# Contexts and Embeddings for Closed Shallow Action Graphs

**Gian Luca Cattani, James J. Leifer and Robin Milner**

**University of Cambridge, Computer Laboratory**

**July 2000**

**Abstract:** Action calculi, which have a graphical presentation, were introduced to develop a theory shared among different calculi for interactive systems. The $\pi$-calculus, the $\lambda$-calculus, Petri nets, the Ambient calculus and others may all be represented as action calculi. This paper develops a part of the shared theory.

A recent paper by two of the authors was concerned with the notion of *reactive system*, essentially a category of process contexts whose behaviour is presented as a reduction relation. It was shown that one can, for any reactive system, uniformly derive a labelled transition system whose associated behavioural equivalence relations (e.g. trace equivalence or bisimilarity) will be congruential, under the condition that certain *relative pushouts* exist in the reactive system. In the present paper we treat *closed, shallow* action calculi (those with no free names and no nested actions) as a generic application of these results. We define a category of action graphs and embeddings, closely linked to a category of contexts which forms a reactive system. This connection is of independent interest; it also serves our present purpose, as it enables us to demonstrate that appropriate relative pushouts exist.

Complemented by work to be reported elsewhere, this demonstration yields labelled transition systems with behavioural congruences for a substantial class of action calculi. We regard this work as a step towards comparable results for the full class.

## Contents

| | | |
|---|---|---|
| **1** | **Introduction** | 3 |
| **2** | **Review** | 4 |
| **3** | **Examples and a problem** | 7 |
| **4** | **Solution strategy** | 12 |
| **5** | **Closed shallow action graphs** | 15 |
| **6** | **Embeddings** | 17 |
| **7** | **Contexts** | 24 |
| **8** | **Functors from embeddings to contexts** | 30 |
| **9** | **Functor inverses** | 35 |
| **10** | **Relative pushouts for contexts** | 37 |
| **11** | **Related and future work** | 44 |
| | **References** | 46 |
| | **Appendix** | 49 |

# 1 Introduction

Action calculi [20] were introduced to develop a theory shared among different calculi for interactive systems. The $\pi$-calculus, the $\lambda$-calculus, Petri nets, the Ambient calculus of Cardelli and Gordon [3] and others may all be represented as action calculi. But hitherto this shared framework has not yielded a substantial theoretical advance. New calculi continue to be developed independently, though they often have much in common.

One approach to the development is first to define a labelled transition system (LTS), then to base upon it some notion of behavioural pre-order or equivalence. Though in simple calculi a tractable behavioural congruence[1] may then be readily achieved, this is becoming harder with recent calculi. It has become more common, in fact, to start with a structural congruence and an unlabelled reduction relation, or *reaction relation* as we shall call it here, and then define a notion of barbed equivalence [22]. But again this does not readily yield behavioural congruence[2].

A recent paper by two of the authors [17] announced a method for deriving an LTS from a reaction relation which, under certain conditions, ensures that induced behavioural equivalences such as bisimilarity are congruential. This result is very general; but the work was motivated by action calculi, and it remains to show that the required conditions can be met for them. Here we confine our attention to closed, shallow action calculi: those with no free names and no nested actions. For these we show that, under mild assumptions on the reaction relation and the class of action contexts, the main required condition is met. The other condition is also met, as will be shown in Leifer's forthcoming PhD Dissertation [16]. We believe these results extend to all action calculi; we also believe that our assumptions will be often satisfied. This will establish action calculi as a setting in which to seek elements of a general theory of interaction.

This work is part of a long-term project loosely centred on action calculi and their graphical presentation. A geometric study of how interactive processes behave, including how they change their own configuration, is of course strongly motivated by the prominence of the internet and mobile computing. The study has at least two theoretical progenitors. One is the theory of mobile processes exemplified by the $\pi$-calculus [21], which moved away from a linear syntax for processes by introducing a structural congruence inspired by the Chemical Abstract Machine of Berry and Boudol [2]. The other is the theory of graph rewriting, going back to the categorical presentation by Ehrig [5] in terms of a double pushout construction; this line was later enriched by Corradini and Montanari [4] by explicit use of an algebraic theory, similar to that of Milner's flowgraphs [19]. We believe that the arrival of mobile computing raises the status of graphical models of interaction from a mere benefit to a necessity.

