Synthesising Labelled Transitions and Operational Congruences in Reactive Systems, Part 2

James J. Leifer

N° 4395

March 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4395--FR+ENG

Synthesising Labelled Transitions and Operational Congruences in Reactive Systems, Part 2

James J. Leifer

Thème 1 — Réseaux et systèmes
Projet Moscova

Rapport de recherche n° 4395 — March 2002 — 48 pages

Abstract: This paper is the second in a series of two. It relies on its companion, Part 1, to motivate the central problem addressed by the series, namely: how to synthesise labelled transitions for reactive systems and how to prove congruence results for operational equivalences and preorders defined above those transitions. The purpose of this paper is (i) to show that the hypotheses required of functorial reactive systems from Part 1, for example the sliding properties of IPO (idem pushout) squares, are indeed satisfied for functors of a general form; (ii) to illustrate an example of a functorial reactive system based on Milner's action calculi, which satisfy the RPO (relative pushout) hypothesis required in the proofs of congruence from Part 1.

Key-words: labelled transition, structural congruence, bisimulation, action calculi, pushout

Synthèse de transitions étiquetées et de congruences opérationnelles pour les systèmes réactifs, deuxième partie

Résumé : Cet article forme le second d'une série. Le lecteur est renvoyé à l'article associé (première partie) pour un exposé de la motivation du problème auquel s'attaque cette série, à savoir comment synthétiser des transitions étiquetées pour des systèmes réactifs et comment prouver des résultats de congruence pour des équivalences et préordres opérationnels définis sur ces transitions. Le but de cet article est, premièrement, de montrer que les hypothèses de la définition des systèmes réactifs fonctoriels de première partie, par exemple les propriétés de glissement des diagrammes d'IPO ("idem pushout"), sont bien satisfaites pour des foncteurs d'une certaine forme générale ; deuxièmement, d'illustrer un exemple de système réactif fonctoriel basé sur les "action calculi" de Milner, qui vérifient les hypothèses de RPO ("relative pushout") des preuves de congruence de première partie.

Mots-clés : transition étiquetée, congruence structurelle, bisimulation, action calculi, somme amalgamée

Contents

1 Introduction

1.1 Motivation and goals

This paper is the second in a series of two. I rely on its companion [Lei02], referred to as Part 1 throughout, to motivate the central problem addressed by the series, namely: how to synthesise labelled transitions for reactive systems and how to prove congruence results for operational equivalences and preorders defined above those transitions. Thus, the present paper uses Part 1 as a base, sacrificing self-containedness in order to avoid repetition, though the key definitions are recapitulated here in Section 2.

The purpose of this paper is to twofold:

(i) to show that the hypotheses required of *functorial reactive systems* from Part 1 are indeed satisfied for functors of a general form;

(ii) to illustrate an example of a functorial reactive system, namely action graphs, which satisfy the RPO (relative pushout) hypothesis required in the proofs of congruence from Part 1.

Of the two, goal (i) is more critical since the techniques employed for generating the functor and proving the relevant properties are widely applicable. The argument does not fit neatly within standard category theory, though, and requires the use of *precategories* — which have a partial composition operator, unlike the total one of categories. It seems possible that this work could be recast by replacing precategories with bicategories. One of the primary reasons for presenting (i) in detail is to expose this potential recasting as a challenge to the (enriched) category theory community. This question is discussed in detail in Subsection 3.7.

For goal (ii), I introduce action graphs (a subclass of Milner's action calculi). The main motivation for defining these is to show that the properties required of them in the definition of a functorial reactive system follow smoothly from the work on goal (i). The proof of the existence of RPOs, by contrast, requires highly combinatorial manipulations of graph contexts and embeddings. For the sake of brevity, I omit this proof, pointing instead to its full treatment in my Ph.D. [Lei01b].

1.2 Outline

The outline of this paper is as follows:

Section 2: This section recapitulates the key definitions from Part 1 needed in the present paper.

Section 3: As mentioned above in (i) the proofs of congruence in Part 1 rest on some hypotheses about the functor $\mathcal{F}$. The most important one of these is that $\mathcal{F}$ allows an IPO square upstairs to "slide" so as to leave the $\mathcal{F}$-image of the square invariant. This section eases the burden of showing that these hypotheses are satisfied by giving an abstract way of generating a functor satisfying them from simpler data. The section starts by assuming the existence of a *well-supported precategory* $\mathbf{A}$, which is like a category but lacks some arrow compositions and has extra structure for extracting and renaming the *support* of an arrow. The motivation for this comes from the raw contexts in Section 4 for which composition of two contexts is not defined unless their node sets (supports) intersect trivially. I derive from $\mathbf{A}$ two categories and a functor between them. The upstairs category is formed from $\mathbf{A}$ by adding extra information to the objects, so as to make composition total. The downstairs category is formed by quotienting away the supports. The functor $\mathcal{F}$ maps arrows to their quotient equivalence classes. By reasoning abstractly, I show that $\mathcal{F}$ allows IPO sliding and has all of the other properties required of functorial reactive systems (see Definition 2.5). By instantiating these results, as in Section 4 for example, one gets "for free" a functorial reactive system provided that one starts with a well-supported precategory, a light burden.

Section 4: This section gives a significant example of a family of functorial reactive systems. The contexts are derived from closed, shallow action graphs, those with no nesting and no free names. Their graph structure includes forked and discarded wiring connecting interfaces on each node and on the inside and outside of each context. By instantiating the results of the preceding section, we construct a functorial reactive system with the following properties: the downstairs category does not distinguishes the occurrences of controls, as desired when modelling agents of a process calculus; the upstairs one does distinguish them, thus providing sufficient RPOs.

Section 5: The section sets out the main open questions, whose solutions are critical to the application of this work to a wide variety of process calculi.

1.3 Acknowledgements

I thank Robin Milner (my Ph.D. supervisor) for his inspiration and warmth. I learned more about the ways of doing research and the style of presenting it from him than anyone else. I had extensive and valuable discussions with Luca Cattani and Peter Sewell. Many colleagues provided suggestions, for which I am grateful: Tony Hoare, Martin Hyland, Philippa Gardner, Michael Norrish, Andy Pitts, John Power, Edmund Robinson, and Lucian Wischik

I was supported by the following funding sources: EPSRC Research Grant GR/L62290 and an NSF Graduate Research Fellowship.

$$\begin{tikzcd}
\cdot \arrow[r, "f_0"] \arrow[d, "f_1"'] & \cdot \arrow[d, "g_0"] \\
\cdot \arrow[r, "g_1"'] & m
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[rr, "f_0"] \arrow[dd, "f_1"'] & & \cdot \arrow[dd, "g_0"] \arrow[dl, "h_0"'] \\
& \cdot \arrow[dr, "h"] & \\
\cdot \arrow[ur, "h_1"] \arrow[rr, "g_1"'] & & m
\end{tikzcd}
\qquad
\begin{tikzcd}
\cdot \arrow[rrr, "f_0"] \arrow[dd, "f_1"'] & & & \cdot \arrow[ddd, "g_0"] \arrow[dll, "h_0"'] \arrow[dl, "h'_0"] \\
& \cdot \arrow[r, dotted, "k"] \arrow[ddrr, "h"'] & \cdot \arrow[ddr, "h'"] & \\
\cdot \arrow[ur, "h_1"] \arrow[urr, bend right, "h'_1"'] & & & \\
\cdot \arrow[u, phantom] \arrow[rrr, "g_1"'] & & & m
\end{tikzcd}$$

1(1) 1(2) 1(3)

Figure 1: Construction of an RPO

2 Recapitulation of key definitions and theorems

This section recalls the key definitions and theorems from Part 1 needed in this paper. The original motivation and explanation of each is not repeated, but can be found by following each pointer into Part 1.

Definition 2.1 (RPO — see Definition 2.4 in Part 1) In any category $\mathbf{C}$, consider a commuting square (Figure 1(1)) consisting of $g_0 f_0 = g_1 f_1$. An *RPO* is a triple h_0, h_1, h satisfying two properties:

commutation: $h_0 f_0 = h_1 f_1$ and $h h_i = g_i$ for $i = 0, 1$ (Figure 1(2));

universality: for any h'_0, h'_1, h' satisfying $h'_0 f_0 = h'_1 f_1$ and $h' h'_i = g_i$ for $i = 0, 1$, there exists a unique mediating arrow k such that $h' k = h$ and $k h_i = h'_i$ (Figure 1(3)). ■

Terminology: A triple, such as h'_0, h'_1, h' given above, that satisfies the commutation property, i.e. $h'_0 f_0 = h'_1 f_1$ and $h' h'_i = g_i$ for $i = 0, 1$, is often called a *candidate*. Thus an RPO triple is a candidate for which there is a unique mediating arrow from it to any other candidate.

Definition 2.2 (IPO — see Definition 2.5 in Part 1) The commuting square in Figure 1(1) is an IPO if the triple g_0, g_1, id_m is an RPO. ■

Proposition 2.3 (IPOs in a full subcategory — see Proposition 2.10 in Part 1) Let m, m' be objects of $\mathbf{C}$ and let $\mathbf{C}'$ be a full subcategory of $\mathbf{C}$ satisfying the following property:

$$\begin{tikzcd}
m \arrow[r, "f_0"] \arrow[d, "f_1"'] & \cdot \arrow[d, "g_0"] \\
\cdot \arrow[r, "g_1"'] & m'
\end{tikzcd}$$

2

$$\mathsf{obj}\,\mathbf{C}' \supseteq \{ n \in \mathsf{obj}\,\mathbf{C} \;\;/\;\; \exists h \in \mathbf{C}(m, n) \;\&\; \exists h' \in \mathbf{C}(n, m') \} \; .$$

Suppose the square in Figure 2 commutes, where $f_i, g_i \in \mathbf{C}'$ for $i = 0, 1$. Then the square is an IPO in $\mathbf{C}$ iff it is an IPO in $\mathbf{C}'$. ■

Definition 2.4 (reactive system— see Definition 2.1 in Part 1) A *reactive system* consists of a category $\mathbf{C}$ with added structure. We let m, n range over objects. $\mathbf{C}$ has the following extra components:

- a distinguished object 0 (not necessarily initial);
- a set of *reaction rules* called $\mathsf{Reacts} \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0, m)^2$, a relation containing pairs of agents with common codomain;
- a subcategory $\mathbf{D}$ of $\mathbf{C}$, whose arrows are the *reactive contexts*, with the property that $D_1 D_0 \in \mathbf{D}$ implies $D_1, D_0 \in \mathbf{D}$. ■

Notation: $a, b \in \mathbf{C}$ range over arrows with domain 0, called agents, and $C, D, F \in \mathbf{C}$ range over arbitrary arrows (contexts).

Definition 2.5 (functorial reactive system— see Definition 3.1 in Part 1) Let $\mathbf{C}$ be a reactive system. A *functorial reactive system over* $\mathbf{C}$ consists of a functor $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ which maps a distinguished object $\varepsilon \in \mathsf{obj}\,\hat{\mathbf{C}}$ to 0 (the distinguished object of $\mathbf{C}$) and which satisfies the following properties.

$\mathcal{F}$ **lifts agents:** for any $a : 0 \to m$ there exists $\mathtt{a} : \varepsilon \to \mathtt{m}$ such that $\mathcal{F}(\mathtt{a}) = a$.

$\mathcal{F}$ **creates isos:** if $\mathcal{F}(\mathtt{C})$ is an iso then $\mathtt{C}$ is an iso.

$\mathcal{F}$ **creates compositions:** if $\mathcal{F}(\mathtt{C}) = C_1 C_0$, there exist $\mathtt{C}_0, \mathtt{C}_1 \in \hat{\mathbf{C}}$ such that $\mathtt{C} = \mathtt{C}_1 \mathtt{C}_0$ and $\mathcal{F}(\mathtt{C}_i) = C_i$ for $i = 0, 1$.

$\mathcal{F}$ **allows IPO sliding:** for any IPO square as in Figure 3 and any arrow $\mathtt{F}'_0$ with $\mathcal{F}(\mathtt{F}'_0) = \mathcal{F}(\mathtt{F}')$ there exist $\mathtt{C}_0, \mathtt{C}'_0, \mathtt{F}_0$ forming an IPO square as in Figure 4 with

$$\begin{tikzcd} \cdot \arrow[r, "\mathtt{C}"] \arrow[d, "\mathtt{F}'"'] & \cdot \arrow[d, "\mathtt{F}"] \\ \cdot \arrow[r, "\mathtt{C}'"'] & \cdot \end{tikzcd} \quad 3 \qquad \begin{tikzcd} \cdot \arrow[r, "\mathtt{C}_0"] \arrow[d, "\mathtt{F}'_0"'] & \cdot \arrow[d, "\mathtt{F}_0"] \\ \cdot \arrow[r, "\mathtt{C}'_0"'] & \cdot \end{tikzcd} \quad 4$$

$$\mathcal{F}(\mathtt{C}_0) = \mathcal{F}(\mathtt{C}) \qquad \mathcal{F}(\mathtt{C}'_0) = \mathcal{F}(\mathtt{C}') \qquad \mathcal{F}(\mathtt{F}_0) = \mathcal{F}(\mathtt{F}) \; .$$ ■

Notation: Uppercase teletype characters denote arrows in $\hat{\mathbf{C}}$ and lowercase teletype characters $(\mathtt{a}, \mathtt{l}, \ldots)$ denote arrows with domain ε in $\hat{\mathbf{C}}$. The $\mathcal{F}$ images of these are agents in $\mathbf{C}$. The special domain requirement of $\mathtt{a}, \mathtt{l}, \ldots$ is left tacit throughout: thus $(\exists \mathtt{l} \in \hat{\mathbf{C}}.\ \ldots)$ means $(\exists \mathtt{l} \in \hat{\mathbf{C}}.\ \mathsf{Dom}\,\mathtt{l} = \varepsilon \;\&\; \ldots)$.

Definition 2.6 ($\mathcal{F}$ has all redex-RPOs— see Definition 3.4 in Part 1) A functorial reactive system $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ *has all redex-RPOs* if any square, such as in Figure 5, has an RPO, provided that $\mathcal{F}(\mathtt{D}) \in \mathbf{D}$ and that there exists $r \in \mathbf{C}$ such that $(\mathcal{F}(\mathtt{l}), r) \in \mathsf{Reacts}$.

$$\begin{tikzcd} \cdot \arrow[r, "\mathtt{a}"] \arrow[d, "\mathtt{l}"'] & \cdot \arrow[d, "\mathtt{F}"] \\ \cdot \arrow[r, "\mathtt{D}"'] & \cdot \end{tikzcd} \quad 5$$ ■

Proposition 2.7 (characterisation of $\longrightarrow\!\!\triangleright$ — see Proposition 3.2 in Part 1) $a \longrightarrow\!\!\triangleright a'$ iff there exist $\mathtt{a}, \mathtt{l}, \mathtt{D} \in \hat{\mathbf{C}}$ and $r \in \mathbf{C}$ such that $\mathtt{a} = \mathtt{D}\mathtt{l}$ and

$$a' = \mathcal{F}(\mathtt{D})r \qquad \mathcal{F}(\mathtt{D}) \in \mathbf{D} \qquad (\mathcal{F}(\mathtt{l}), r) \in \mathsf{Reacts}$$
$$\mathcal{F}(\mathtt{a}) = a \ .$$

■

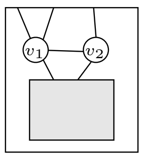
3 Sliding IPO squares

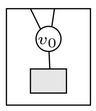
3.1 Introduction and motivation

Part 1 presents a series of congruence proofs, each one for a different operational equivalence. The theme throughout is the separation of "reactive system" into two categories with a functor $\mathcal{F}$ between them: the domain of $\mathcal{F}$, i.e. $\hat{\mathbf{C}}$, is a category in which RPOs and IPOs exist; the codomain of $\mathcal{F}$, i.e. $\mathbf{C}$, is a category containing (i) the agents that perform reactions and labelled transitions and (ii) the agent contexts that serve as the labels and specify the closure condition for congruence. This separation is useful because the category for which we prove a congruence result is typically not one in which RPOs exist, as I show in the next section when considering categories of graph contexts. Thus functorial reactive systems are a powerful generalisation of reactive systems.

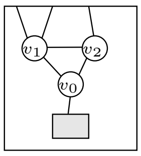
This separation imposes a burden, though, in the form of the list of requirements given in Definition 2.5, i.e.: $\mathcal{F}$ lifts agents, creates isos, creates compositions, and allows IPO sliding. The motivation for the current section is to alleviate this burden by showing that all of these properties follow directly if we construct $\hat{\mathbf{C}}$, $\mathbf{C}$, and $\mathcal{F}$ from a precategory $\mathbf{A}$ in a particular way. The assumption is that $\mathbf{A}$ is easier to construct than the others. (The next section gives a concrete instance of $\mathbf{A}$.)

Precategories are defined formally below. By way of motivation, though, let us look informally at an example we have in mind when deriving $\mathcal{F} : \hat{\mathbf{C}} \rightarrow \mathbf{C}$ from $\mathbf{A}$. A precategory is just like a category but has a composition operation that is partial, i.e. not always defined. For example, consider a precategory $\mathbf{A}$ of "raw contexts". For the purpose of this example, take the objects to be natural numbers (representing the number of ports in an interface). Take the arrows $m \rightarrow n$ to be just like normal graphs but with some added structure, namely an inner and outer interface. An arrow is thus a doughnut-shaped graph consisting of a set of nodes (which we call the *support*) and an arc relation; the arcs link nodes to one another and to the interface ports; m (ordered) interface ports sit on the inside "hole" and n (ordered ones) on the outside. (These are simpler than the graph contexts that appear in the next section.) Composition consists of placing one raw context inside the hole of another and joining together common ports. To see how this works, consider arrows $A_0 : 1 \rightarrow 2$ and $A_1 : 2 \rightarrow 3$:

$A_1 =$ 

$A_0 =$ 

Then their composition, which we denote $A_1 \oplus A_0 : 1 \rightarrowtail 3$, is as follows:

$A_1 \oplus A_0 =$ 

This example reveals why composition is partial in $\mathbf{A}$. If we form A_0' from A_0 by renaming the node v_0 by v_2 then the supports of A_0' and A_1 are *not disjoint*. Thus the composition $A_1 \oplus A_0'$ is undefined since there is no clear choice for the support of the composite.

