Labels from reductions: Towards a general theory. In *Proc. CALCO'05*, volume 3629 of *LNCS*, pages 30–50. Springer, 2005.

[20] J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Proc. CONCUR'00*, volume 1877 of *LNCS*, pages 243–258. Springer, 2000.

[21] J. J. Leifer and R. Milner. Transition systems, link graphs and Petri nets. *Journal of Mathematical Structures in Computer Science*, 16(6):989–1047, 2006.

[22] I. Mackie, L. Román, and S. Abramsky. An internal language for autonomous categories. *Applied Categorical Structures*, 1(3):311–343, 1993.

[23] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, 1996.

[24] R. Milner. Bigraphical reactive systems. In *Proc. CONCUR'01*, volume 2154 of *LNCS*, pages 16–35. Springer, 2001.
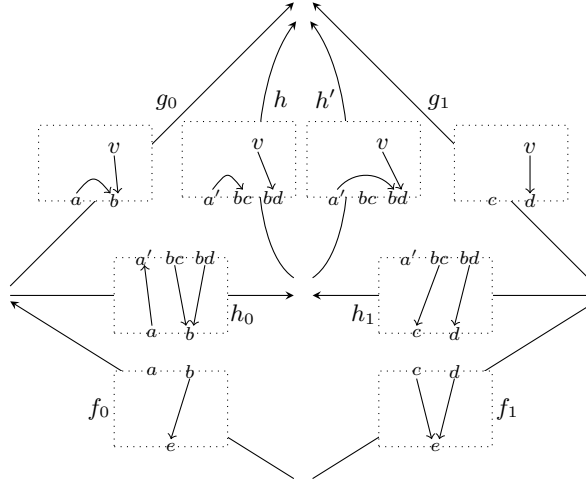
Figure 2: A counterexample.

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

[26] A. S. Murawski and C.-H. L. Ong. Exhausting strategies, joker games and full completeness for IMLL with unit. *Theoretical Computer Science*, 294(1-2):269–305, 2003.

[27] J. Rathke and P. Sobocinski. Deconstructing behavioural theories of mobility. In *IFIP TCS*, volume 273, pages 507–520. Springer, 2008.

[28] J. Rathke and P. Sobocinski. Deriving structural labelled transitions for mobile ambients. In *Proc. CONCUR'08*, volume 5201 of *LNCS*, pages 462–476. Springer, 2008.

[29] V. Sassone and P. Sobociński. Reactive systems over cospans. In *Proc. LICS'05*, pages 311–320. IEEE Computer Society Press, June 2005.

[30] A. Schmitt and J.-B. Stefani. The Kell calculus: A family of higher-order distributed process calculi. In *Proc. GC'04*, volume 3267 of *LNCS*, pages 146–178. Springer, 2004.

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

# A   Counterexample to RPOs in $T$

Consider the following span $f_0, f_1$:

$$f_0 = (\{a, b, e\}, \{(b, e)\}) : e \multimap I \to (a \multimap b) \multimap I$$
$$f_1 = (\{c, d, e\}, \{(c, e), (d, e)\}) : e \multimap I \to (c \otimes d) \multimap I$$

and its relative bound $g_0, g_1$:

$$g_0 = (\{v\} + \{a, b\}, \{(a, b), (v, b)\}) : (a \multimap b) \multimap I \to I$$
$$g_1 = (\{v\} + \{c, d\}, \{(v, d)\}) : (c \otimes d) \multimap I \to I.$$

They are both depicted in Figure 2.

Now, we can construct two RPO candidates: $(h_0, h_1, h)$ and $(h_0, h_1, h')$, they are defined as follows (see Figure 2):

$$h_0 = (\{a, b, a', bc, bd\}, \{(a, a'), (bc, b), (bd, b)\}) :$$
$$(a \multimap b) \multimap I \to (a' \multimap (bc \otimes bd)) \multimap I$$
$$h_1 = (\{c, d, a', bc, bd\}, \{(bc, c), (bd, d)\}) :$$
$$(c \otimes d) \multimap I \to (a' \multimap (bc \otimes bd)) \multimap I$$
$$h = (\{v\} + \{a', bc, bd\}, \{(a', bc), (v, bd)\}) :$$
$$(a' \multimap (bc \otimes bd)) \multimap I \to I$$
$$h' = (\{v\} + \{a', bc, bd\}, \{(a', bd), (v, bd)\}) :$$
$$(a' \multimap (bc \otimes bd)) \multimap I \to I$$

Let us suppose that there exists another candidate $(k_0, k_1, k)$ which is the real RPO. Then there must exist two (unique) mediating maps $j, j'$, such that

$$k = h \circ j \qquad\qquad k = h' \circ j'$$
$$h_0 = j \circ k_0 = j' \circ k_0 \qquad\qquad h_1 = j \circ k_1 = j' \circ k_1 \qquad (13)$$

First of all, notice that the node $v$ belongs to $k$, and its port has to be connected to a name $w$ in the inner interface of $k$, such name is linked to $b$ in $k_0$ and to $y$ in $k_1$. In this way we ensure that $g_0 = k \circ k_0$ and $g_1 = k \circ k_1$.

Next, notice that the idle name $a$ must be exported in the "middle interface" of the RPO, otherwise the construct triple, named $(t_0, t_1, t)$, is not an RPO:
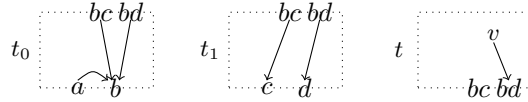


it is subsumed by $(h_0, h_1, h)$ using the following unique morphism:
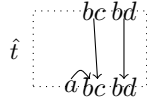


So, call $i$ the exported idle name in $k_1$.

Moreover, the downward names exported in the middle interface are exactly 2 (call them $p$ and $q$), in fact the number of possible combinations for accessing the common name $e$ are $|\{c, d\}| \times |\{b\}| = 2$. In particular, in $k_1$, one of this downward names (say $p$) is linked to $c$, instead the other one ($q$) to $d$. (It is

easy to find out some counterexamples if these properties do not hold, see the RPO construction for directed bigraphs.)

Now, to equate $k_1$ with $h_1$, we have to map in $j$ $bc$ to $p$, $bd$ to $q$, and $i$ to $a$. In practice, we have defined a renaming, which is an iso. Applying the same reasoning to $j'$, we obtain another iso.

For absurdity suppose that equations in (13) hold. From the fact that the RPOs are equal up-to iso, we can derive the absurdity that $k$ is "equal" to $h$ and $h'$ at the same time.