Two previous strands of work in the project are especially relevant. Closed action calculi were investigated by Gardner [7]; in particular she studied their expressive power in comparison with arbitrary action calculi. The second strand is Sewell's [23] derivation of LTSs for term rewriting, with parametric rewriting rules and extended by parallel composition, so as to obtain behavioural

[1]This term is intended broadly as an equivalence or pre-order preserved by all or many contexts, and defined in terms of a pattern of dynamic behaviour – e.g. traces, failures or bisimilarity.

[2]Recent work [6] has considerably improved the means by which behavioural congruence can be achieved starting from barbed equivalence, but it remains non-trivial.

congruence. He proposed the notion of *colouring* to keep track of component occurrences. His work has given us guidance on the uniform approach developed here, and on what congruences it should yield.

In Section 2 we review two topics, reactive systems and action calculi, in enough detail to make the paper self-contained. In Section 3 we give informally several examples which enable us to state the main problem more precisely, and motivate our solution strategy. In Section 4 we divide the problem into two parts. The solution to the first part will appear in Leifer's PhD Dissertation [16]. The present paper is concerned only with the second part, but in solving it we lay out a theory of action graphs which we believe will have wider application. Section 4 ends with an outline of the ensuing sections. In the concluding section, Section 11, we discuss related work and our proposals for further development of these ideas.

# 2 Review

In this section we review reactive systems and action calculi, as far as we need them.

## Reactive systems reviewed

The work of [17], in establishing conditions under which derived behavioural relations will be congruential, was done in the very general setting of a *reactive system*. For present purposes we need not define this concept rigorously; it is enough to describe informally a simple version of it[3]. A reactive system is a category whose objects $\alpha, \beta, \ldots$ are considered as interfaces, among which there is a distinguished "null" interface $\epsilon$, and whose arrows $C, D, \ldots$ are called *contexts*. Contexts with domain $\epsilon$ are called *actions*; we shall use lower case letters $a, b, \ldots$ for actions. A reactive system has one further ingredient: a set $\textsc{Reacts}$ of pairs of actions with like codomain. Each such pair $(l, r)$, called a *reaction rule*, means informally "$l$ can become $r$ in any context". We capture this formally by defining the reaction relation

$$\longrightarrow\!\!\!\triangleright \;\stackrel{\text{def}}{=}\; \{(C \circ l, C \circ r) \mid (l, r) \in \textsc{Reacts}\}\ .$$

We write $a \longrightarrow\!\!\!\triangleright\, a'$ for a member $(a, a')$ of this relation. Following the $\lambda$-calculus, we call $l$ the *redex* of the rule and $r$ its *contractum*. We may omit the composition symbol, e.g. writing $Cl$ for $C \circ l$.

Now, to obtain a useful behavioural equivalence for a reactive system, we wish to define a *labelled transition system* (LTS), i.e. a family $\{\xrightarrow{\lambda}\!\!\!\triangleright \mid \lambda \in \Lambda\}$ of relations indexed by a set $\Lambda$; in terms of this one can formulate familiar behavioural relations. We would like $\Lambda$ to be simple, and the resulting relations to be congruences. A first attempt at $\Lambda$ is the set of *all* contexts; we might define

$$a \xrightarrow{C}\!\!\!\triangleright\, a' \quad \stackrel{\text{def}}{\iff} \quad Ca \longrightarrow\!\!\!\triangleright\, a'\ . \tag{†}$$

[3] A more refined version may require that reaction rules be applied only within a subcategory of contexts, the so-called *reactive* contexts. For example, in CCS guarding contexts are not reactive.