There are several possible ways of building a true category (i.e. with a total composition relation) from the data provided by $\mathbf{A}$. Two possible ways are as follows:

- We can construct a category $\hat{\mathbf{C}}$ whose objects are pairs (m, U) where m is an object of $\mathbf{A}$ (in this case a natural number) and U is a set. An arrow $(m_0, U_0) \rightarrowtail (m_1, U_1)$ consists of an $\mathbf{A}$-arrow $A : m_0 \rightarrowtail m_1$ for which $U_0 \subseteq U_1$ and $U_1 \setminus U_0$ is equal to the support of A. Thus we can incorporate A_1 (given above) into many possible arrows in $\hat{\mathbf{C}}$, e.g. $(2, \varnothing) \xrightarrow{A_1} (3, \{v_1, v_2\})$ and $(2, \{w\}) \xrightarrow{A_1} (3, \{w, v_1, v_2\})$. As a result composition is always well-defined: if $(m_i, U_i) \xrightarrow{A_i} (m_{i+1}, U_{i+1})$ are $\hat{\mathbf{C}}$-arrows for $i = 0, 1$ then the supports of A_0 and A_1 are disjoint.

- We can construct a category $\mathbf{C}$, whose objects are the objects of $\mathbf{A}$ and whose arrows are $\simeq$-equivalence classes of $\mathbf{A}$-arrows. Two $\mathbf{A}$-arrows are $\simeq$-equivalent iff they are graph-isomorphic, i.e. have (potentially) different supports but look like the same graphs. Composition for this category is also well-defined since it is always possible when composing arrows to find representatives of each equivalence class with disjoint supports.

One might consider a third way, i.e. to use the arrows of $\mathbf{A}$ but rename the supports so as to make them disjoint when composing arrows; but this yields a bicategory, since composition is not associative. I consider this possibility in detail in Subsection 3.7.

What is the relationship between $\hat{\mathbf{C}}$ and $\mathbf{C}$? There is a simple functor $\mathcal{F}$ which discards the set component of each $\hat{\mathbf{C}}$-object (i.e. $\mathcal{F} : (m, U) \mapsto m$) and maps each arrow to its $\simeq$-equivalence class. As we see later in this section, this functor has all of the desired properties listed earlier, e.g. $\mathcal{F}$ allows IPO sliding.

The rest of this section repeats the above constructions in full formality.

3.2 Properties of A

First we formally define a precategory (see [MSS00]):

Definition 3.1 (precategory) A *precategory* $\mathbf{A}$ consists of similar data to that of a category: a collection of objects $m, n, \ldots$; a collection of arrows $\mathbf{A}(m, m')$ between objects m and m'; an identity arrow $\mathsf{id}_m \in \mathbf{A}(m, m)$ for all m; and a *partial* composition operation, which we write here as $\oplus : \mathbf{A}(m_1, m_2) \times \mathbf{A}(m_0, m_1) \rightharpoonup \mathbf{A}(m_0, m_2)$ on arrows. Identity: composition with an identity arrow is always well-defined, i.e. for all $A : m_0 \rightarrowtriangle m_1$, we have that $\mathsf{id}_{m_1} \oplus A = A = A \oplus \mathsf{id}_{m_0}$ and both compositions are defined. Associativity: if $A_2 \oplus A_1$ and $A_1 \oplus A_0$ are defined then either both $A_2 \oplus (A_1 \oplus A_0)$ and $(A_2 \oplus A_1) \oplus A_0$ are undefined or both are defined and equal. ■

Next we define some special properties of a precategory. These properties form a specification (used in Section 4) which any precategory is required to satisfy in order to make use of the constructions and propositions of this section.

Definition 3.2 (well-supported precategory) $\mathbf{A}$ is a *well-supported precategory* iff it is a precategory and it satisfies the following properties:

- $\mathbf{A}$ has a *support function* $|\cdot|$ that maps an arrow to a set such that $A_1 \oplus A_0$ is defined iff $|A_1| \cap |A_0| = \varnothing$ and $\mathsf{Dom}\, A_1 = \mathsf{Cod}\, A_0$. The support function satisfies additionally two axioms:

$$|A_1 \oplus A_0| = |A_1| \uplus |A_0| \qquad \text{(provided } A_1 \oplus A_0 \text{ is defined)} \qquad \textsc{Supp-comp}$$
$$|\mathsf{id}_m| = \varnothing \; . \qquad \textsc{Supp-id}$$

- For any arrow $A \in \mathbf{A}(m_0, m_1)$ and any injective map ρ for which $\mathsf{Dom}\, \rho \supseteq |A|$, there exists an arrow $\rho{\bullet}A \in \mathbf{A}(m_0, m_1)$, which is called the *support translation by* ρ *of* A, where:

$$\rho{\bullet}\mathsf{id}_m = \mathsf{id}_m \qquad \textsc{Trans-id-r}$$
$$\rho{\bullet}(A_1 \oplus A_0) = \rho{\bullet}A_1 \oplus \rho{\bullet}A_0 \qquad \textsc{Trans-comp-r}$$
$$\mathsf{Id}_{|A|}{\bullet}A = A \qquad \textsc{Trans-id-l}$$
$$(\rho_1 \circ \rho_0){\bullet}A = \rho_1{\bullet}(\rho_0{\bullet}A) \qquad \textsc{Trans-comp-l}$$
$$\rho_0 \restriction |A| = \rho_1 \restriction |A| \text{ implies } \rho_0{\bullet}A = \rho_1{\bullet}A \qquad \textsc{Trans-res}$$
$$|\rho{\bullet}A| = \rho|A| \; . \qquad \textsc{Trans-supp}$$

A note about Trans-comp-r: Since ρ is injective, Trans-supp implies that the LHS is defined iff the RHS is defined. (These axioms are similar to those of Honda's Rooted P-Sets [Hon00], though his application concerns free names and renaming.)

■

3.3 Construction of $\hat{\mathbf{C}}$

We now turn to problem of building a genuine category from a well-supported precategory $\mathbf{A}$. The idea is to enrich the object structure with enough data so that composition is always defined. This construction is captured in the following definition:

Definition 3.3 (track) Given a well-supported precategory $\mathbf{A}$, the *track* of $\mathbf{A}$ is a category $\hat{\mathbf{C}}$. An object of $\hat{\mathbf{C}}$ is a *profile*: a pair (m, U) where m is an object of $\mathbf{A}$ and U is a set. We let p range over profiles and adopt the firm convention that the components of a profile p are always written (m, U) with suitable decoration, e.g. $p' = (m', U')$ and $p_i = (m_i, U_i)$. An arrow $p_0 \xrightarrow{A} p_1$ consists of an arrow $A \in \mathbf{A}(m_0, m_1)$ such that $U_0 \subseteq U_1$ and $|A| = U_1 \setminus U_0$. We always include the profiles when referring to an arrow of $\hat{\mathbf{C}}$ since different $\hat{\mathbf{C}}$-arrows can be constructed from the same $\mathbf{A}$-arrow, each with different profiles. The identities of $\hat{\mathbf{C}}$ are defined by $\mathsf{id}_p \mathrel{\hat{=}} p \xrightarrow{\mathsf{id}_m} p$. Composition is defined in terms of the underlying composition in $\mathbf{A}$:

$$p_0 \xrightarrow{A_0} p_1 \xrightarrow{A_1} p_2 \mathrel{\hat{=}} p_0 \xrightarrow{A_1 \oplus A_0} p_2 \; .$$ ■

Proposition 3.4 If $\hat{\mathbf{C}}$ is the track of $\mathbf{A}$ then $\hat{\mathbf{C}}$ is a category.

Proof

Composition is well-defined: if $p_i \xrightarrow{A_i} p_{i+1}$ are arrows for $i = 0, 1$, then $|A_1| \cap |A_0| = (U_2 \setminus U_1) \cap (U_1 \setminus U_0) = \varnothing$, so $A_1 \oplus A_0$ is defined. Furthermore, $U_0 \subseteq U_1 \subseteq U_2$ and

$$U_2 \setminus U_0 \;=\; (U_2 \setminus U_1) \uplus (U_1 \setminus U_0) \;=\; |A_1| \uplus |A_0| \;=^{\text{SUPP-COMP}}\; |A_1 \oplus A_0| \; ,$$

so $p_0 \xrightarrow{A_1 \oplus A_0} p_2$ is an arrow of $\hat{\mathbf{C}}$ as desired.

The identity arrows are well-defined: By SUPP-ID, $|\mathsf{id}_m| = \varnothing$, so for any $p = (m, U)$, $p \xrightarrow{\mathsf{id}_m} p$ is an arrow of $\hat{\mathbf{C}}$.

Composition is associative and respects identities: Immediate because these same properties hold in $\mathbf{A}$. ■

3.4 Operations on $\hat{\mathbf{C}}$

In order to motivate the following results, let us recall the intuitions about the functor $\mathcal{F}$ (defined formally later): $\mathcal{F}$ maps $p_0 \xrightarrow{A} p_1$ to an isomorphism equivalence class of A and throws away the set data contained in the profiles p_0, p_1. To prove that $\mathcal{F}$ allows IPO sliding, we require two things. (i) We have to understand how two $\hat{\mathbf{C}}$-arrows are related when they have the same $\mathcal{F}$ image, i.e. $\mathcal{F}(p_0 \xrightarrow{A} p_1) = \mathcal{F}(p'_0 \xrightarrow{A'} p'_1)$. (ii) From the first piece of information, we have to slide similarly an IPO square whose left leg is $p_0 \xrightarrow{A} p_1$ to one whose left leg is $p'_0 \xrightarrow{A'} p'_1$.

For (i), it is clear that we can perform some *profile translation* (defined precisely later) on $p_0 \xrightarrow{A} p_1$ to replace the set components of p_0 and p_1 and then perform a support translation on the resulting arrow to arrive at $p'_0 \xrightarrow{A'} p'_1$. If these two operations (profile translation and support translation) are iso functors (and thus preserve categorical constructions) then we can accomplish (ii).

These two operations are not in fact iso functors on the whole of $\hat{\mathbf{C}}$ but they are on certain *convex subcategories*, as defined next. Fortunately, the subcategories in question are rich enough to be proxies for $\hat{\mathbf{C}}$ with respect to IPO squares, as shown in the result immediately following the definition:

Definition 3.5 (convex subcategories of $\hat{\mathbf{C}}$) For any sets $U_0 \subseteq U_1$, we write $\hat{\mathbf{C}}_{U_0,U_1}$ for the *convex subcategory of* $\hat{\mathbf{C}}$ *w.r.t.* U_0, U_1, namely the full subcategory of $\hat{\mathbf{C}}$ formed by taking only those profiles (m, U) for which $U_0 \subseteq U \subseteq U_1$. ■

Proposition 3.6 Suppose that Figure 6 commutes in $\hat{\mathbf{C}}$. Then it is an IPO in $\hat{\mathbf{C}}$ iff it is an IPO in $\hat{\mathbf{C}}_{U,U_2}$.

$$\begin{tikzcd} p \arrow[r, "A_0"] \arrow[d, "A_1"'] & p_0 \arrow[d, "B_0"] \\ p_1 \arrow[r, "B_1"'] & p_2 \end{tikzcd}$$

6

Proof For any pair of arrows $p \xrightarrow{C} p' \xrightarrow{C'} p_2$, we have that $U \subseteq U' \subseteq U_2$, so $p' \in \mathsf{obj}\,\hat{\mathbf{C}}_{U,U_2}$ (Definition 3.5) hence $\hat{\mathbf{C}}_{U,U_2}$ satisfies the hypothesis of Proposition 2.3, whence the result follows. ■

Now we now define precisely profile translation and establish that it is an iso functor:

Proposition 3.7 (profile translation is an iso functor) If $W_1 = W_0 \uplus W$ and $W'_1 = W'_0 \uplus W$ then the following operation, called *profile translation*, induces an isomorphism of categories $\mathcal{H} : \hat{\mathbf{C}}_{W_0,W_1} \rightarrow \hat{\mathbf{C}}_{W'_0,W'_1}$,

$$\begin{aligned} \mathcal{H} :\ & (m, W_0 \uplus V) && \mapsto && (m, W'_0 \uplus V) \qquad \text{for } V \subseteq W \\ \mathcal{H} :\ & (p_0 \xrightarrow{A} p_1) && \mapsto && \mathcal{H}(p_0) \xrightarrow{A} \mathcal{H}(p_1)\ . \end{aligned}$$

Proof

$\mathcal{H}$ is well-defined on arrows: Suppose that $(p_0 \xrightarrow{A} p_1) \in \hat{\mathbf{C}}_{W_0,W_1}$ where $U_i = W_0 \uplus V_i$, $i = 0, 1$ for some $V_0 \subseteq V_1 \subseteq W$. Now, $W'_0 \uplus V_0 \subseteq W'_0 \uplus V_1$ and

$$(W'_0 \uplus V_1) \setminus (W'_0 \uplus V_0) \ =\ V_1 \setminus V_0 \ =\ |A|$$

so $(\mathcal{H}(p_0) \xrightarrow{A} \mathcal{H}(p_1)) \in \hat{\mathbf{C}}_{W'_0,W'_1}$ as desired.

$\mathcal{H}$ is a functor: Consider the action of $\mathcal{H}$ on identities:

$$\mathcal{H}(\mathsf{id}_p) \ =\ \mathcal{H}(p \xrightarrow{\mathsf{id}_m} p) \ =\ \mathcal{H}(p) \xrightarrow{\mathsf{id}_m} \mathcal{H}(p) \ =\ \mathsf{id}_{\mathcal{H}(p)}$$

and on compositions:

$$\begin{aligned} \mathcal{H}(p_0 \xrightarrow{A_0} p_1 \xrightarrow{A_1} p_2) &= \mathcal{H}(p_0 \xrightarrow{A_1 \oplus A_0} p_2) \\ &= \mathcal{H}(p_0) \xrightarrow{A_1 \oplus A_0} \mathcal{H}(p_2) \\ &= \mathcal{H}(p_0) \xrightarrow{A_1} \mathcal{H}(p_1) \xrightarrow{A_0} \mathcal{H}(p_2) \end{aligned}$$

$\mathcal{H}$ **is an isomorphism of categories:** By symmetry, the map $\mathcal{H}' : \hat{\mathbf{C}}_{W_0',W_1'} \rightarrow \hat{\mathbf{C}}_{W_0,W_1}$ defined by

$$\begin{aligned} \mathcal{H}' : \ & (m, W_0' \uplus V) && \mapsto (m, W_0 \uplus V) \qquad \text{for } V \subseteq W \\ \mathcal{H}' : \ & (p_0 \xrightarrow{A} p_1) && \mapsto \mathcal{H}'(p_0) \xrightarrow{A} \mathcal{H}'(p_1) \ . \end{aligned}$$

is also a functor and clearly inverts $\mathcal{H}$, as desired. ■

Finally we lift the support translation operation from $\mathbf{A}$ to $\hat{\mathbf{C}}$ in a straightforward way. This definition induces a functor on convex subcategories of $\hat{\mathbf{C}}$

Proposition 3.8 (support translation is an iso functor) Given $W_0 \subseteq W_1$ and an injection ρ with $\mathsf{Dom}\,\rho \supseteq W_1$, the following operation, called *support translation*, induces an isomorphism of categories $\rho\text{•}(\cdot) : \hat{\mathbf{C}}_{W_0,W_1} \rightarrow \hat{\mathbf{C}}_{\rho W_0,\rho W_1}$,

$$\begin{aligned} \rho\text{•}(m, U) & \ \hat{=}\ (m, \rho U) \\ \rho\text{•}(p_0 \xrightarrow{A} p_1) & \ \hat{=}\ \rho\text{•}p_0 \xrightarrow{\rho\text{•}A} \rho\text{•}p_1 \ , \end{aligned}$$

where $\rho\text{•}A$ is the support translation by ρ of A in $\mathbf{A}$.

Proof

$\rho\text{•}(\cdot)$ **is well-defined on arrows:** Suppose that $(p_0 \xrightarrow{A} p_1) \in \hat{\mathbf{C}}_{W_0,W_1}$. Then $W_0 \subseteq U_0 \subseteq U_1 \subseteq W_1 \subseteq \mathsf{Dom}\,\rho$. Thus $\rho U_0 \subseteq \rho U_1$ and

$$\rho U_1 \setminus \rho U_0 \ =^{\rho \text{ injective}} \ \rho(U_1 \setminus U_0) = \rho|A| \ =^{\textsc{Trans-supp}} \ |\rho\text{•}A| \ ,$$

so $\rho\text{•}(\cdot)$ maps an arrow to an arrow.

$\rho\text{•}(\cdot)$ **is a functor:** By Trans-id-r and Trans-comp-r, $\rho\text{•}(\cdot)$ preserves identities and compositions, so is a functor.