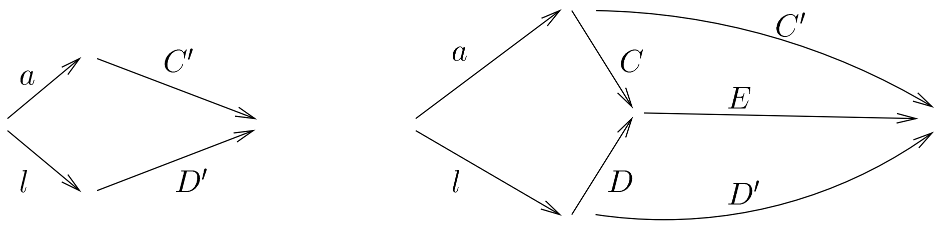


Figure 1: Commuting diagrams which may define labelled transitions

With this simple definition it turns out that the associated bisimulation equivalence, and others too, are congruences. But the family of all contexts is large and unwieldy. Can we make do with a subset of the contexts?

Intuitively, we would like to have a transition $a \xrightarrow{C} a'$ only when $C$ is "just large enough" so that $Ca = Dl$ for some reaction rule $(l, r)$ and context $D$. Then we would have $a' = Dr$ as the result of the transition. But what does "just large enough" mean? Suppose that $C'a = D'l$ , i.e. the left-hand diagram in Figure 1 commutes. Informally speaking, $C'$ may add more to $a$ than is needed to create the redex $l$, i.e. $C'$ may be larger than "just large enough". A "smaller" context $C$ may make the right-hand diagram of Figure 1 commute for some $D$ and $E$. Then $C$ is a more informative transition label than $C'$; the transition $a \xrightarrow{C} Dr$ tells us that $Ca \longrightarrow Dr$, hence also $C'a \longrightarrow D'r$ (because $C' = EC$ and $D' = ED$), which is what the transition $a \xrightarrow{C'} D'r$ would have told us.

**Example 1 (the $\pi$-calculus)** Consider the asynchronous $\pi$-calculus [12], whose terms under structural congruence form a reactive system with the generic reaction rule

$$\overline{x}\langle z\rangle \mid x(y).P \longrightarrow \{z/y\}P \ ,$$

i.e. the message $z$ received on channel $x$ replaces $y$ in $P$. Let $C' = [\ ] \mid \overline{x}\langle z\rangle \mid z(w).Q$. Then according to the definition (†) we would have the labelled transition $x(y).P \xrightarrow{C'} \{z/y\}P \mid z(w).Q$. But we do not want this clumsy $C'$ as a label in our LTS; it suffices to take the simpler context $C = [\ ] \mid \overline{x}\langle z\rangle$, giving the transition

$$x(y).P \xrightarrow{C} \{z/y\}P \ ,$$

which is a mere notational variant of the standard $\pi$-calculus transition $x(y).P \xrightarrow{x\langle z\rangle} \{z/y\}P$. In this case $D = [\ ]$ and $E = [\ ] \mid z(w).Q$. ■

In general, given the first diagram in Figure 1, what is a *minimal* triple $C, D, E$ making the second diagram commute? This minimality is made precise by Definition 46 in Section 10, which we reproduce here:

**Definition (relative pushout)** In any category **C**, let $f_i, h_i$ $(i = 0, 1)$ be arrows such that $h_0 f_0 = h_1 f_1$ (see Figure 2). A triple $g_0, g_1, g$ is called a *relative pushout (RPO) for* $f_0, f_1$ *wrt* $h_0, h_1$ if

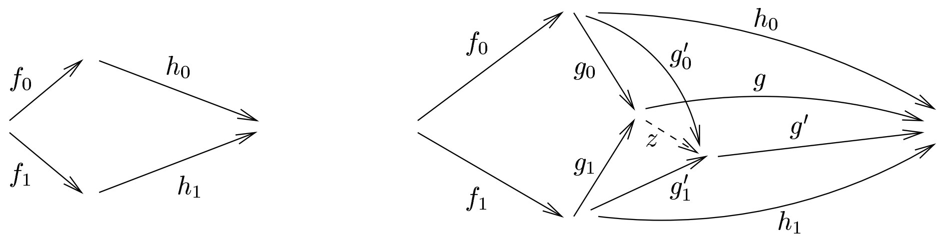



Figure 2: A relative pushout $g_0, g_1, g$

$g_0 f_0 = g_1 f_1$ and $g g_i = h_i$, and moreover for any other triple $g'_0, g'_1, g'$ such that $g'_0 f_0 = g'_1 f_1$ and $g' g'_i = h_i$, there is an unique mediating arrow $z$ such that $z g_i = g'_i$ and $g' z = g$. ∎

We shall often call a triple $g'_0, g'_1, g'$ such that $g'_0 f_0 = g'_1 f_1$ and $g' g'_i = h_i$ a *candidate*; the definition makes clear the sense in which the RPO is the best candidate. An RPO, when it exists, is unique up to isomorphism; we therefore sometimes refer to *the* RPO.