$\rho\text{•}(\cdot)$ **is an iso functor:** Note that ρ has an inverse $\rho^{-1} : \mathsf{Im}\,\rho \rightarrowtail \mathsf{Dom}\,\rho$. Furthermore,

$$\rho^{-1}\text{•}(\cdot) : \hat{\mathbf{C}}_{\rho W_0,\rho W_1} \rightarrow \hat{\mathbf{C}}_{W_0,W_1}$$

is a functor for the same reasons (given above) that $\rho\text{•}(\cdot)$ is. By Trans-comp-l and Trans-id-l, the functors $\rho\text{•}(\cdot)$ and $\rho^{-1}\text{•}(\cdot)$ invert each other, so $\rho\text{•}(\cdot)$ is an isomorphism of categories, as required. ■

3.5 Construction of C

We now turn to the construction of $\mathbf{C}$, which was described informally in Subsection 3.1. Recall that the arrows of $\mathbf{C}$ are equivalence classes of arrows of $\mathbf{A}$. In this subsection we make precise the underlying equivalence relation and the construction of $\mathbf{C}$ and verify that $\mathbf{C}$ is a well-defined category.

Definition 3.9 ($\hat{\simeq}$-equivalence for A) Given two arrows $A, A' : m_0 \rightarrowtriangle m_1$ in $\mathbf{A}$, we say that they are $\hat{\simeq}$-*equivalent*, written $A \hat{\simeq} A'$, iff there exists a bijection $\rho : |A| \rightarrowtail\!\!\!\twoheadrightarrow |A'|$ such that $\rho \bullet A = A'$. By TRANS-ID-L and TRANS-COMP-L, $\hat{\simeq}$ is an equivalence relation. ■

Now the construction of $\mathbf{C}$ is straightforward:

Definition 3.10 (support quotient) Given a well-supported precategory $\mathbf{A}$, the *support quotient* of $\mathbf{A}$ is a category $\mathbf{C}$. The objects of $\mathbf{C}$ are the objects of $\mathbf{A}$. The arrows $m_0 \rightarrowtriangle m_1$ of $\mathbf{C}$ are $\hat{\simeq}$-equivalence classes of arrows in $\mathbf{A}$:

$$\mathbf{C}(m_0, m_1) \;\hat{=}\; \{[A]_{\hat{\simeq}} \;\,/\;\, A \in \mathbf{A}(m_0, m_1)\} \;.$$

Identities: $\mathsf{id}_m \in \mathbf{C}(m, m) \;\hat{=}\; [\mathsf{id}_m]_{\hat{\simeq}}$. Composition: if $A_1 \oplus A_0$ is defined in $\mathbf{A}$ then $[A_1]_{\hat{\simeq}}[A_0]_{\hat{\simeq}} \hat{=} [A_1 \oplus A_0]_{\hat{\simeq}}$. ■

This definition yields a well-defined category:

Proposition 3.11 If $\mathbf{C}$ is the support quotient of $\mathbf{A}$ then $\mathbf{C}$ is a category.

Proof

Composition is total: Consider any two arrows in $\mathbf{C}$ such as $[A_i]_{\hat{\simeq}} : m_i \rightarrowtriangle m_{i+1}$ for $i = 0, 1$. Let W be a fresh set in bijection with $|A_1|$, as witnessed by $\rho : |A_1| \rightarrowtail\!\!\!\twoheadrightarrow W$. Then $\rho \bullet A_1 \oplus A_0$ is defined since $|\rho \bullet A_1| \cap |A_0| =^{\text{TRANS-SUPP}} W \cap |A_0| = \emptyset$; thus

$$[A_1]_{\hat{\simeq}}[A_0]_{\hat{\simeq}} \;=\; [\rho \bullet A_1]_{\hat{\simeq}}[A_0]_{\hat{\simeq}} \;=\; [\rho \bullet A_1 \oplus A_0]_{\hat{\simeq}} \;,$$

as desired.

Composition is well-defined: Let $[A_i]_{\hat{\simeq}} = [A'_i]_{\hat{\simeq}}$ for $i = 0, 1$ with both $A_1 \oplus A_0$ and $A'_1 \oplus A'_0$ defined. Claim: $[A_1]_{\hat{\simeq}}[A_0]_{\hat{\simeq}} = [A'_1]_{\hat{\simeq}}[A'_0]_{\hat{\simeq}}$. By hypothesis, there exist bijections $\rho_i : |A_i| \rightarrowtail\!\!\!\twoheadrightarrow |A'_i|$ such that $A'_i = \rho_i \bullet A_i$ for $i = 0, 1$. Since $|A_1| \cap |A_0| = \emptyset$ and $|A'_1| \cap |A'_0| = \emptyset$, we can define $\rho \;\hat{=}\; \rho_0 \uplus \rho_1$, a union of bijections with disjoint domains and disjoint codomains. Now,

$$\rho \bullet (A_1 \oplus A_0) \;=^{\text{TRANS-COMP-R}}\; \rho \bullet A_1 \oplus \rho \bullet A_0 \;=^{\text{TRANS-RES}}\; \rho_1 \bullet A_1 \oplus \rho_0 \bullet A_0 \;=\; A'_1 \oplus A'_0$$

so

$$[A_1]_{\hat{\simeq}}[A_0]_{\hat{\simeq}} \;=\; [A_1 \oplus A_0]_{\hat{\simeq}} \;=\; [A'_1 \oplus A'_0]_{\hat{\simeq}} \;=\; [A'_1]_{\hat{\simeq}}[A'_0]_{\hat{\simeq}} \;,$$

as desired.

Composition is associative: Follows from the associativity of the underlying composition in $\mathbf{A}$.

Composition respects identities: Follows from the fact that composition respects identities in $\mathbf{A}$. ■

3.6 Construction of $\mathcal{F}$

In this final subsection we define a functor $\mathcal{F}$ from $\hat{\mathbf{C}}$ (the track of $\mathbf{A}$) to $\mathbf{C}$ (the support quotient of $\mathbf{A}$). We then verify that $\mathcal{F}$ has all the required properties, i.e.: $\mathcal{F}$ lifts agents, creates isos, creates compositions, and allows IPO sliding. For the first we verify a stronger property defined below, namely $\mathcal{F}$ lifts arrows by their domain. The reason for this choice is that there is no postulated distinguished object in $\mathbf{A}$ or $\mathbf{C}$ corresponding to the 0 of a functorial reactive system (see Definition 2.5), which is required when defining the property "$\mathcal{F}$ lifts agents". However, the stronger property "$\mathcal{F}$ lifts arrows by their domain" is well-defined.

Definition 3.12 (support-quotienting functor) Let $\mathbf{A}$ be a well-supported precategory and $\hat{\mathbf{C}}$ and $\mathbf{C}$ be as in Definition 3.3 and Definition 3.10. Then we define a map $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ called the *support-quotienting functor*:

$$\begin{aligned} \mathcal{F} :\ & (m,U) && \mapsto\ m \\ \mathcal{F} :\ & (p_0 \xrightarrow{A} p_1) && \mapsto\ \mathcal{F}(p_0) \xrightarrow{[A]_{\simeq}} \mathcal{F}(p_1)\ . \end{aligned}$$ ■

Lemma 3.13 $\mathcal{F}$ is a functor.

Proof Observe the action on identities:

$$\mathcal{F}(\mathsf{id}_p) \ = \ \mathcal{F}(p \xrightarrow{\mathsf{id}_m} p) \ = \ [\mathsf{id}_m]_{\simeq} \ = \ \mathsf{id}_m$$

and on compositions:

$$\begin{aligned} \mathcal{F}(p_0 \xrightarrow{A_0} p_1 \xrightarrow{A_1} p_2) \ &= \ \mathcal{F}(p_0 \xrightarrow{A_1 \oplus A_0} p_2) \\ &= \ [A_1 \oplus A_0]_{\simeq} \\ &= \ [A_1]_{\simeq}[A_0]_{\simeq} \\ &= \ \mathcal{F}(p_1 \xrightarrow{A_1} p_2)\,\mathcal{F}(p_0 \xrightarrow{A_0} p_1) \end{aligned}$$ ■

Now we prove all the easy properties of $\mathcal{F}$:

Theorem 3.14 Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be the support-quotienting functor constructed from $\mathbf{A}$. Then:

- $\mathcal{F}$ lifts arrows by their domain: if $\mathcal{F}(p_0) = \mathsf{Dom}\,[A]_{\simeq}$ then there exists $p_0 \xrightarrow{B} p_1$ such that $\mathcal{F}(p_0 \xrightarrow{B} p_1) = [A]_{\simeq}$.
- $\mathcal{F}$ creates isos: if $\mathcal{F}(p_0 \xrightarrow{A} p_1)$ is an iso then $p_0 \xrightarrow{A} p_1$ is an iso.
- $\mathcal{F}$ creates compositions: if

$$\mathcal{F}(p'_0 \xrightarrow{B} p'_2) \ = \ [A_1]_{\simeq}[A_0]_{\simeq}$$

then there exist $\hat{\mathbf{C}}$-arrows $p'_i \xrightarrow{B_i} p'_{i+1}$ with

$$\begin{aligned} \mathcal{F}(p'_i \xrightarrow{B_i} p'_{i+1}) &= [A_i]_{\simeq} \qquad \text{for } i = 0,1 && (1) \\ p'_0 \xrightarrow{B} p'_2 &= p'_0 \xrightarrow{B_0} p'_1 \xrightarrow{B_1} p'_2 && (2) \end{aligned}$$

Proof

$\mathcal{F}$ **lifts arrows by their domain:** Suppose $[A]_{\simeq} : (m_0, n_0) \to (m_1, n_1)$, thus $A : (m_0, n_0) \to (m_1, n_1)$ in $\mathbf{A}$. Let W be a fresh set in bijection with $|A|$, as witnessed by $\rho : |A| \rightarrowtail\!\!\!\!\twoheadrightarrow W$. Then $p_0 \xrightarrow{\rho\bullet A} p_1$ is an arrow in $\hat{\mathbf{C}}$, where $U_1 \mathrel{\hat{=}} U_0 \uplus W$. Furthermore $\mathcal{F}(p_0 \xrightarrow{\rho\bullet A} p_1) = [A]_{\simeq}$ as desired.

$\mathcal{F}$ **creates isos:** Suppose $\mathcal{F}(p_0 \xrightarrow{A} p_1)$ is an iso, i.e. there exists an $\mathbf{A}$-arrow $A' : m_1 \to m_0$ such that:

$$\begin{aligned} [A']_{\simeq}[A]_{\simeq} &= \mathsf{id}_{m_0} = [\mathsf{id}_{m_0}]_{\simeq} \\ [A]_{\simeq}[A']_{\simeq} &= \mathsf{id}_{m_1} = [\mathsf{id}_{m_1}]_{\simeq} \end{aligned}$$

Without loss of generality, assume that $|A| \cap |A'| = \varnothing$. Then $A' \oplus A \simeq \mathsf{id}_{m_0}$ and $A \oplus A' \simeq \mathsf{id}_{m_1}$. By Supp-id and Trans-id-r, $A' \oplus A = \mathsf{id}_{m_0}$ and $A \oplus A' = \mathsf{id}_{m_1}$. By Supp-comp, $|A| = |A'| = \varnothing$. Thus $U_1 = U_0$ and $p_1 \xrightarrow{A'} p_0$ is a $\hat{\mathbf{C}}$-arrow. Moreover,

$$p_0 \xrightarrow{A} p_1 \xrightarrow{A'} p_0 = p_0 \xrightarrow{A' \oplus A} p_0 = p_0 \xrightarrow{\mathsf{id}_{m_0}} p_0 = \mathsf{id}_{p_0}$$

and symmetrically. Thus $p_0 \xrightarrow{A} p_1$ is an iso in $\hat{\mathbf{C}}$ as desired.

$\mathcal{F}$ **creates compositions:** Without loss of generality, assume that $|A_1| \cap |A_0| = \varnothing$. Then there exist p_0, p_1, p_2 such that $p_i \xrightarrow{A_i} p_{i+1}$ are arrows in $\hat{\mathbf{C}}$ for $i = 0,1$. By the definition of $\mathcal{F}$, there exists a bijection $\rho : |A_1| \uplus |A_0| \rightarrowtail\!\!\!\!\twoheadrightarrow |B|$ such that $\rho\bullet(A_1 \oplus A_0) = B$; moreover $m_0 = m'_0$ and $m_2 = m'_2$. Let $B_i \mathrel{\hat{=}} \rho\bullet A_i$ for $i = 0,1$. Let $(m'_1, U'_1) \mathrel{\hat{=}} (m_1, U'_0 \uplus |B_0|)$, thus defining p'_1. We claim that $p'_i \xrightarrow{B_i} p'_{i+1}$ are arrows in $\hat{\mathbf{C}}$ for $i = 0,1$; there are three things that we need to check:

$$\begin{gathered} U'_0 \subseteq U'_0 \uplus |B_0| \subseteq U'_0 \uplus |B| = U'_2 \\ U'_1 \setminus U'_0 = |B_0| \\ U'_2 \setminus U'_1 = (U'_0 \uplus |B|) \setminus (U'_0 \uplus |B_0|) = |B| \setminus |B_0| = |B_1| \,. \end{gathered}$$

Now, $B_1 \oplus B_0 = B$ by Trans-comp-r, from which (2) follows. Also, by Trans-res, $B_i \simeq A_i$ for $i = 0,1$, from which (1) follows.

■

An aside on the condition "$\mathcal{F}$ creates compositions": This looks tantalisingly close to the Conduché fibration property [Con72, Joh99, BF00], especially if one says that two decompositions in $\hat{\mathbf{C}}$ are equivalent if one is the result of a support translation of the other. Perhaps there is some 2-category version of the Conduché property which works exactly if one thinks of $\hat{\mathbf{C}}$ as a 2-category with support translations as 2-cells. Let us now return to the main flow of the argument.

Finally, we prove the key property, namely that $\mathcal{F}$ allows IPO sliding. The proof follows the outline given in Subsection 3.4. We start with two $\hat{\mathbf{C}}$-arrows with the same image under $\mathcal{F}$, namely $\mathcal{F}(p \xrightarrow{A_1} p_1) = \mathcal{F}(p' \xrightarrow{A_1'} p_1')$. The first arrow is the left leg of an IPO square in $\hat{\mathbf{C}}$. This square is also an IPO in $\hat{\mathbf{C}}_{U,U_2}$, the convex subcategory w.r.t. U, U_2 (Definition 3.5), where $U \subseteq U_2$ are the sets in, respectively, the upper-left and lower-right profiles of the square. We isomorphically transform this subcategory by profile translation and then support translation, gaining a new square that has three properties: it has the same image under $\mathcal{F}$ as the original; its left leg is $p' \xrightarrow{A_1'} p_1'$; it is an IPO in a convex subcategory, so is an IPO in $\hat{\mathbf{C}}$.

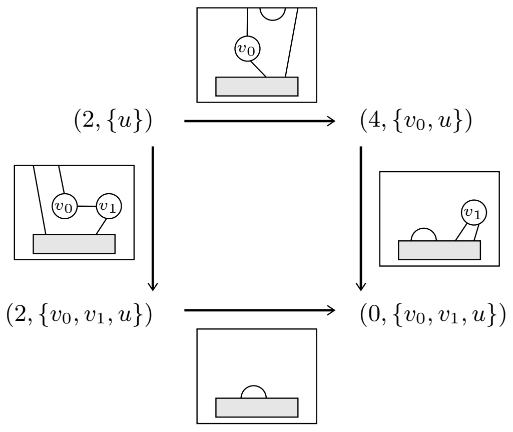
Before looking at the technical details, it is useful to consider a concrete case of sliding. Because we have not formally defined the graph contexts referred to at the beginning of this section, it is impossible to be precise about which commuting squares are IPOs and which are not. Nonetheless, the activity of "sliding" is relevant for all commuting squares, whether or not they are IPOs.

Let us consider a category of graph contexts formed as the track (Definition 3.3) of the precategory or raw contexts informally defined in Subsection 3.1. The arrows of this category are just like the raw contexts (doughnut-shaped graphs with an inner and outer interface) but with profiles (Definition 3.3) rather than just natural numbers as objects.

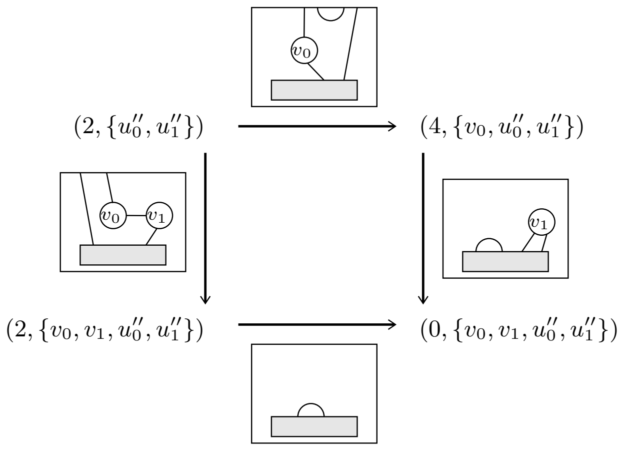
Consider the square in Figure 7(1). Its left leg has the same $\mathcal{F}$ image as the left leg of the square in Figure 7(3): the two graph contexts look essentially the same, the only difference being the supports. With a profile translation, we can replace the singleton set $\{u\}$ in the top-left corner of Figure 7(1) with a fresh 2-element set $\{u_0'', u_1''\}$, as shown in Figure 7(2). The freshness is essential to prevent clashes with the other nodes present in the square, namely v_0, v_1. Now, if ρ is defined as follows:

$$\begin{aligned} \rho : v_i \mapsto v_i' \qquad & i = 1, 2 \\ \rho : u_i'' \mapsto u_i' \qquad & i = 1, 2 \end{aligned}$$

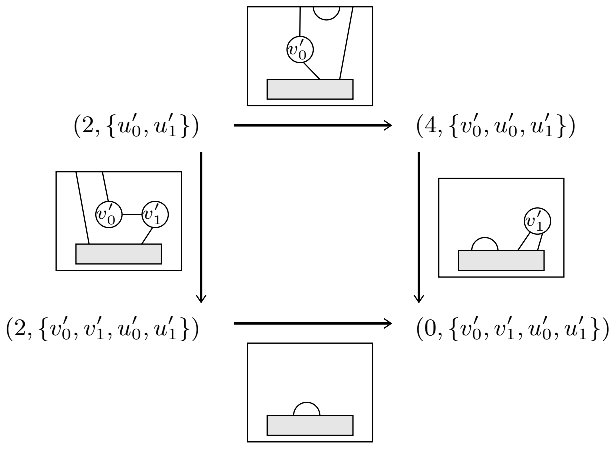
then the support translation by ρ of Figure 7(2) yields Figure 7(3), as desired. Since the passage from Figure 7(1) to Figure 7(2) and then to Figure 7(3) was effected by iso functors, all the universal properties of Figure 7(1) (e.g. being an IPO) hold of Figure 7(3) too.



7(1) A commuting square before sliding


$(2,\{u''_0,u''_1\})$
$(4,\{v_0,u''_0,u''_1\})$
$(2,\{v_0,v_1,u''_0,u''_1\})$
$(0,\{v_0,v_1,u''_0,u''_1\})$
v_0
v_0
v_1
v_1

7(2) First we apply profile translation ...



7(3) ...and then support translation by ρ

Figure 7: Sliding

Theorem 3.15 ($\mathcal{F}$ allows IPO sliding) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be the support-quotienting functor constructed from $\mathbf{A}$. Then $\mathcal{F}$ allows IPO sliding (Definition 2.5).

$$\begin{tikzcd} p \arrow[r, "A_0"] \arrow[d, "A_1"'] & p_0 \arrow[d, "B_0"] \\ p_1 \arrow[r, "B_1"'] & p_2 \end{tikzcd}$$

8

Proof Consider any IPO square in $\hat{\mathbf{C}}$, as in Figure 8, and any arrow $p' \xrightarrow{A'_1} p'_1$ with $\mathcal{F}(p' \xrightarrow{A'_1} p'_1) = \mathcal{F}(p \xrightarrow{A_1} p_1)$; thus $A'_1 = \alpha\bullet A_1$ for some bijection $\alpha : |A_1| \to |A'_1|$ and $U'_1 = U' \uplus \alpha|A_1|$.

By Proposition 3.6, Figure 8 is an IPO in $\hat{\mathbf{C}}_{U,U_2}$. Let U'' be a fresh set in bijection with U', as witnessed by $\mu : U'' \rightarrowtail\!\!\!\!\twoheadrightarrow U'$. Let $U''_2 \mathrel{\hat{=}} U'' \uplus (U_2 \setminus U)$. Then $U_2 \setminus U = U''_2 \setminus U''$ so by Proposition 3.7, there is a profile translation $\mathcal{H} : \hat{\mathbf{C}}_{U,U_2} \to \hat{\mathbf{C}}_{U'',U''_2}$ which is an iso functor and whose action on profiles is:

$$\begin{tikzcd} \mathcal{H}(p) \arrow[r, "A_0"] \arrow[d, "A_1"'] & \mathcal{H}(p_0) \arrow[d, "B_0"] \\ \mathcal{H}(p_1) \arrow[r, "B_1"'] & \mathcal{H}(p_2) \end{tikzcd}$$

9

$$\mathcal{H} : (m, U \uplus V) \mapsto (m, U'' \uplus V) \qquad \text{for } V \subseteq U_2 \setminus U$$

and whose action on arrows leaves the underlying $\mathbf{A}$-arrow component unchanged. Since isomorphisms of categories preserve universal constructions, Figure 9 is an IPO in $\hat{\mathbf{C}}_{U'',U''_2}$ and has the same image under $\mathcal{F}$ as Figure 8 does.

Let W be a fresh set in bijection with $|B_1|$, as witnessed by $\beta : |B_1| \rightarrowtail\!\!\!\!\twoheadrightarrow W$. Let $\rho \mathrel{\hat{=}} \mu \uplus \alpha \uplus \beta$, a union of bijections with mutually disjoint domains and codomains. Also $\mathsf{Dom}\,\rho = U'' \uplus |A_1| \uplus |B_1| = U''_2$. Because ρ is bijective, Proposition 3.8 implies that there is a support translation $\rho\bullet(\cdot) : \hat{\mathbf{C}}_{U'',U''_2} \to \hat{\mathbf{C}}_{\rho U'',\rho U''_2}$ which is an iso functor. Iso functors preserve universal constructions, so Figure 10 is a IPO in $\hat{\mathbf{C}}_{\rho U'',\rho U''_2}$ and has the same image under $\mathcal{F}$. By Proposition 3.6, this square is an IPO in $\hat{\mathbf{C}}$. Moreover,

$$\begin{tikzcd} \rho\bullet\mathcal{H}(p) \arrow[r, "\rho\bullet A_0"] \arrow[d, "\rho\bullet A_1"'] & \rho\bullet\mathcal{H}(p_0) \arrow[d, "\rho\bullet B_0"] \\ \rho\bullet\mathcal{H}(p_1) \arrow[r, "\rho\bullet B_1"'] & \rho\bullet\mathcal{H}(p_2) \end{tikzcd}$$

10

$$\rho\bullet(\mathcal{H}(p) \xrightarrow{A_1} \mathcal{H}(p_1)) \;=\; \mu\bullet\mathcal{H}(p) \xrightarrow{\alpha\bullet A_1} (\mu \uplus \alpha)\bullet\mathcal{H}(p_1) \;=\; p' \xrightarrow{A'_1} p'_1$$

as desired. ■

3.7 Replacing precategories with bicategories

As explained in Section 1, one of the main motivations for presenting the sliding arguments of this section is to advertise them to the (enriched) category theory community. The reason is that precategories are probably not optimal: it would be worthwhile to rework these arguments, replacing precategories with something more "natural" from enriched category theory, e.g. bicategories. Such a change might have side-effects on the results of Part 1 as well, which are discussed later in this subsection.

Recall from Subsection 3.1 that the composition operation $\oplus$ of $\mathbf{A}$ is partial, not total, since it requires the supports of the operands to be disjoint: $A_1 \oplus A_0$ is defined iff $|A_1| \cap$

$|A_0| = \varnothing$ and $\mathsf{Dom}\, A_1 = \mathsf{Cod}\, A_0$ (Definition 3.2). Composition can be made total by forcing the supports of the operands to be disjoint, i.e. defining a new composition $\circ$ for which

$$A_1 \circ A_0 = (\iota_1 \centerdot A_1) \oplus (\iota_0 \centerdot A_0)$$

where $\iota_j : |A_j| \rightarrowtail |A_1| + |A_0|$ is the j-th injection into the coproduct of the supports in **Set** (the category of sets and functions), for $j = 0, 1$. This new definition of composition is total. It is not strictly associative, nor does it have strict identities — which is exactly the kind of situation that bicategories cater for! Let the 2-cells from $A : m \rightarrowtail n$ to $A' : m \rightarrowtail n$ be the support translations $\rho : |A| \rightarrowtail |A'|$ such that $\rho \centerdot A = A'$. All of these 2-cells are isos. Moreover, there are 2-cells mediating between the compositions $A_2 \circ (A_1 \circ A_0)$ and $(A_2 \circ A_1) \circ A_0$ as generated from the underlying isos in **Set** between the coproducts $|A_2| + (|A_1| + |A_0|)$ and $(|A_2| + |A_1|) + |A_0|$. It remains to be checked that all the bicategory coherence requirements are satisfied, but it seems likely since here they rely on the underlying properties of a coproduct, which is a universal construction.

Having indicated how to derive a bicategory from $\mathbf{A}$, it may be possible to go a step further, namely to dispense with $\mathbf{A}$ and $\hat{\mathbf{C}}$ altogether and *to postulate* the existence of a bicategory $\mathbf{B}$ whose 2-cells are all isos. Since the 2-cells are isos, the property "the 1-cells A, A' have a 2-cell between them" is an equivalence relation on the 1-cells of $\mathbf{B}$. We can thereby construct the "support quotient" (cf. Definition 3.10) of $\mathbf{B}$ by quotienting out the 2-cells, thus obtaining a category $\mathbf{E}$. This quotienting is an enriched functor $\mathcal{G} : \mathbf{B} \rightarrowtail \mathbf{E}$.

Before looking at the advantages gained by giving this functor a central role, let us consider the price to be paid. First, the congruence results of Part 1 depend on the existence of RPOs in the "upstairs" category $\hat{\mathbf{C}}$ of a functorial reactive system $\mathcal{F} : \hat{\mathbf{C}} \rightarrowtail \mathbf{C}$. The domain of the functor $\mathcal{G}$ is instead a bicategory, thus necessitating a bicategorical version of RPOs, which we might call *2-RPOs*. One possible definition for 2-RPOs (of which Peter Sewell has made a preliminary study) is to replace in Definition 2.1 each commuting square and triangle by the relevant 2-cell, and their juxtaposition by horizontal and vertical 2-cell compositions. This change would have knock-on effects in the proofs of congruence which would have to be rewritten accordingly. Also, the existence proofs of RPOs for concrete systems would have to be redone to provide 2-RPOs. Neither change seems daunting. The potential reward of using bicategories is the ability to simplify the collection of axioms postulated about $\mathcal{F}$ in the definition of a functorial reactive system (Definition 2.5). The goal would be to give a neater collection involving $\mathcal{G}$ which might be verified for many cases based directly on general categorical reasoning about quotienting bicategories by their 2-cells.

Finally, note that the strategy of bringing in enriched categorical machinery is directly opposite to Milner's in [Mil01]. He discards $\hat{\mathbf{C}}$ (the track of $\mathbf{A}$) and works directly with a precategory $\mathbf{A}$: he then calculates RPOs in $\mathbf{A}$, slides IPO in $\mathbf{A}$, etc. His approach demands that he be vigilant, since he is required to verify that compositions he encounters in **A** are well-defined. Nonetheless, this extra care seems bearable: for most compositions, a trivial inspection of the operands suffices. It is therefore a challenge to category theorists to provide a collection of theorems for smoothly manipulating bicategories (for example) in which the effort needed to apply these results is less than the effort needed to work in the setting of precategories.

4 Action graph contexts

4.1 Introduction

As promised in the previous section, the present one gives a substantial example of a precategory **A-Ixt** of raw contexts based on action calculi. (The “Ixt” is for “insertion context”, a terminology explained in Subsection 4.6.) The need to handle graphs was the original motivation for functorial reactive systems: as we will see in Subsection 4.3, RPOs do not always exist for **C-Ixt**, the support quotient (Definition 3.10) of **A-Ixt**, so it is necessary to consider an upstairs category which does possess sufficient RPOs and a functor down to **C-Ixt**.

The precategory **A-Ixt** has all the extra structure required of **A**, namely a support function and a support translation operation, so is a well-supported precategory (Definition 3.2). By direct instantiation of the results of the previous section, we can construct three things: the track of **A-Ixt**, which we call $\hat{\mathbf{C}}$**-Ixt**; the support quotient of **A-Ixt**, which we call **C-Ixt**; and a functor $\mathcal{F}$: $\hat{\mathbf{C}}$**-Ixt** $\rightarrow$ **C-Ixt**. These are related in the following table to their counterparts from the Section 3:

well-supported precategory:	**A**	**A-Ixt**
track:	$\hat{\mathbf{C}}$	$\hat{\mathbf{C}}$**-Ixt**
support-quotienting functor:	$\downarrow \mathcal{F}$	$\downarrow \mathcal{F}$
support quotient:	**C**	**C-Ixt**

Thus by Theorem 3.14 and Theorem 3.15, $\mathcal{F}$ lifts arrows by their domain, creates isos, creates compositions, and allows IPO sliding. Furthermore **A-Ixt** has a distinguished object 0, hence there are distinguished objects 0 and ε of **C-Ixt** and $\hat{\mathbf{C}}$**-Ixt**, with $\mathcal{F}(\varepsilon) = 0$, whence $\mathcal{F}$ lifts agents. Thus, any choice of reaction rules Reacts for **C-Ixt** and reactive context subcategory **D** of **C-Ixt** (Definition 2.4) yields a functorial reactive system (Definition 2.5).

The main hurdle then in using the congruence results of the section “Further congruence results” in Part 1 is proving that $\mathcal{F}$ has all redex-RPOs (Definition 2.6). It turns out that this property fails for some choices of reaction rules, but that it *does* hold for a rich class of them.

As promised in Section 3 of Part 1, the category **C-Ixt** of graph contexts (the *codomain* of $\mathcal{F}$) does *not* admit enough RPOs for subtle reasons. Examples are shown in Subsection 4.3. The cause is the lack of sufficient intensional information as to *which* node in contexts C_0 or C_1, say, corresponds to a node in the composite C_0C_1. It is exactly $\hat{\mathbf{C}}$**-Ixt**, the *domain* of $\mathcal{F}$, that contains just enough structure to track how the nodes of two contexts are related to the nodes of their composition. By the definition of “$\mathcal{F}$

has all redex-RPOs", the proof obligation is to show that sufficient RPOs exist in $\hat{\mathbf{C}}$**-Ixt** (Theorem 4.16). The proof of their existence is not done in the present paper, but is contained in full in [Lei01b].

The development of action calculi presented in this section intersects with that of [CLM00]; however the latter makes no use of functorial reactive systems, a key contribution of the present paper. Note as well that there are some differences in naming: in this paper I write $\hat{\mathbf{C}}$**-Ixt** and **C-Ixt**; in [CLM00], these are denoted $\mathbf{PIns}_0$ and $\mathbf{ACxt}_0$ respectively.

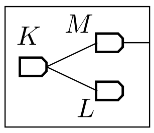
4.2 Action calculi reviewed

I review a restricted class of the action calculi which were presented in [Mil96]. The rest of this section and the next make no specific use of the algebra of action calculi shown here since all of the work is done directly on graphs. Nonetheless, the design decisions taken when defining graphs are guided by the algebraic presentation of action calculi, so the latter provide valuable motivation.

A *closed, shallow action calculus* is a strict monoidal category whose objects are natural numbers $k, m, n, o \ldots$, and whose arrows are called *action graphs* and written $a : (k, o)$, $b : (m, n)$. (I avoid the usual arrow notation $a : k \rightarrowtail o$, reserving it for the context arrows of reactive systems.) The *tensor product* of these two action graphs is $a \otimes b : (k+m, o+n)$; the *composition* of $a : (k, o)$ and $b : (o, m)$ is $a \cdot b : (k, m)$; the *identity* action graph of arity (m, m) is i_m. The order of composition is not conventional in category theory: it connotes the orientation of the action graphs we work with. (Note that this composition is the horizontal juxtaposition of action graphs and has nothing to do with contextual composition which we consider later.) A pair of natural numbers (k, o) is an *arity*; let $\alpha, \beta, \ldots$ range over arities. I deal only with closed, shallow action calculi and so usually omit these adjectives from now on.

An action calculus has a *control signature* $\mathcal{K} = \{K, L, \ldots\}$ consisting of *controls*, each of which has an arity. There are constants $\mathsf{p} : (2, 2)$, $\mathsf{c} : (1, 2)$ and $\omega : (1, 0)$ for permutation, copy and discard. These constants represent only the swapping, sharing and elimination of *arcs*, not of *nodes*. They satisfy simple equations, e.g. $\mathsf{c} \cdot \mathsf{p} = \mathsf{c}$ representing the commutativity of copying. There is also an operator called *reflexion* [Mil94] (similar to the "trace" of Joyal et al. [JSV96]) which we need not detail here.

Finally, each action calculus has a binary *reaction relation* $\longrightarrow\!\!\!\triangleright$, relating action graphs of equal arity. This relation is preserved by all constructions, i.e. by composition, tensor product and reflexion.



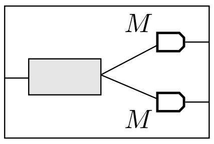


Figure 11: The action graph $K{\cdot}\mathsf{c}{\cdot}(M \otimes L)$ and the context $-_{1,1}{\cdot}\mathsf{c}{\cdot}(M \otimes M)$

For examples of action graphs, let $K : (0,1)$, $M : (1,1)$ and $L : (1,0)$ be controls. Then the following are action graphs, with their arities:

$$\begin{aligned} K \otimes M &: (1,2) \\ K{\cdot}\mathsf{c}{\cdot}(M \otimes L) &: (0,1) \\ (K{\cdot}M) \otimes (M{\cdot}L) &: (1,1)\ . \end{aligned}$$

Composition $\cdot$ binds tighter than tensor $\otimes$, so the last can be written $K{\cdot}M \otimes M{\cdot}L$.

A *context* C is an action graph containing a single hole with arity α, written $-_\alpha$. I omit the arity, writing $-$, if it is determined by the rest of the context or the surrounding discussion. Thus a context $C : \alpha \rightarrowtail \beta$ is an action graph of arity β with a hole of arity α. Here are two contexts along with their domain and codomain arities (the arity of the hole being fully determined in the second case):

$$\begin{aligned} -_{1,1}{\cdot}\mathsf{c}{\cdot}(M \otimes M) &: (1,1) \rightarrowtail (1,2) \\ K{\cdot}-{\cdot}L &: (1,1) \rightarrowtail (0,0)\ . \end{aligned}$$

Figure 11 shows an action graph and a context using a graphical notation. It uses nodes (rectangles with two blunt corners) to represent occurrences of controls, and arcs to represent composition. An action graph with arity (m,n) has m *source* ports on its left side and n *target* ports on its right side. A control node or hole of arity (m,n) has m target ports at its left side n source ports at its right side. At a source node, branching of arcs represents c and absence of arcs represents ω.