The relative pushout is a simple category-theoretic concept which may have a variety of applications (though we have not yet found the concept applied in the literature). RPOs have pleasant properties. When they exist, we may use the "just large enough" contexts $C$ which they provide as the labels of an LTS. We showed in [17] that they yield sensibly modest label-sets; moreover, we proved that the associated bisimilarity and other equivalences are congruences.

## Action calculi reviewed

We here review a restricted class of the action calculi which were presented in [20]. A *closed, shallow action calculus* is a strict monoidal category whose objects are integers $k, \ell, m, n, \ldots$, and whose arrows are actions written $a : (k, \ell)$, $b : (m, n)$ etc. (We avoid the usual arrow notation $a : k \to \ell$, reserving it for the context arrows of reactive systems.) The *tensor product* of these two actions is $a \otimes b : (k + m, \ell + n)$; the *composition* of $a : (k, \ell)$ and $b : (\ell, m)$ is $a \cdot b : (k, m)$; the *identity* action of arity $(m, m)$ is $\mathsf{i}_m$. The order of composition is not conventional in category theory; we adopt it because it connotes the orientation of the action graphs we shall work with. We call $(k, \ell)$ an *arity* and let $\alpha, \beta, \ldots$ range over arities. In this paper we deal only with closed, shallow action calculi; we usually omit these adjectives from now on.

A (closed, shallow) action calculus has a set of *controls* $K, L, \ldots$, each with 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 a *trace* operator in the sense of Joyal et al. [13], also known as *reflexion*, which we need not detail here.

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

For examples of actions, let $K : (0, 1)$, $M : (1, 1)$ and $L : (1, 0)$ be controls. Then the following


Figure 3: The action $K \cdot \mathsf{c} \cdot (M \otimes L)$ and the context $[_{1,1}] \cdot \mathsf{c} \cdot (M \otimes M)$

are actions, with their arities:

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

We let "$\cdot$" bind tighter than "$\otimes$", so the last can be written $K \cdot M \otimes M \cdot L$.

A *context* $C$ is an action containing a single hole with arity $\alpha$, written $[_\alpha]$. We omit the arity, writing $[\ ]$, if it is determined by the rest of the context or the surrounding discussion. Thus a context $C : \alpha \to \beta$ is a term of arity $\beta$ with a hole of arity $\alpha$. Here are two contexts with their source and target arities (the arity of the hole being fully determined in the second case):

$$\begin{array}{rcl} [_{1,1}] \cdot \mathsf{c} \cdot (M \otimes M) & : & (1,1) \to (1,2) \\ K \cdot [\ ] \cdot L & : & (1,1) \to (0,0) \,. \end{array}$$

Figure 3 shows an action 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 or a hole with arity $(m,n)$ has $m$ *source* ports on its left side and $n$ *target* ports on its right side. A control node of arity $(m,n)$ has $m$ target ports at its rectangular end and $n$ source ports at its blunted end. At a source node, branching of arcs represents $\mathsf{c}$ and absence of arcs represents $\omega$.

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 $C : \alpha \to \beta$ and $D : \beta \to \gamma$, written here $D \circ C$ (note the conventional order of composition), is formed by replacing the hole in $D$ by $C$. Composition is clearly associative, and there is an identity context $\mathsf{id}_\alpha = [_\alpha]$ for each arity. An action $a : \alpha$ can be considered as a context $a : (0,0) \to \alpha$ whose hole has minimum arity. We shall use lower case letters $a, \ldots$ for actions.