Two contexts are equal if the algebraic theory equates them, treating the hole as a control distinct from all others. The composition of two contexts $C : \alpha \rightarrowtail \beta$ and $D : \beta \rightarrowtail \gamma$, written here DC (note the conventional order of composition), is formed by replacing the hole in D by C and joining the arcs according to the common ports. (This is context composition, not horizontal action graph composition described earlier.) Composition is clearly associative, and there is an identity context $\mathsf{id}_\alpha = -_\alpha$ for each arity. An action graph $a : \alpha$ can be considered as a context $a : (0,0) \rightarrowtail \alpha$ whose hole has minimum arity. We shall use lower case letters $a, \ldots$ for action graphs.

We have thus associated with an action calculus a reactive system **C-Ixt**, whose objects are arities, with distinguished null arity $(0, 0)$, and whose arrows are contexts, including action graphs.

4.3 Examples and a problem

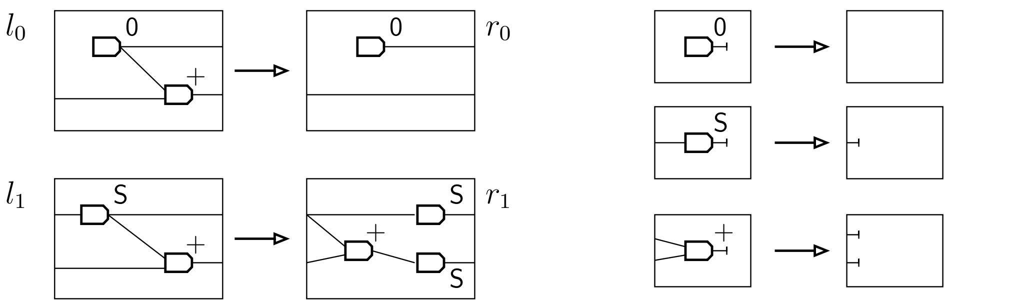
In this subsection I give examples of specific RPOs in **C-Ixt**, illustrating several phenomena. I end with an example showing cases in which RPOs fail to exist; this motivates the strategy of defining a category "upstairs" (the domain of a functor with codomain **C-Ixt**) for which enough RPOs do exist.

Remember that **C-Ixt** is really a family of reactive systems arising from action calculi; each is determined by a control signature and a set of reaction rules.

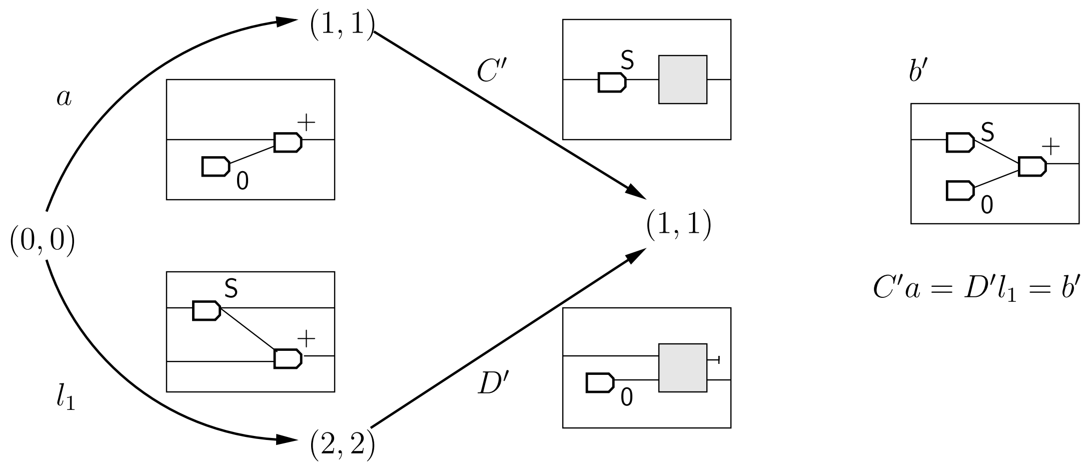
Example 4.1 (arithmetic) I first illustrate how an RPO can determine a labelled transition, using an action calculus for elementary arithmetic having controls $\mathsf{0} : (0, 1)$, $\mathsf{S} : (1, 1)$, and $+ : (2, 1)$. The reactive system is shown in Figure 12; it is an example of the sharing graphs of Hasegawa [Has99], which add sharing to the interaction nets of Lafont [Laf90]. Nodes represent subexpressions, and the forking of arcs allows these to be shared. The reaction rules are in the top diagram; the garbage collection rules allow unattached expressions to be incrementally destroyed.

The middle diagram shows an action graph a occurring in a larger one b', which also contains an occurrence of the redex l_1 of the rule for S. The contexts C' and D' correspond to the two occurrences, which overlap. Now what is the "just large enough" context C which extends a to contain l_1? It is not quite C', because C' has destroyed the possibility of sharing S which is offered by l_1. In fact it is C as shown in the lower diagram; it may not seem "smaller" than C', but it is indeed a factor of C', as witnessed by the context E. (C' cannot be a factor of C; no context F surrounding C' can cause its S-node to be shared.) So our synthesised labelled transition system will admit the transition $a \xrightarrow{C} Dr_1$. We would expect to add further controls, e.g. for subtraction, before obtaining an interesting behavioural congruence.

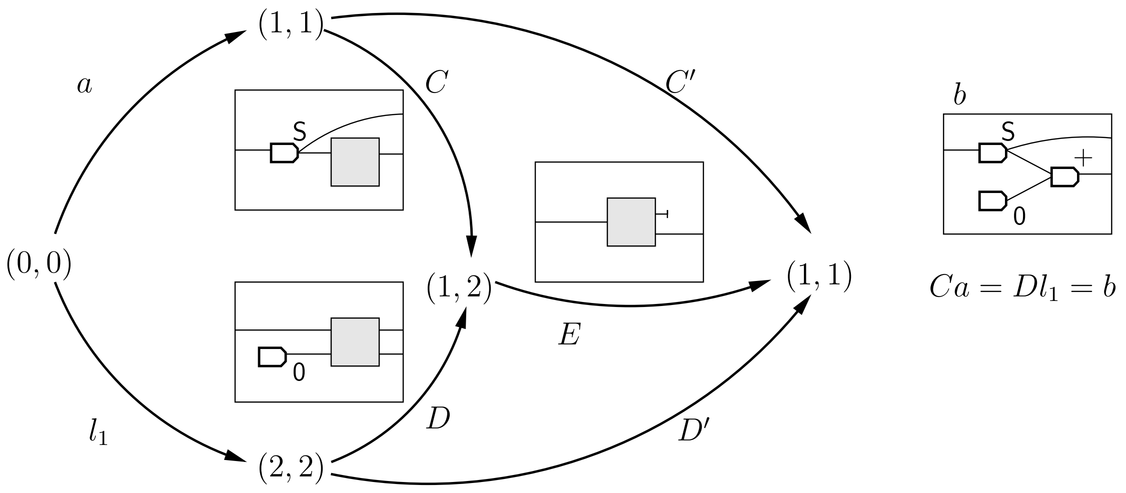
Example 4.2 (wiring) The preceding example used the forking and deletion of arcs to represent the sharing of components. This non-linearity is a pervasive feature in process calculi. CCS and the π-calculus depend heavily on it; witness the double occurrence of x in its reaction rule for CCS: $\bar{x}.a \mid x.b \longrightarrow a \mid b$ (and similarly for the π-calculus). The redex has no direct representation in the family of action calculi introduced in this subsection because we confine our attention to *shallow, closed* action graphs, i.e. ones without nesting of graphs (needed for prefixing) and free names. Without such limitations, the redex is exactly modelled by an action graph with a forked arc (representing the sharing of x) connected to two controls representing the output and input prefixes, containing inside a and b respectively. See [Mil96] for many examples.



Arithmetic rules Garbage collection rules



An action graph a overlapping a redex l_1



An RPO for a and l_1 w.r.t. C' and D'

Figure 12: A reactive system for arithmetic (Example 4.1)

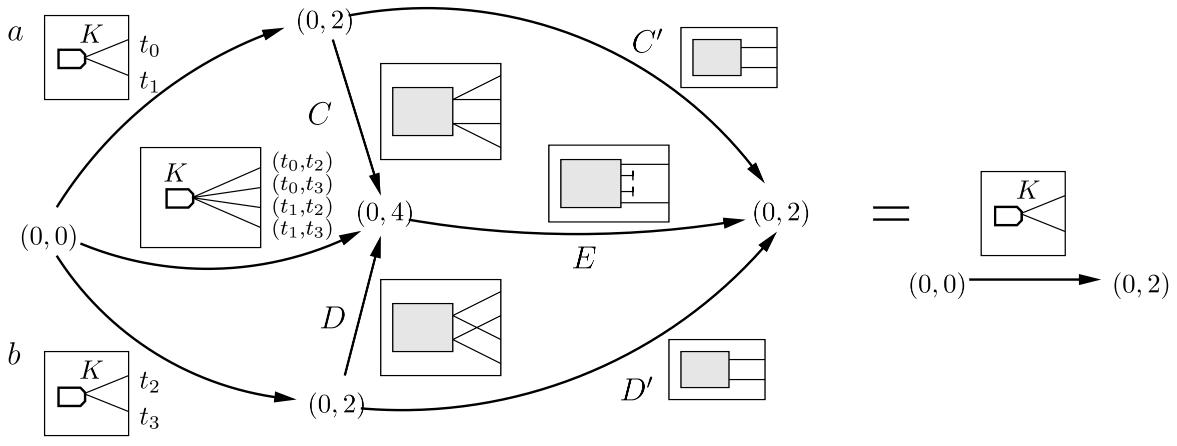


Figure 13: An RPO for copied wiring (Example 4.2)

Non-linearity can give rise to RPOs which are more complex than one might expect. Figure 13 shows two identical action graphs $a = b = K{\cdot}\mathsf{c}$, where $K : (0, 1)$; using the identity contexts $C' = D' = -_{0,2}$ they are embedded in $K{\cdot}\mathsf{c}$. But the RPO C, D, E does not consist of identity contexts! A candidate might choose to identify t_0 in a with either t_2 or t_3 in b, and similarly for t_1. To be the "best" candidate, the C, D, E must handle all these pairings; to indicate this we have indexed its targets by pairs in the diagram. In fact we have

$$Ca = Db = K{\cdot}\mathsf{c}{\cdot}(\mathsf{c} \otimes \mathsf{c}) \ .$$

Example 4.3 (reflexion) A surprising phenomenon is how the presence of reflexion can affect the RPO. Let $K, N : (1, 1)$, $L : (0, 2)$ and $M : (2, 0)$, and recall that i_1 is the identity of arity $(1, 1)$ for action graph composition. Figure 14 shows $a = L{\cdot}(\mathsf{i}_1 \otimes K)$ and $b = (\mathsf{i}_1 \otimes K){\cdot}M$ embedded in $C'a = D'b = L{\cdot}(N \otimes K){\cdot}M$. The contexts C' and D' do not involve reflexion. In the RPO C, D, E shown we have $Ca = Db = (\mathsf{i}_1 \otimes L){\cdot}(\mathsf{p} \otimes K){\cdot}(\mathsf{i}_1 \otimes M)$; this extends a by only one control (M) in order to create an occurrence of b. The contexts C and D do not use reflexion, but E *does* use it. If reflexion is forbidden then the RPO C^+, D^+, E^+ is such that $C^+a = D^+b$ contains N; this would yield a more complex synthesised labelled transition relation.

These examples do not exhaust the phenomena which arise in finding RPOs in **C-Ixt**, but they indicate that the general construction will not be trivial. The reader may feel that, having coped informally with a number of phenomena, we are well on the way to finding RPOs in every case. However, they do not always exist in **C-Ixt**! Here is a counter-example.

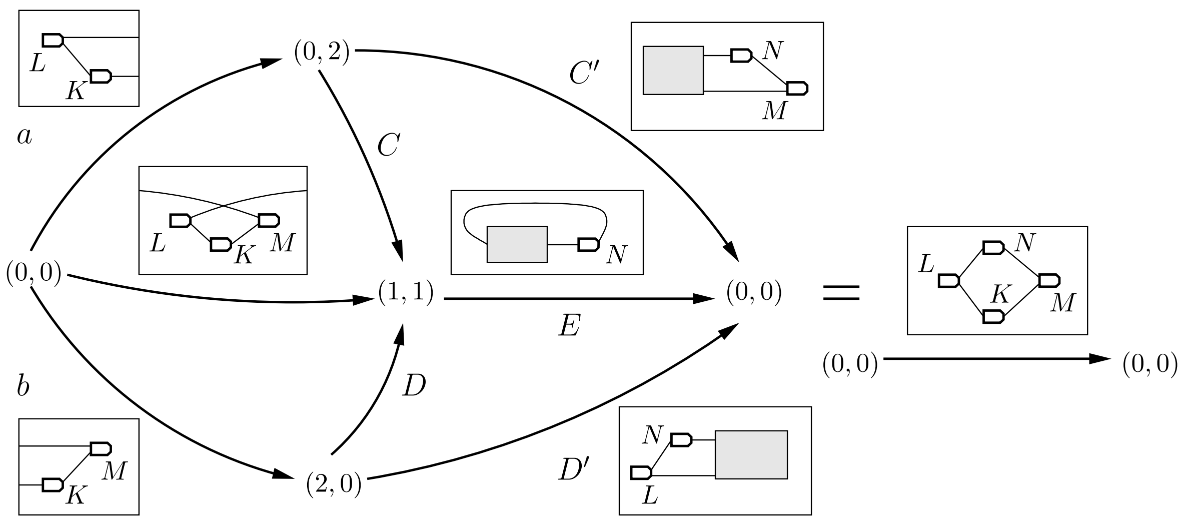


Figure 14: An RPO using reflexion (Example 4.3)

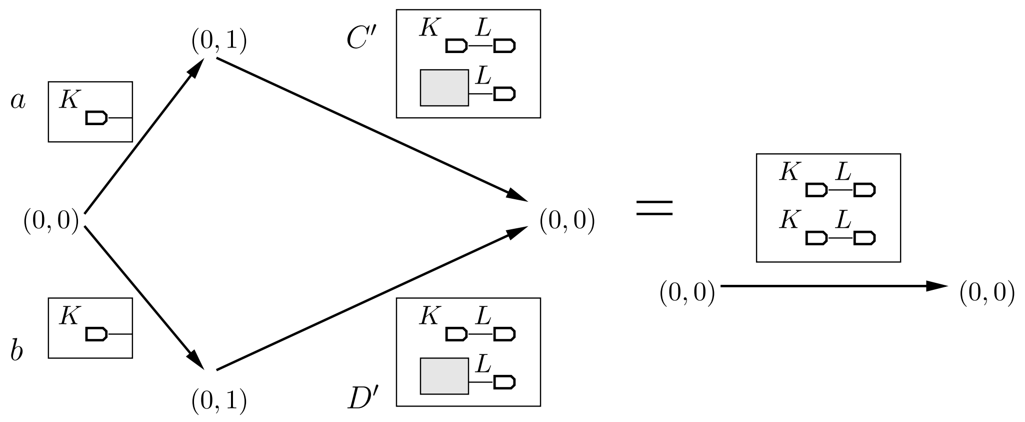
Example 4.4 (missing RPO) Let $K : (0,1)$ and $L : (1,0)$. Let $a = b = K$ and let $C' = D' = K{\cdot}L \otimes -_{0,1}{\cdot}L$ with arity $(0,1) \rightarrow (0,0)$. Then $C'a = D'b = K{\cdot}L \otimes K{\cdot}L$; this is shown in the upper diagram of Figure 15. The lower diagram shows two candidate RPOs, for which it is easy to verify commutation:

$$\begin{array}{rcccc} C_0, D_0, E_0 & = & -, & -, & K{\cdot}L \otimes -{\cdot}L \\ C_1, D_1, E_1 & = & - \otimes K, & K \otimes -, & -{\cdot}(L \otimes L)\ . \end{array}$$

But if there were an RPO, then contexts C, D, F_0, F_1 would exist as shown making the diagram commute, yielding a contradiction as follows: Neither C nor D can contain any control, since $F_0C = F_0D = -$. Hence from $CK = DK$ we deduce $C = D$, using the criterion for context equality stated above (since the control K appears in neither C nor D). Hence $- \otimes K = F_1C = F_1D = K \otimes -$, a contradiction.

This counter-example involves neither copying nor discard of arcs, so is applicable to *linear* action calculi too [LM02] — those in which all source ports bear exactly one arc. Thus we cannot attribute the lack of RPOs to c and ω, even though they demand careful treatment as shown in Example 4.1 and Example 4.2.