We have thus associated with an action calculus a reactive system $\mathbf{ACxt}_0$, whose objects are arities, with distinguished null arity $(0,0)$, and whose arrows are contexts, including actions. The subscript $0$ in $\mathbf{ACxt}_0$ refers to the closed shallow constraint.

# 3 Examples and a problem

In this section we give examples of specific RPOs in $\mathbf{ACxt}_0$, illustrating several phenomena. We end with an example showing cases in which RPOs fail to exist; this motivates the solution strategy of the ensuing section.

Remember that $\mathbf{ACxt}_0$ is really a family of reactive systems arising from action calculi; each is determined by a set of controls and a set of reaction rules.

**Example 2 (arithmetic)** We first illustrate how a relative pushout can determine a labelled transition, using an action calculus for elementary arithmetic having controls $\mathsf{0}$, $\mathsf{S}$ and $+$ with arities (0,1), (1,1) and (2,1). The reaction system is shown in Figure 4; it is an example of the sharing graphs of Hasegawa [10], which add sharing to the interaction nets of Lafont [15]. 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.

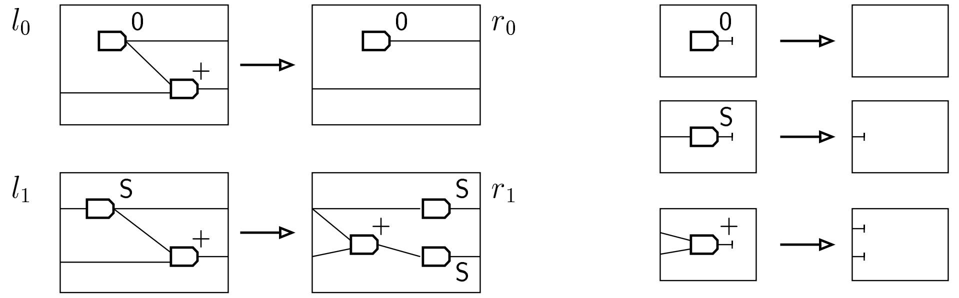
The middle diagram shows an action (graph) $a$ occurring in a larger one $b'$, which also contains an occurrence of the redex $l_1$ of the rule for $\mathsf{S}$. The contexts $C'$ and $D'$ correspond to the two occurrences, which overlap. Now what is the "just large enough" context $C$ which extends $a$ to contain $l_1$? It is not quite $C'$, because $C'$ has destroyed the possibility of sharing $\mathsf{S}$ 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 $\mathsf{S}$-node to be shared.) So our derived 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 3 (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 practical systems. The $\pi$-calculus depends heavily on it; witness the double occurrence of $x$ in its reaction rule shown in Example 1. Non-linearity can give rise to RPOs which are more complex than one might expect. Figure 5 shows two identical actions $a = b = K\cdot\mathsf{c}$, 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

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

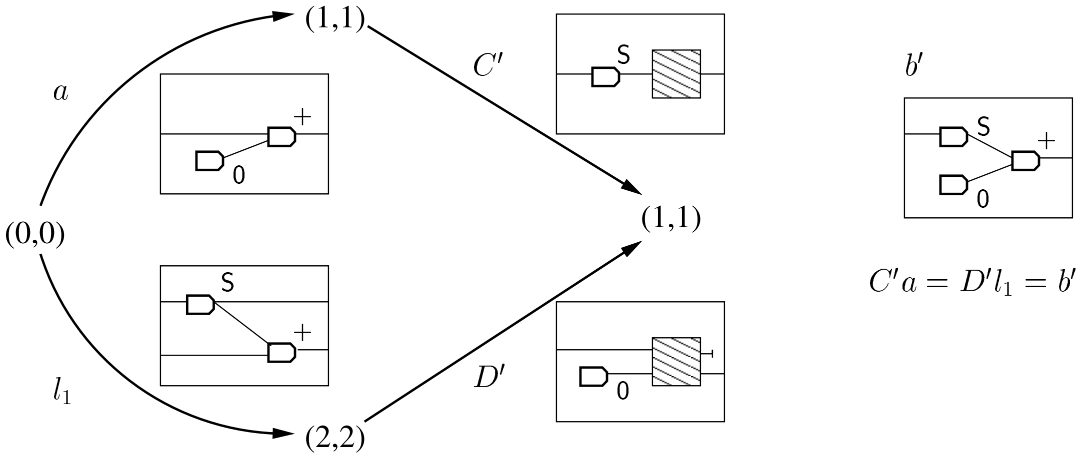
**Example 4 (reflexion)** A surprising phenomenon is how the presence of reflexion can affect the RPO. Let $K, N : (1,1)$, $L : (0,2)$ and $M : (2,0)$, and recall that $\mathsf{i}_1$ is the identity of arity $(1,1)$ for action composition. Figure 6 shows $a = L\cdot(\mathsf{i}_1 \otimes K)$ and $b = (\mathsf{i}_1 \otimes K)\cdot M$ 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 transition label in the derived LTS. ∎