The counter-example also illustrates *why* RPOs do not always exist in **C-Ixt**. The equations $C_0K = D_0K = K$ and $C_1K = D_1K = K \otimes K$ hold respectively for the two candidates; but if we “track” the two *occurrences* of $a = K$ and $b = K$ through these equations, we find that they correspond to the *same* occurrence of K in the first case, and to two *different* occurrences in the second case. This is a key to solving our problem; we seek a suitable refinement of **C-Ixt** that possesses the RPOs we need. We expect its



A context equation $C'a = D'b$

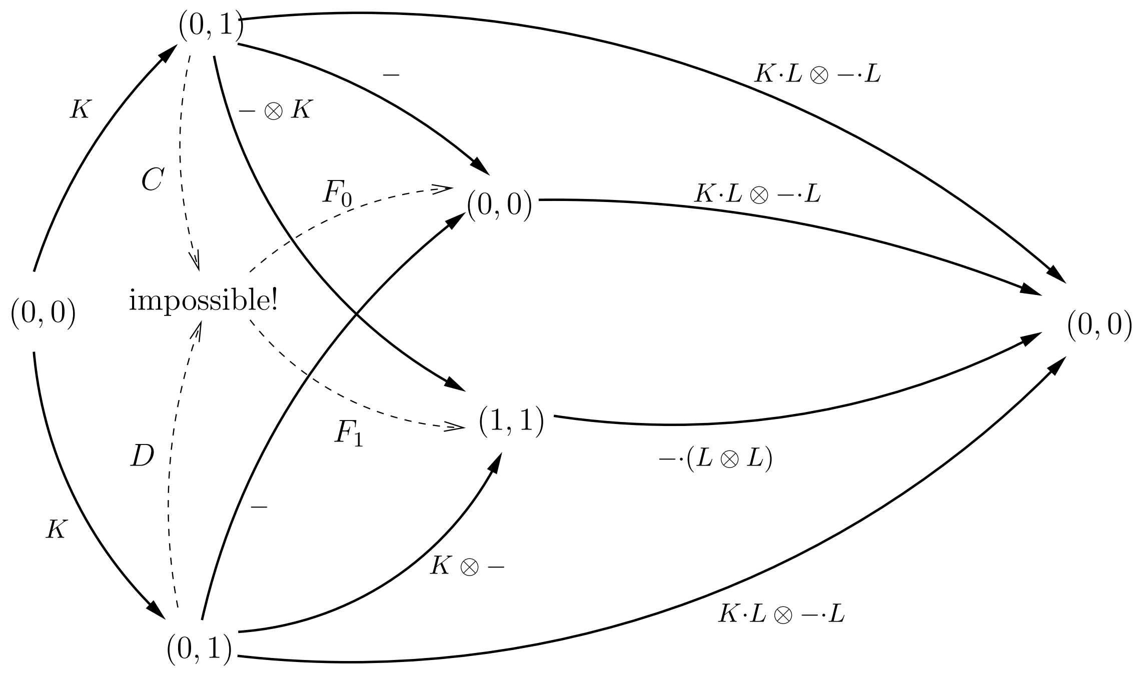


Figure 15: A missing RPO in **C-Ixt** (Example 4.4)

contexts to track the occurrences of nodes. This is a similar idea to that of “colouring”, which has been suggested by Sewell to replace his dissection based definitions [Sew].

The strategy is as follows. The next two subsections are devoted to the construction of **A-Ixt**, a well-supported precategory of arities and raw contexts. Replaying the constructions of Section 3, we derive two categories from **A-Ixt** and a functor between them $\mathcal{F} : \hat{\mathbf{C}}\textbf{-Ixt} \rightarrow \mathbf{C}\textbf{-Ixt}$. Finally we state the RPO existence theorem for $\hat{\mathbf{C}}\textbf{-Ixt}$ (Theorem 4.16), thus showing that $\mathcal{F}$ has all redex-RPOs (Theorem 4.17).

4.4 Closed shallow action graphs

In this subsection, after introducing some notation, I define a class of action graphs. These graphs are enriched in the next subsection to form *raw contexts* (graphs with a hole in them), the arrows of the precategory **A-Ixt**. No attempt is made to verify formally that the action graphs presented here correspond to action calculi terms quotiented by the action calculi axioms (which are not presented here). Such an undertaking represents future work and will be of more value when the tractable graph-theoretic features include free names and nesting.

Notation I write $[m]$ for the ordinal number $\{0, 1, \ldots, m-1\}$. The disjoint sum $\sum_{i \in I} X_i$ of a family of sets is taken to be the set $\bigcup_{i \in I}(\{i\} \times X_i)$. A particular case is when $I = [n]$; then the disjoint sum may be written, without parentheses, as $X_0 + X_1 + \cdots + X_{n-1}$. Examples in this section take the form $S = \sum_{v \in V} S_v + [m] + [n]$, a ternary disjoint sum, the first summand of which is itself a disjoint sum; S has elements of the form $(0, (v, s))$ for each $v \in V$ and $s \in S_v$, $(1, i)$ for each $i \in [m]$, and $(2, j)$ for each $j \in [n]$.

If the sets X and Y are disjoint, I often write their union as $X \uplus Y$. This notation is to be distinguished from a disjoint sum. In particular, $(X_1 \uplus X_2) \uplus X_3 = X_1 \uplus (X_2 \uplus X_3)$ and will often be written without parentheses; on the other hand, the disjoint sums $X_1 + X_2 + X_3$, $(X_1 + X_2) + X_3$ and $X_1 + (X_2 + X_3)$ are all distinct but in bijective correspondence. If $f : X \rightarrow Z$ and $g : Y \rightarrow Z$ are two functions with X disjoint from Y, then $f \uplus g : X \uplus Y \rightarrow Z$ is their union.

I use $f : X \rightarrowtail Y$, $f : X \twoheadrightarrow Y$ and $f : X \rightarrowtail\!\!\!\!\!\twoheadrightarrow Y$ for respectively injective, surjective and bijective functions, and $f : X \hookrightarrow Y$ for an injection which is an inclusion. Finally, $\circ$ denotes function composition, Id_X the identity function on the set X, and $\varnothing_X$ the empty function from $\varnothing$ to X.

Definition 4.5 (control signature) We fix a control signature $\mathcal{K}$, a set of *controls*, equipped with an *arity function* called *arity* : $\mathcal{K} \rightarrow \mathbb{N}^2$ and let $K, L, \ldots$ range over $\mathcal{K}$. For *arity*$(K) = (m, n)$ we write $K : (m, n)$; in this case, two functions extract the components of the pair: *arin*$(K) \mathrel{\hat{=}} m$ and *arout*$(K) \mathrel{\hat{=}} n$. ■

Definition 4.6 (action graphs) A *(closed, shallow) action graph* $G = (m, n, V, \mathit{contr}, \mathit{src})$ comprises an arity (m, n), a set V of *nodes*, called a *support*, a *control* map $\mathit{contr} : V \rightharpoonup \mathcal{K}$ assigning a control in $\mathcal{K}$ to each node in V, and a *source* map $\mathit{src} : T \rightharpoonup S$ assigning a *source (port)* in S to each *target (port)* in T, where

- the *source set* $S \mathrel{\hat{=}} \sum_{v \in V}[\mathit{arout}(\mathit{contr}(v))] + [m]$ comprises the *binding* sources for each $v \in V$ and the *input* sources indexed by $[m]$;
- the *target set* $T \mathrel{\hat{=}} \sum_{v \in V}[\mathit{arin}(\mathit{contr}(v))] + [n]$ comprises the *argument* targets for each $v \in V$ and the *output* targets indexed by $[n]$. ■

Nomenclature We may write a graph as $G = (V, \mathit{contr}, \mathit{src}) : (m, n)$, or just $G = (V, \mathit{contr}, \mathit{src})$ when the arity is understood. We denote the empty graph $(\varnothing, \varnothing_{\mathcal{K}}, \varnothing_{\varnothing}) : (0, 0)$ by $\mathbf{0}$. We shall abbreviate $\mathit{arin}(\mathit{contr}(v))$ to $\mathit{arin}(v)$ etc., when there is no ambiguity. We denote the injections induced by the disjoint sums S and T as follows:

$\mathit{bind}(v) : [\mathit{arout}(v)] \rightarrowtail S$	for the binding sources of each $v \in V$;
$\mathit{in} : [m] \rightarrowtail S$	for the input sources;
$\mathit{arg}(v) : [\mathit{arin}(v)] \rightarrowtail T$	for the argument targets of each $v \in V$;
$\mathit{out} : [n] \rightarrowtail T$	for the output targets. ■

We shall write $\mathit{bind}(v, i)$ and $\mathit{arg}(v, j)$ for the ports $\mathit{bind}(v)(i)$ and $\mathit{arg}(v)(j)$. For any injection f into a set A we write A^f for the range of f; thus for example S^{in} is the set of all input sources and $T^{\mathit{arg}(v)}$ the set of argument targets of v. We shall also write for example S^{bind} for $\biguplus_{v \in V} S^{\mathit{bind}(v)}$. With this notation we can represent our partitions as

$$S = S^{\mathit{in}} \uplus S^{\mathit{bind}}$$
$$T = T^{\mathit{out}} \uplus T^{\mathit{arg}} \ .$$

An example of an action graph with arity $(1, 3)$ is shown in Figure 16, with node names and the control map omitted. The whole graph is in a rectangle, with input sources at the left and output targets at the right. Nodes are drawn as rectangles with two corners blunted to give orientation; we may want to tilt some of them, as here, or even turn them upside down. The three nodes have arities $(1, 1)$, $(2, 1)$ and $(1, 2)$. The arcs represent the source map, with arrows pointing from source to target. Ports could be drawn as blobs, but this is somewhat redundant; a target is always indicated by exactly one incoming arc, and we indicate a source with no outgoing arcs by a little aborted arc.

Cycles are allowed. The graphs studied here are for action calculi which are *closed*, meaning that free names such as $x, y, \ldots$ are not used as sources, and *shallow*, meaning that nodes do not contain action graphs nested inside. I study the closed, shallow action

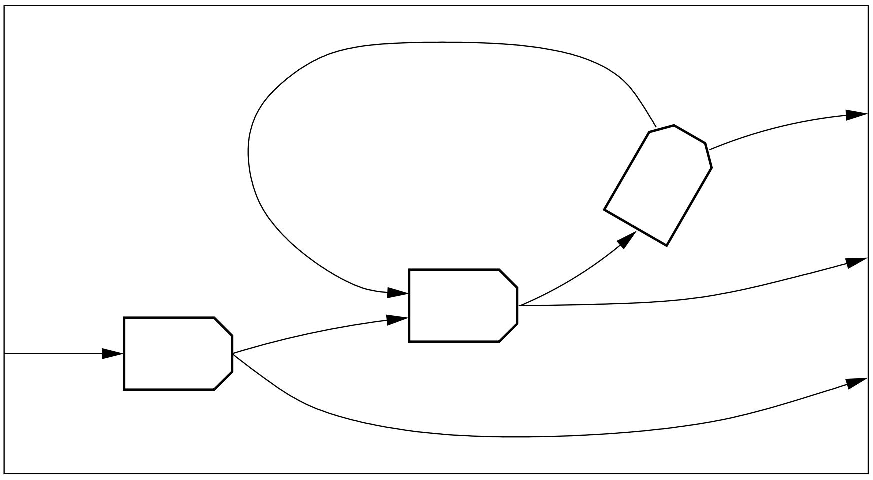


Figure 16: A closed shallow action graph

graphs in this paper as a preliminary to future work on a full graphical presentation of action calculi, since notions such as graph contexts are more clearly handled in this simpler setting.

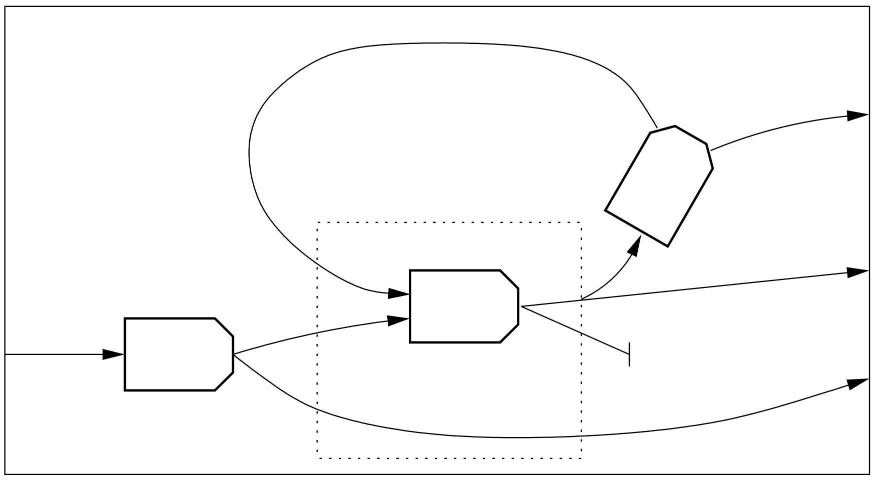
Convention Action graphs are almost always denoted by G suitably subscripted. Rather than always explicitly list all their primary components V, *contr*, and *src*, or their derived components S, T, *bind*, *in* etc., we shall follow a firm convention that the names of these components are standard, decorated as necessary to connote the graph's name.

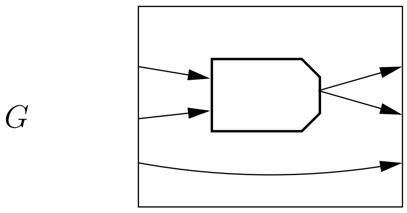
4.5 The well-supported precategory A-Ixt of arities and raw contexts

I proceed now to construct the well-supported precategory **A-Ixt** of raw contexts with a support function and support translation operation (Definition 3.2).

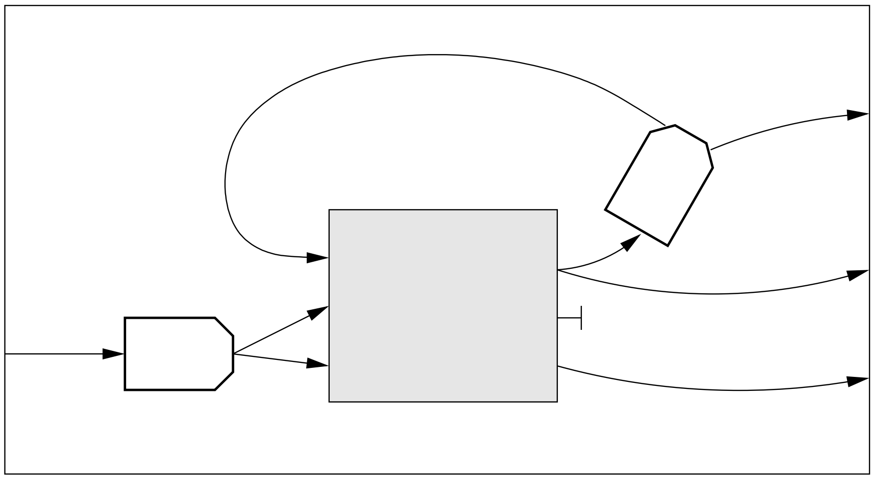
The intuition of "context" is well-supplied by Figure 17; the graph G occurs *inside* the dotted rectangle in G', and we may think of the context in which G is placed as that part of G' lying *outside* the dotted rectangle. A context is therefore an action graph, but with a little more structure since it has an internal as well as an external interface. The internal interface is often called a *hole*. (I do not consider here contexts with more than one hole.)

The lower diagram shows the context C which results when G is excised from G'; in the notation of what follows, $G' = CG$ (C composed with G). Note the new targets and sources on respectively the left and right sides of C's internal interface; in particular,





The action graph G is a subgraph of G'



The corresponding context

Figure 17: The bottom context composed with the middle action graph yields the top action graph

the middle internal source lacks any targets and therefore represents the discard of the corresponding output target of G — or of any other graph — when placed in the hole.

Definition 4.7 (raw context) A *(closed, shallow, loose) raw context* $A = (V, \mathit{contr}, \mathit{src})$ of arity (m', n') to (m, n), written $A : (m', n') \rightarrow (m, n)$, comprises a support V, which is a set of nodes, a *control* map $\mathit{contr} : V \rightarrow \mathcal{K}$ assigning a control in $\mathcal{K}$ to each node in V, and a *source* map $\mathit{src} : T \rightarrow S$ assigning a *source (port)* in S to each *target (port)* in T, where

- the *source set* $S \mathrel{\hat{=}} \sum_{v \in V} [\mathit{arout}(v)] + [m] + [n']$ comprises the *binding* sources for each $v \in V$, the *input* sources indexed by $[m]$, and the *upput* sources indexed by $[n']$;
- the *target set* $T \mathrel{\hat{=}} \sum_{v \in V} [\mathit{arin}(v)] + [n] + [m']$ comprises the *argument* targets for each $v \in V$, the *output* targets indexed by $[n]$, and the *downput* targets indexed by $[m']$.

Furthermore, we require that the looseness condition is satisfied (see nomenclature defined below):

$$\mathit{src}(T^{\mathit{down}}) \cap S^{\mathit{up}} = \varnothing \; . \qquad \text{LOOSE}$$

The looseness condition precludes a "tight" loop from the back of the hole to the front, such as is formed by reflexion in action calculi. It does not preclude such a loop proceeding via a control, which indeed occurs in Figure 17; thus the contexts permit only limited reflexion, which simplifies the definition of composition. (See Section 5 for further discussion.)

Finally, $|A|$ is the support of A, i.e. V in this case. ■

Note that an action graph G is just a raw context A as above in which $m' = n' = 0$.

Nomenclature As for action graphs, there are induced injections for raw contexts as follows:

$\mathit{bind}(v) : [\mathit{arout}(v)] \rightarrowtail S$	for the binding sources of each $v \in V$;
$\mathit{in} : [m] \rightarrowtail S$	for the input sources;
$\mathit{up} : [n'] \rightarrowtail S$	for the upput sources;
$\mathit{arg}(v) : [\mathit{arin}(v)] \rightarrowtail T$	for the argument targets of each $v \in V$;
$\mathit{out} : [n] \rightarrowtail T$	for the output targets;
$\mathit{down} : [m'] \rightarrowtail T$	for the downput targets.