These examples do not exhaust the phenomena which arise in finding RPOs in $\mathbf{ACxt}_0$, 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 $\mathbf{ACxt}_0$! Here is a counter-example.

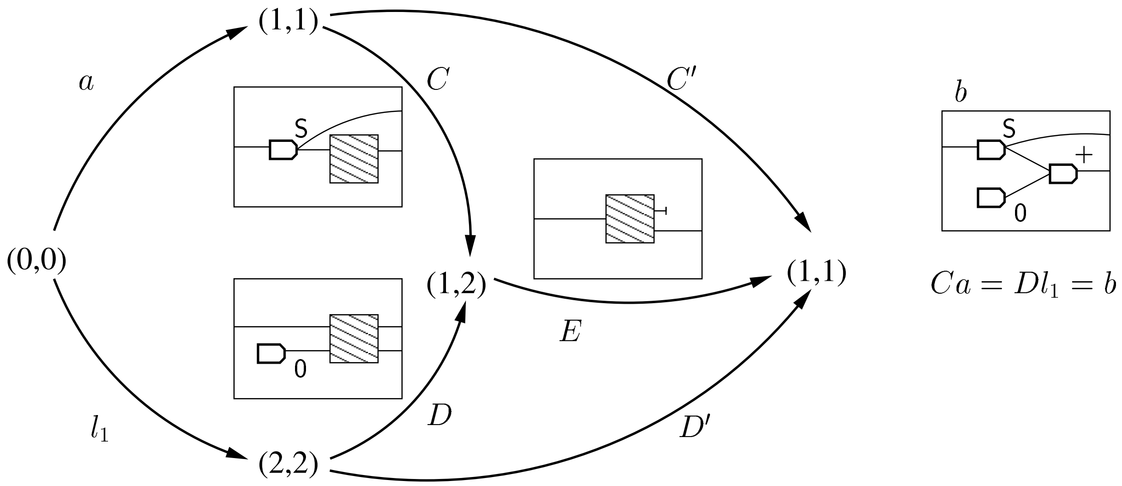



Arithmetic rules

Garbage collection rules




An action $a$ overlapping a redex $l_1$




A relative pushout for $a$ and $l_1$ wrt $C'$ and $D'$

Figure 4: A reactive system for arithmetic (Example 2)