With naming similar to that in action graphs for the partitions of S and T, we have

$$\begin{aligned} S &= S^{\mathit{bind}} \uplus S^{\mathit{in}} \uplus S^{\mathit{up}} \\ T &= T^{\mathit{arg}} \uplus T^{\mathit{out}} \uplus T^{\mathit{down}} \; . \end{aligned}$$

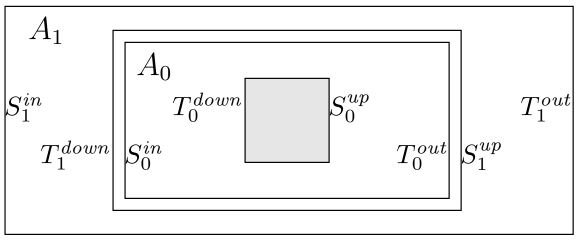


Figure 18: The composition of two raw contexts

Raw contexts cannot be composed when the supports of each operand in a composition intersect non-trivially. Composition *is* well-defined when the operands have disjoint supports— one of the required properties of a well-supported precategory (Definition 3.2).

Definition 4.8 (identity and composition for raw contexts) The *identity* raw context $\mathsf{id}_{(m,n)} = (\varnothing, \varnothing_{\mathcal{K}}, src) : (m,n) \rightarrowtriangle (m,n)$ has

$$S \mathrel{\hat{=}} \varnothing + [m] + [n] = S^{in} \uplus S^{up} \qquad\qquad src : \begin{cases} down(i) \mapsto in(i) & i \in [m] \\ out(j) \mapsto up(j) & j \in [n] \ . \end{cases}$$
$$T \mathrel{\hat{=}} \varnothing + [n] + [m] = T^{out} \uplus T^{down}$$

Let $A_i = (V_i, contr_i, src_i) : (m_i, n_i) \rightarrowtriangle (m_{i+1}, n_{i+1})$ be two raw contexts for $i = 0, 1$, with $V_0 \cap V_1 = \varnothing$. Their *composition*, illustrated in Figure 18 by nesting A_0 inside A_1, is

$$A_1 \oplus A_0 \mathrel{\hat{=}} A_2 = (V_2, contr_2, src_2) : (m_0, n_0) \rightarrowtriangle (m_2, n_2) \ ,$$

where $V_2 \mathrel{\hat{=}} V_1 \uplus V_0$ and $contr_2 \mathrel{\hat{=}} contr_1 \uplus contr_0$ (thus determining $arin_2 = arin_1 \uplus arin_0$ and likewise $arout_2, bind_2, arg_2$), and

$$S_2 \mathrel{\hat{=}} \textstyle\sum_{v \in V_2} [arout_2(v)] + [m_2] + [n_0] = (S_1^{bind} \uplus S_0^{bind}) \uplus S_1^{in} \uplus S_0^{up}$$
$$T_2 \mathrel{\hat{=}} \textstyle\sum_{v \in V_2} [arin_2(v)] + [n_2] + [m_0] \ = (T_1^{arg} \uplus T_0^{arg}) \uplus T_1^{out} \uplus T_0^{down} \ .$$

Note (see Figure 18) that T_1^{down} is in bijection with S_0^{in} and $[m_1]$, while S_1^{up} is in bijection with T_0^{out} and $[n_1]$. It remains to define the source function src_2; this is done in terms of two auxiliary functions $\sigma_i : S_i \rightarrowtriangle S_2$ $(i = 0, 1)$ which describe how sources of A_0 and A_1

"become" sources of A_2:

$$\sigma_0(s) \mathrel{\hat{=}} \begin{cases} s & \text{if } s \in S_0^{bind} \uplus S_0^{up} \\ src_1\, down_1(i) & \text{if } s = in_0(i) \in S_0^{in} \end{cases}$$

$$\sigma_1(s) \mathrel{\hat{=}} \begin{cases} s & \text{if } s \in S_1^{bind} \uplus S_1^{in} \\ \sigma_0 src_0\, out_0(j) & \text{if } s = up_1(j) \in S_1^{up} \end{cases}$$

$$src_2(t) \mathrel{\hat{=}} \begin{cases} \sigma_1 src_1(t) & \text{if } t \in T_1^{arg} \uplus T_1^{out} \\ \sigma_0 src_0(t) & \text{if } t \in T_0^{arg} \uplus T_0^{down} \end{cases} .$$

■

We have adopted the symbol "$\oplus$" for composition of raw contexts as a reminder that it is partial: the supports of the operands are required to be disjoint in order for a composition to be defined.

Proposition 4.9 (raw contexts form a precategory) If A_0 and A_1 are raw contexts with $|A_1| \cap |A_0| = \varnothing$ then $A_1 \oplus A_0$ is a raw context too. (In particular, $\oplus$ preserves the Loose condition.) Composition is associative and has identity id (in the way required of a precategory, see Definition 3.1).

Proof Follows immediately from Propositions 21 and 22 of [CLM00]. ■

By definition composition satisfies Supp-comp and Supp-id (Definition 3.2), i.e.

$$|A_1 \oplus A_0| = |A_1| \uplus |A_0| \qquad \text{Supp-comp}$$
$$|\mathsf{id}_m| = \varnothing . \qquad \text{Supp-id}$$

The only task remaining in order to show that **A-Ixt** is a well-supported precategory is the definition of a support translation operation and the verification of its properties.

Definition 4.10 (support translation for raw contexts) Given a raw context $A : (m_0, n_0) \rightarrow (m_1, n_1)$ and an injection ρ whose domain contains the support of A, i.e. $\mathsf{Dom}\,\rho \supseteq |A|$, we define the *support translation by ρ of A*, written $\rho \bullet A$, as follows: $\rho \bullet A \mathrel{\hat{=}} A'$, where:

$$\begin{aligned} V' &\mathrel{\hat{=}} \rho V \\ contr'(v) &\mathrel{\hat{=}} contr(\rho^{-1}(v)) \qquad \text{thus determining } arin, arout, bind, arg \\ S' &\mathrel{\hat{=}} \textstyle\sum_{v \in V'} [arout'(v)] + [m_1] + [n_0] \\ T' &\mathrel{\hat{=}} \textstyle\sum_{v \in V'} [arin'(v)] + [n_1] + [m_0] \end{aligned}$$

thus determining bijections $\rho^{\mathsf{S}} : S \rightarrowtail\!\!\!\!\twoheadrightarrow S'$ and $\rho^{\mathsf{T}} : T' \rightarrowtail\!\!\!\!\twoheadrightarrow T$ whence we define:

$$src' \mathrel{\hat{=}} \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}} .$$

■

This definition satisfies the appropriate healthiness conditions:

Proposition 4.11 (support translation is well-defined) If A satisfies LOOSE then so does $\rho{\scriptstyle\bullet}A$, thus $\rho{\scriptstyle\bullet}(\cdot)$ is a well-defined operation on raw contexts.

Proof Immediate from $src' \mathrel{\hat{=}} \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}}$, since ρ^{S} and ρ^{T} are the identity on upput sources and downput targets respectively. (Proposition 24 in [CLM00].) ■

Furthermore, support translation satisfies the remaining axioms in the definition of a well-supported precategory (Definition 3.2):

Proposition 4.12 (support translation properties) All of the following properties hold:

$$\begin{aligned}
\rho{\scriptstyle\bullet}\mathsf{id}_m &= \mathsf{id}_m && \textsc{Trans-id-r}\\
\rho{\scriptstyle\bullet}(A_1 \oplus A_0) &= \rho{\scriptstyle\bullet}A_1 \oplus \rho{\scriptstyle\bullet}A_0 && \textsc{Trans-comp-r}\\
\mathsf{Id}_{|A|}{\scriptstyle\bullet}A &= A && \textsc{Trans-id-l}\\
(\rho_1 \circ \rho_0){\scriptstyle\bullet}A &= \rho_1{\scriptstyle\bullet}(\rho_0{\scriptstyle\bullet}A) && \textsc{Trans-comp-l}\\
\rho_0 \restriction |A| = \rho_1 \restriction |A| \text{ implies } \rho_0{\scriptstyle\bullet}A &= \rho_1{\scriptstyle\bullet}A && \textsc{Trans-res}\\
|\rho{\scriptstyle\bullet}A| &= \rho|A| \ . && \textsc{Trans-supp}
\end{aligned}$$

Proof All of these except the last follow from Proposition 25 in [CLM00]. TRANS-SUPP follows immediately from Definition 4.10. ■

Thus:

Theorem 4.13 **A-Ixt** is a well-supported precategory. ■

4.6 Constructing a functorial reactive system

Having established that **A-Ixt** is a well-supported precategory, the rest of the results of Section 3 are immediately applicable, as we next see. We let $\hat{\mathbf{C}}$**-Ixt** be the track of **A-Ixt** (Definition 3.3), a category of profiles and *insertion contexts.* The arrows are called "insertion contexts" to remind us that they "insert" the nodes of the domain profile into the codomain profile.

Now let **C-Ixt** be the support quotient (Definition 3.10) of **A-Ixt**. Finally let $\mathcal{F} : \hat{\mathbf{C}}\textbf{-Ixt} \rightarrow \mathbf{C}\textbf{-Ixt}$ be the support-quotienting functor (Definition 3.12). Then by Theorem 3.14, $\mathcal{F}$ lifts arrows by their domain, creates isos, and creates compositions. By Theorem 3.15, $\mathcal{F}$ allows IPO sliding.

A-Ixt has a distinguished object $(0,0)$: as stated immediately after Definition 4.7, a raw context with domain $(0,0)$ is identical to an action graph. Recall that **C-Ixt** has the same objects as **A-Ixt** does, so the distinguished object 0 of **C-Ixt** (viewed as a reactive system) is just $(0,0)$. Likewise, let $\boldsymbol{\varepsilon} \mathrel{\hat{=}} (0,0,\varnothing)$, the distinguished object for $\hat{\mathbf{C}}$**-Ixt**. Since $\mathcal{F}(\boldsymbol{\varepsilon}) = 0$, we have that $\mathcal{F}$ lifts agents, and thus:

Theorem 4.14 ($\mathcal{F}$ is functorial reactive system) The support-quotienting functor $\mathcal{F} : \hat{\mathbf{C}}\textbf{-Ixt} \rightarrow \mathbf{C}\textbf{-Ixt}$ is a functorial reactive system for any choice of $\mathbf{D}$ and Reacts. ■

Thus all the congruence results of Section 3 in Part 1 are applicable to $\mathcal{F}$ (except Theorem 3.34 since we are not dealing with multi-hole contexts) with one proviso: we wish to determine choices of $\mathbf{D}$ and Reacts such that $\mathcal{F}$ has all redex-RPOs.

$$\begin{tikzcd}
\varepsilon \arrow[r, "G_0"] \arrow[d, "G_1"'] & p_0 \arrow[d, "A_0"] \\
p_1 \arrow[r, "A_1"'] & p
\end{tikzcd}$$
19

With regard to $\mathbf{D}$, the strongest result obtainable is with $\mathbf{D} = \mathbf{C}\textbf{-Ixt}$, i.e. with all contexts in $\mathbf{C}\textbf{-Ixt}$ reactive, so let us fix on this. Recall from Definition 2.6, that $\mathcal{F}$ has all redex-RPOs if any square in $\hat{\mathbf{C}}\textbf{-Ixt}$, such as in Figure 19, has an RPO provided that there exists $r \in \mathbf{C}\textbf{-Ixt}$ such that $(\mathcal{F}(\varepsilon \xrightarrow{G_1} p_1), r) \in \mathsf{Reacts}$.

So the key question is: for which choice of redexes l (first components of the Reacts relation) do there exist RPOs in $\hat{\mathbf{C}}\textbf{-Ixt}$ for squares such as Figure 19 where $\mathcal{F}(\varepsilon \xrightarrow{G_1} p_1) = l$. The answer is that for almost an any choice of redexes, except the most pathological, RPOs exist. In order to make precise which are these pathological cases, we need a definition about action graphs.

Definition 4.15 (OUTPUT-CONTROLLED) In any graph, a target is *controlled* if its source is a bound source. G is OUTPUT-CONTROLLED if all its output targets are controlled; formally:

$$src(T^{out}) \subseteq S^{bind} \ . \qquad (\text{OUTPUT-CONTROLLED})$$

Since $\mathsf{Cod}\, src = S^{in} \uplus S^{bind}$, an equivalent formulation is:

$$src(T^{out}) \cap S^{in} = \varnothing \ . \qquad (\text{OUTPUT-CONTROLLED})$$

■

Informally, a graph satisfies this definition when it contains no “floating” wires connecting the left-hand interface ports with the right-hand ones.

Recall that in $\hat{\mathbf{C}}\textbf{-Ixt}$ a context $\varepsilon \xrightarrow{G} (m, n, U)$, whose domain is the empty profile ε, is actually a graph $G : (m, n)$ with $|G| = U$. Thus the definition of OUTPUT-CONTROLLED applies to contexts in $\hat{\mathbf{C}}\textbf{-Ixt}$ with domain ε.

RPOs do exist provided one of the legs is OUTPUT-CONTROLLED:

Theorem 4.16 Let $C_0G_0 = C_1G_1 = G$ in $\hat{\mathbf{C}}\textbf{-Ixt}$, where G_1 is OUTPUT-CONTROLLED. Then there exists an RPO for G_0, G_1 w.r.t. C_0, C_1.

Proof See Corollary 6.28 in [Lei01b]. ■

Now note that if G is OUTPUT-CONTROLLED then $\rho \bullet G$ is too for any injection ρ with $\mathsf{Dom}\, \rho \supseteq |G|$. Thus it is well-defined to say that a context with domain $(0, 0)$ in $\mathbf{C}\textbf{-Ixt}$ (the category formed by quotienting out support translations) is OUTPUT-CONTROLLED.

Because Theorem 4.16 is asymmetric in the use of the OUTPUT-CONTROLLED hypothesis, it directly implies the following.

Theorem 4.17 ($\mathcal{F}$ has all redex-RPOs) The functorial reactive system $\mathcal{F} : \hat{\mathbf{C}}\textbf{-Ixt} \rightarrow \mathbf{C}\textbf{-Ixt}$ has all redex-RPOs provided that every redex is OUTPUT-CONTROLLED, i.e. provided that for all $(l, r) \in \mathsf{Reacts}$, l is OUTPUT-CONTROLLED. ■

Happily, the requirement that redexes be OUTPUT-CONTROLLED is a tolerable one since redexes that are not are pathological (see Chapter 7 in [Lei01b]). Finally, Theorem 4.17 implies that all of the congruence results of Section 3 in Part 1 (except the one for multi-hole contexts) are immediately applicable to $\mathcal{F} : \hat{\mathbf{C}}\textbf{-Ixt} \rightarrow \mathbf{C}\textbf{-Ixt}$, as desired.

5 Conclusions and future work

Constructing RPOs is difficult, as witnessed by the daunting complexity in Chapter 6, "RPOs for action graph contexts", in [Lei01b] (the proof of Theorem 4.17 in the present paper). It is therefore desirable that it not be done afresh for each new process calculus. This is the impetus for the leading example in this paper, namely action calculi-like contexts. Recall that the functorial reactive system $\mathcal{F} : \hat{\mathbf{C}}\textbf{-Ixt} \rightarrow \mathbf{C}\textbf{-Ixt}$ is really a family with varying choices of control signature $\mathcal{K}$ and of reaction rules Reacts. The fragment of action calculi considered in Section 4 is thus a *framework*: *if* one can express states of a computational model using controls (for atoms) and arcs (for names and sharing) and reactions in terms of redexes that satisfy the condition OUTPUT-CONTROLLED *then* RPOs exist and hence Section 3 of Part 1 gives labelled transitions and operational congruences.

A framework is often deficient for two reasons: (i) it provides insufficient benefits to compensate for the effort of casting a specific example in the framework's form; (ii) it is not rich enough to cover all the phenomena that one wants. Criticism (ii) hits home. The fragment of action calculi presented in this paper is inadequate for most purposes: there is no nesting of action graphs (needed for prefixing), no free names, no binding, limited reflexion, no choice operator, and no multi-hole redexes (needed to represent faithfully metavariables in reaction rules). I describe future work below that addresses these other issues. Criticism (i), however, is unjustified when applied to the action calculi shown here: getting "for free" a labelled transition system and a collection of operational congruences is a worthwhile benefit.

The reverse situation held for action calculi as originally presented by Milner [Mil96]: criticism (ii) was not such a problem but criticism (i) was: up until now there has been little tangible reward to using action calculi to present specific examples of process calculi.

The central goal of future work is to construct a framework with rich enough structure (perhaps like Milner's original action calculi) to dodge criticism (ii) while still providing the rewards of labelled transitions and operational congruences.