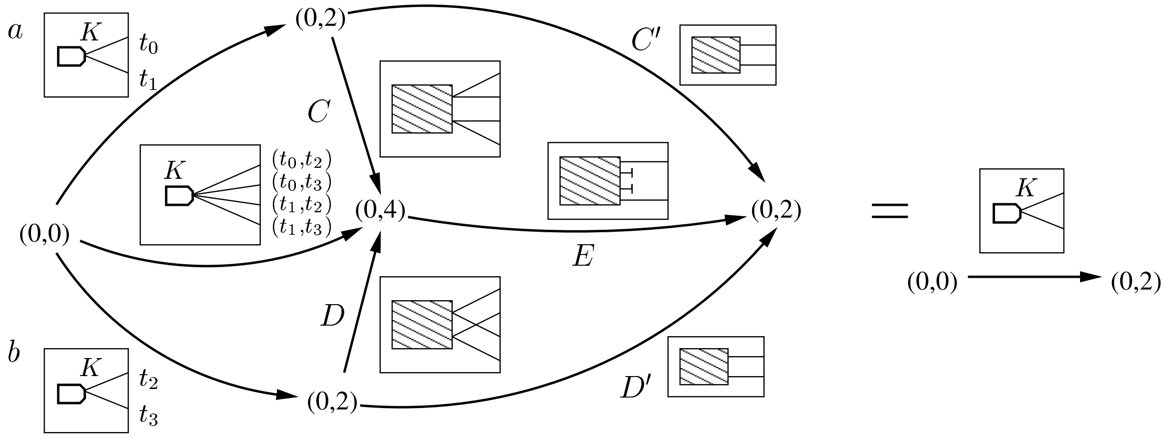



Figure 5: A relative pushout for copied wiring (Example 3)

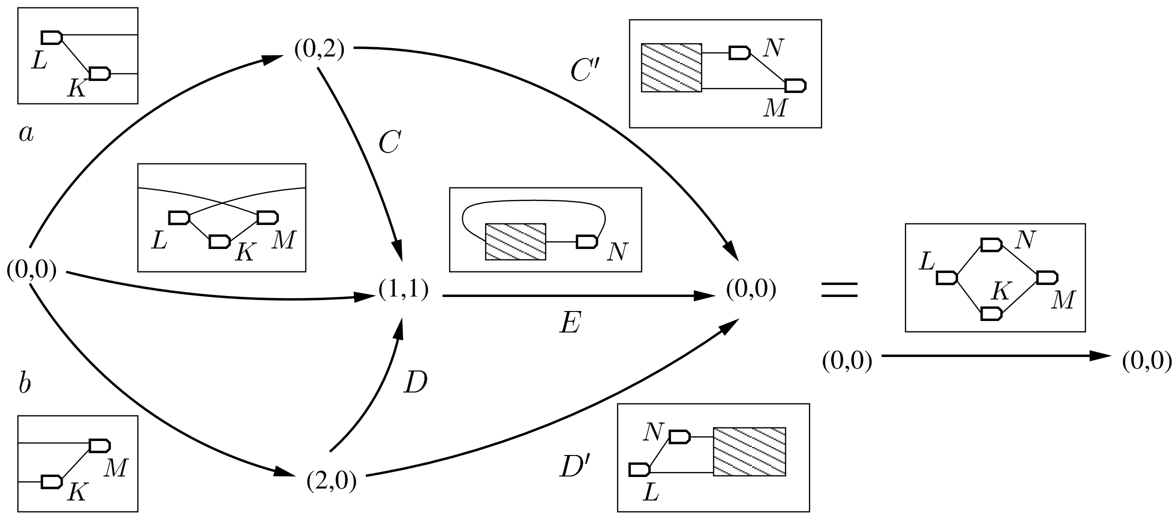
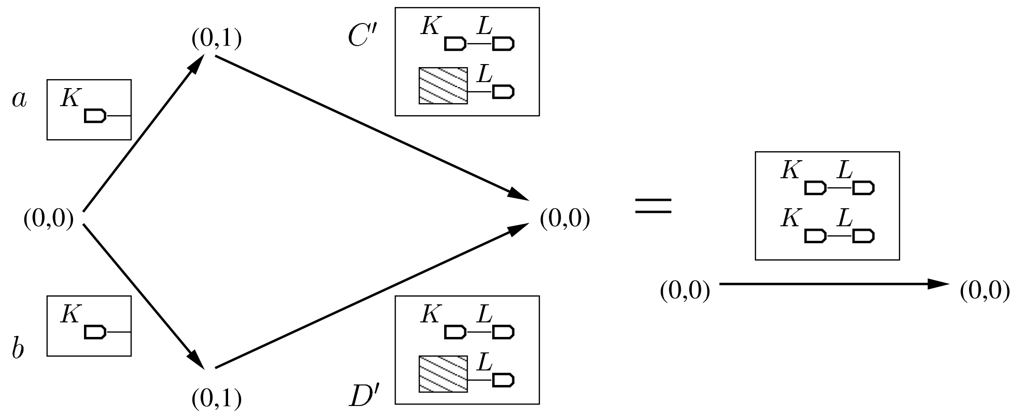