To this end, it will be important to pursue several research directions:

multi-hole redexes: The basic reaction rule of the π-calculus is

$$\bar{x}\langle y\rangle.a \mid x(z).b \quad \longrightarrow \quad a \mid \{y/z\}b \ . \tag{3}$$

There is a problem of adequately representing (3) in a reactive system. Currently I require that the reaction rules Reacts consist of pairs of *agents*, not *agent contexts*. As a result, the only way to represent (3) is via an infinite family of rules, one for each possible instantiation of a and b. Thus there might be labelled transitions of the form

$$\bar{x}\langle y\rangle \xrightarrow{-|x(z).b} \tag{4}$$

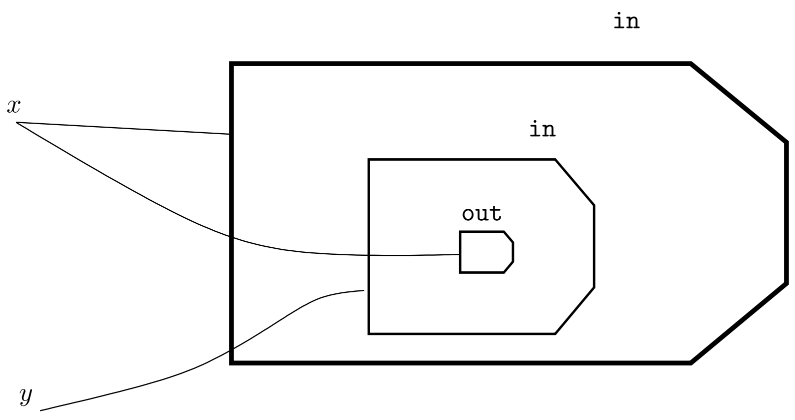


Figure 20: Nesting of nodes

for all instantiations of b. But these are ungainly labels: the only important part of any label of the form $- \mid x(z).b$ is the top-level structure, namely $x(z)$; the b is irrelevant.

How can we winnow down the family of labels to a single one? The answer lies in using the uniformity present in (3). The a and b there are really metavariables and the rule can be expressed thus:

$$\bar{x}\langle y\rangle.-_1 \mid x(z).-_2 \quad \longrightarrow\!\!\!\triangleright \quad -_1 \mid \{y/z\}-_2 \;. \tag{5}$$

This rule consists of a pair of 2-hole contexts which capture the uniformity present in it. In Figure 13 of [Mil01], Milner proposes a formalisation of exactly this rule in his bigraphical reactive systems, which cater directly for multi-hole contexts.

If we relax the condition on Reacts and allow it to contain pairs of contexts (L, R) for which $L, R : m \rightarrow n$ are arbitrary contexts, then it is easy to generate the reaction relation $\longrightarrow\!\!\!\triangleright$ (forgetting about the details of $\mathbf{D}$)

$$A \longrightarrow\!\!\!\triangleright A' \quad \text{iff} \quad (\exists (L,R) \in \mathsf{Reacts}, C, D.\ A = DLC \;\&\; A' = DRC)\;.$$

Contrast this to Proposition 2.7 for which the reaction rules are closed-up under context application on the outside, not also on the inside (as shown here). The hard problem is to define labelled transitions $A \xrightarrow{F}\!\!\triangleright A'$. We cannot rely on IPOs anymore since the domain of A is potentially different from the domain of any redex L. A possible replacement for IPOs might be via *hexagon idem pushouts* (which are special kinds of *hexagon relative pushouts*) as suggested by Sewell. Notice how L and A are wrapped up in Figure 21 by arrows

$$\begin{tikzcd}
 & \cdot \arrow[r, "A"] & \cdot \arrow[d, "F'"'] \arrow[dr, "F"] & \\
\cdot \arrow[ur, "B"] \arrow[r] \arrow[dr, "C"'] & \cdot \arrow[u, "B'"'] \arrow[d, "C'"] & \cdot \arrow[r] & \cdot \\
 & \cdot \arrow[r, "L"'] & \cdot \arrow[u, "D'"] \arrow[ur, "D"'] &
\end{tikzcd}$$

21

composed on either side. The hexagon idem pushout property guarantees that there is no lesser hexagon bounded by B', C', F', D'. It seems plausible that such hexagons can yield the single labelled transition $\bar{x}\langle y\rangle \xrightarrow{-_1|x(z).-_2}$ from (5), which is lighter than having the infinite family shown in (4).

In [Sew], Sewell already exploited the uniformity present in reaction rules when looking at term algebras with parallel composition. He achieved lightweight labelled transitions, but made use of complex context dissection lemmas. He is now looking at the problem of recasting his results in terms of universal constructions (such as hexagons). In conclusion, there are two important lines of research related to multi-hole redexes which mirror what was done in my paper: (i) defining labelled transitions by universal constructions and proving congruence for a variety of equivalences and preorders; (ii) showing that the relevant universal constructions exist in categories of interest (term algebras, graphs, etc.).

nesting: The nesting of agents inside nodes is essential for representing prefixing. For example, in CCS the agent $x.y.\bar{x}$ contains three levels of nesting with the name x shared by the outermost and the innermost. In order to represent this with graph-like constructions, we need that each node packages up the continuation agent, e.g. as in Figure 20 which gives a possible representation of the CCS agent above.

At first glance, nesting is a trivial thing: term algebras, for instance, consists of just pure nesting structure with nothing else. It is straightforward to calculate RPOs for them. However the real challenge is the combination of wiring and nesting: Figure 20 has an arc sourced by x that is forked and has targets at multiple levels. In particular, even the notion of context composition is difficult: when composing contexts that fork an arc at different nesting levels, the forks have to be normalised in some way, either by pushing them in to the innermost level or pulling them out to the outermost.

There are many possible representations for nested graphs. One that seems most promising comes out of recent work by Milner [Mil01]: the idea is to treat a nested graph as a flat graph with an added parent function overlaid, a so called *topograph* structure. RPOs can then be calculated by handling the wiring structure and the nesting structure orthogonally.

Even without the full power of nesting, it may be possible to handle the π-calculus (or variants thereof) indirectly. Honda and Yoshida [HY94a] give a translation in terms of "concurrent combinators"; these encode prefixing in a flat structure by using triggers to point to parts of agents that may execute. A synthesised labelled transition relation for their combinator reaction rules might be too intensional since the labels could detect exactly the combinators present in an agent. It is worth trying this, though, to see what kind of equivalences would transpire.

free names and binding: It is straightforward to add free names to graphs: the idea is to enrich the set of source ports to include names. It is more subtle though to make precise how a context can *bind* the names of the thing that goes in its hole. It seems likely that the objects of the category of contexts need to contain name sets to ensure that RPOs exist (analogously to inclusion of node sets in $\hat{\mathbf{C}}\text{-}\mathbf{Ixt}$), i.e. a context is an arrow of the form $(m, n, U, X) \rightarrow (m', n', U', X')$, meaning that its hole is expecting a context with support (nodes) U and names X. (This is similar to Definition 67 in Milner's work on bigraphical reactive systems [Mil01].) It is not clear whether the downstairs category (the analogue of $\mathbf{C}\text{-}\mathbf{Ixt}$) should include these name sets. For example, in π-calculus we write the binding context $(\nu x)-$ without saying which names can fit in the hole. Perhaps the functor from upstairs to downstairs (like $\mathcal{F} : \hat{\mathbf{C}}\text{-}\mathbf{Ixt} \rightarrow \mathbf{C}\text{-}\mathbf{Ixt}$ in this paper) will allow two kinds of IPO sliding: the first with respect to nodes (as I have shown here) and the second with respect to names.

There has been recent work on modelling names and name binding in a categorical way [TFP99, GP99] and it would be exciting to join together those syntactic models with operational semantics, and in particular, the reactive systems shown here.

Name binding carries with it the problem of *scope extrusion* in labelled transitions. In π-calculus, for example, it is a subtle problem to handle the labelled transition of $(\nu x)(\bar{y}\langle x\rangle)$. There are many ways of handling this when looking at bisimulation, for example, but most involve augmenting the definition with freshness side conditions to cater explicitly for extrusion. An alternate approach [CS00] keeps track of extruded names as annotations of the agents themselves, thus gaining the power of extrusion but keeping the simplicity of bisimulation without added conditions. Adding these annotations seems almost identical to adding name sets in the objects of a context category (as outlined above).

sum: As discussed in Subsection 3.5 in Part 1, the proper treatment of summation (such as occurs in CCS and the π-calculus) requires care. It is not good enough for sum to be a free operator (just another control) if we want to describe its reactive behaviour, e.g.

$$\begin{aligned} (\bar{x}.a + b) \mid (x.a' + b') &\quad \longrightarrow\!\!\triangleright \quad a \mid a' \\ \tau.a + b &\quad \longrightarrow\!\!\triangleright \quad a\ . \end{aligned}$$

Consider the second reaction rule. In order for the sum to be rearranged so that $\tau.a$ is on the left and the rest is on the right, we require $+$ to be commutative, associative, and have the empty agent as an identity.

The same problem surfaces for other operators that change the structural congruence (syntactic equality). In the π-calculus, for example, replication is handled by the axiom

$$!\,a \;\equiv\; a \mid !\,a\;. \tag{6}$$

This seems hard to represent in terms of graphs since the RHS has, potentially, more nodes than the LHS. When the problems of finding graph theoretic representation of sum and replication are overcome (see Milner's proposal in Figures 12 and 13 of [Mil01]), it may still be difficult to construct RPOs. A possible solution, at least for replication, is to disallow the structural equality in (6), and instead confine replication to be *input guarded* and to have a reaction rule of the form

$$\bar{x}\langle y\rangle \mid !\,x(z).a \quad\longrightarrow\quad \{y/z\}a \mid !\,x(z).a\;.$$

This is a commonly done in asynchronous variations of π-calculus [HT91].

Another way of handling replication is via an encoding in terms of combinators (already mentioned above) [HY94b]. For input guarded summation, encoding [NP96] is also an option.

full reflexion: The contexts considered in Section 4 have only a limited form of reflexion, as enforced by the condition Loose that prevents tight loops linking a hole to itself. This simplifying assumption makes the composition of contexts and the composition of the associated embeddings easy. (Embeddings, which are morphisms between graphs, are not introduced in this paper but play a central role in the construction of RPOs; see Section 6.2 in [Lei01b].) With full reflexion, the composition of contexts can create arcs that are formed from arbitrarily many segments of arcs present in the inner and outer context. Indeed, it is difficult to prove even that the composition of reflexive embeddings is associative [Lei01a]. Better technology for handling reflexion has been developed in [LM02] for graphs with linear wiring. Proving the existence of RPOs for full reflexion with non-linear wiring represents a future challenge.

inductive presentation of labelled transitions: In Part 1, the way of deriving labelled transitions from reactions yields a direct characterisation. This is in contrast to the inductive presentation usually found in the process calculus literature. Can the gap be bridged, though? In other words, can an inductive presentation of labelled transitions be synthesised from reaction rules? It seems plausible. For any particular reactive system whose syntax is generated from some constructors, one could instantiate Lemma 2.17 of Part 1 (recalled here) by substituting each constructor for C:

$$\frac{\begin{tikzcd}[sep=small] \cdot \arrow[r, "C"] \arrow[d, "F'"'] & \cdot \arrow[d, "F"] \\ \cdot \arrow[r, "C'"'] & \cdot \end{tikzcd} \text{ is an IPO} \qquad a \xrightarrow{F'} a'' \qquad C' \in \mathbf{D}}{Ca \xrightarrow{F} C'a''}\;.$$

There are several interesting questions that follow: Under what conditions can we synthesise an inductive presentation that generates precisely the same labelled transition relation as the direct presentation gives? Under what conditions would an inductive presentation satisfy any of the GSOS rule formats [GV92, Blo93]? If some set of the latter is satisfied, it would be enlightening to compare two different proofs of congruence for strong bisimulation, say — one based on the RPO theory shown in this paper and the other based on GSOS reasoning, particularly as provided by recent categorical treatments of the latter [TP97].

It is not surprising that some of these areas (e.g. free names and multi-hole redexes) require changes to the categorical abstractions of a reactive system, not just cleverer ways of constructing RPOs. This pattern is well-established in this series of papers. I started with reactive systems, then passed to functorial reactive systems when it became clear that RPOs needed to be calculated in a separate category, then considered functorial monoidal reactive system to cater explicitly for RPOs resulting in multi-hole contexts. As argued in Subsection 3.7, it may be advantageous to make an orthogonal extension, replacing categories by bicategories throughout. An intriguing possibility is that this extension is not truly "orthogonal", i.e. the power of enriched category theory could be brought to bear on the problems listed above.

A test of the future work outlined above is to see what labelled transitions and operational congruences are generated for applications such as CCS, π-calculus, λ-calculus, and ambients. It would be exciting (though not likely) to get congruences that coincide exactly with well-known ones (open bisimulation for π-calculus, say). It should not be a cause for dejection if the derived congruences do not match exactly the historically established ones: π-calculus comes with a zoo of bespoke congruences, no single one of which is good for all applications.

The best outcome of this work will be if the mathematical tools that are proposed here ease the path for exploring new primitives and new patterns of behaviour inherent in distributed computing. The development of mathematical techniques for reasoning about hosts and host failure, agent mobility, and cryptography, for example, is critical for guiding infrastructure and language design. Proofs techniques for these models are critical for precise reasoning. Much work needs to be done before the tools of this paper are good enough to address these applications.

References

Curly braces enclose pointers back to the pages in this paper that cite the work.

[BF00] M. Bunge and M. P. Fiore. Unique factorisation lifting functors and categories of linearly-controlled processes. *Mathematical Structures in Computer Science*, 10(2):137–163, 2000. {17}

[Blo93] B. Bloom. Structural operational semantics for weak bisimulations. Technical Report TR-93-1373, Department of Computer Science, Cornell University, August 1993. {45}

[CLM00] G. L. Cattani, J. J. Leifer, and R. Milner. Contexts and embeddings for closed shallow action graphs. Technical Report 496, Computer Laboratory, University of Cambridge, July 2000. Available from http://pauillac.inria.fr/∼leifer/. {23, 36, 37}

[Con72] F. Conduché. Au sujet de l'existence d'adjoints à droite aux foncteurs "image réciproque" dans la catégorie des catégories. *Comptes rendus de l'Académie des sciences A*, pages 891–894, 1972. {17}

[CS00] G. L. Cattani and P. Sewell. Models for name-passing processes: interleaving and causal. In *15th Annual IEEE Symposium on Logic in Computer Science, 26–29 June 2000, Santa Barbara, California, USA*, pages 322–332. IEEE Press, 2000. {43}

[GP99] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax involving binders. In *14th Annual Symposium on Logic in Computer Science, 2–5 July, 1999, Trento, Italy*, pages 214–224. IEEE Press, 1999. {43}

[GV92] J. F. Groote and F. W. Vaandrager. Structural operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992. {45}

[Has99] M. Hasegawa. *Models of sharing graphs: a categorical semantics of let and letrec.* BCS Distinguished Dissertation Series. Springer-Verlag, 1999. {25}

[Hon00] K. Honda. Elementary structures for process theory (1): sets with renaming. *Mathematical Structures in Computer Science*, 10(5):617–663, October 2000. {10}

[HT91] K. Honda and M. Tokoro. An object calculus for asynchronous communication. In *ECOOP '91: European Conference on Object-Oriented Programming,* *Geneva, Switzerland, July 15–19, 1991, Proceedings*, volume 512, pages 133–147. Springer-Verlag, 1991. {44}

[HY94a] K. Honda and N. Yoshida. Combinatory representation of mobile processes. In *POPL '94: 21st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Portland, Oregon, January 17–21, 1994*, pages 348–360. ACM Press, 1994. {42}

[HY94b] K. Honda and N. Yoshida. Replication in concurrent combinators. In *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19–22, 1994, Proceedings*, volume 789, pages 786–805. Springer-Verlag, 1994. {44}

[Joh99] P. Johnstone. A note on discrete Conduché fibrations. *Theory and Application of Categories*, 5(1):1–11, 1999. {17}

[JSV96] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):425–446, 1996. {23}

[Laf90] Y. Lafont. Interaction nets. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, January 1990*, pages 95–108. ACM Press, 1990. {25}

[Lei01a] J. J. Leifer. A category of action graphs and reflexive embeddings. Technical report, Computer Laboratory, University of Cambridge, 2001. To appear. {44}

[Lei01b] J. J. Leifer. *Operational congruences for reactive systems.* PhD thesis, Computer Laboratory, University of Cambridge, 2001. Available in revised form as Technical Report 521,Computer Laboratory, University of Cambridge, 2001, from http://pauillac.inria.fr/~leifer/. {2, 23, 38, 39, 40, 44}

[Lei02] J. J. Leifer. Synthesising labelled transitions and operational congruences in reactive systems, part 1. 2002. Submitted for publication. Available from http://pauillac.inria.fr/~leifer/. {2}

[LM02] J. J. Leifer and R. Milner. Shallow linear action graphs and their embeddings. *Formal Aspects of Computing*, 2002. To appear. Special issue in honour of Rod Burstall. Available from http://pauillac.inria.fr/~leifer/. {28, 44}

[Mil94] R. Milner. Action calculi V: reflexive molecular forms. Third draft; with an appendix by O. Jensen. Available from: ftp://ftp.cl.cam.ac.uk/users/rm135/ac5.ps.Z, 1994. {23}

[Mil96] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, 1996. {23, 25, 40}

[Mil01] R. Milner. Bigraphical reactive systems: basic theory. Technical Report 523, Computer Laboratory, University of Cambridge, 2001. Available from http://www.cl.cam.ac.uk/∼rm135/. {20, 41, 42, 43, 44}

[MSS00] P. Mateus, A. Sernadas, and C. Sernadas. Precategories for combining probabilistic automata. In M. Hofmann, G. Rosolini, and D. Pavlovic, editors, *Proc. CTCS '99*, volume 29 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2000. {9}

[NP96] U. Nestmann and B. C. Pierce. Decoding choice encodings. In *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26–29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996. {44}

[Sew] P. Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*. To appear. {30, 42}

[TFP99] D. Turi, M. P. Fiore, and G. D. Plotkin. Abstract syntax and variable binding. In *14th Annual Symposium on Logic in Computer Science, 2–5 July, 1999, Trento, Italy*, pages 193–202. IEEE Press, 1999. {43}

[TP97] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 – July 2, 1997*, pages 280–291. IEEE Press, 1997. {45}

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399