Figure 6: A relative pushout using reflexion (Example 4)




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

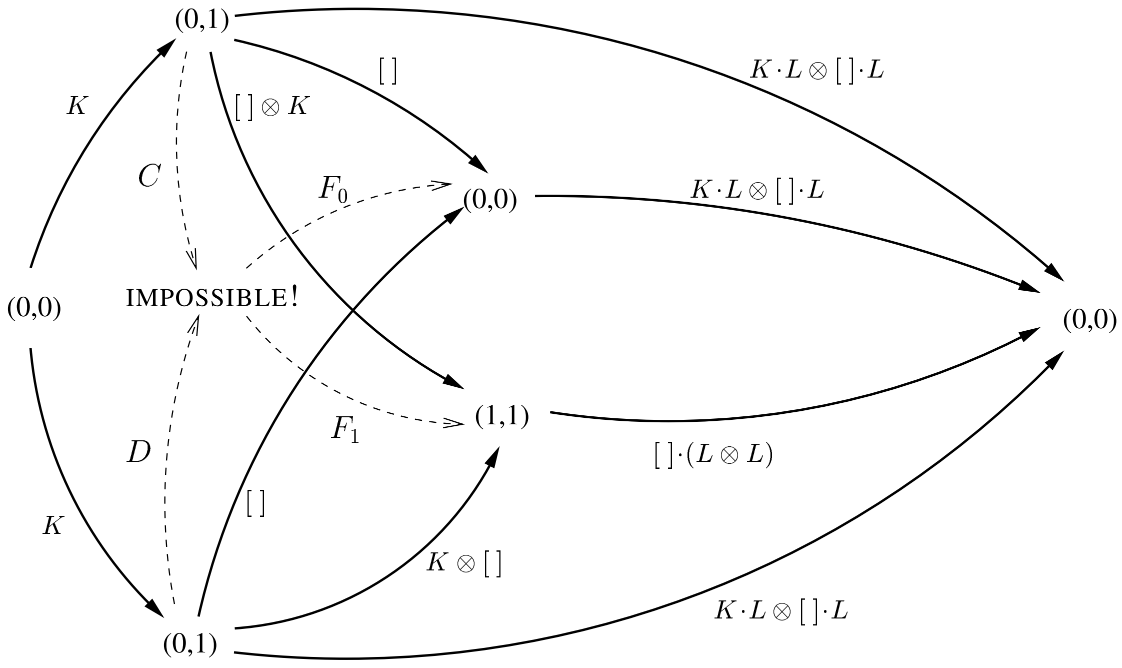



Figure 7: A missing relative pushout in $\mathbf{ACxt}_0$ (Example 5)

**Example 5 (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 $(0,1) \to (0,0)$. Then $C'a = D'b = K\cdot L \otimes K\cdot L$; this is shown in the upper diagram of Figure 7. The lower diagram shows two candidate RPOs, for which it is easy to verify commutation:

$$\begin{array}{rclll} C_0, D_0, E_0 & = & [\;], & [\;], & K\cdot L \otimes [\;]\cdot L \\ C_1, D_1, E_1 & = & [\;] \otimes K\,, & K \otimes [\;]\,, & [\;]\cdot(L \otimes L)\,. \end{array}$$

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 it also denies the existence of all RPOs for *linear* action calculi – those in which all source ports bear exactly one arc. Thus we cannot attribute the lack of RPOs to $\mathsf{c}$ and $\omega$, even though they demand careful treatment as shown in Examples 2 and 3.

The counter-example also illustrates *why* RPOs do not always exist in $\mathbf{ACxt}_0$. The equations $C_0K = D_0K = K$ and $C_1K = D_1K = K \otimes K$ hold respectively for the two candidates; but if we "trail" 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 reactive system which is a suitable refinement of $\mathbf{ACxt}_0$, and which possesses the RPOs we need. We expect its contexts to represent trailing; in fact, it will be a version of the colouring used by Sewell [23] to get appropriate LTSs in certain classes of reactive system.

# 4 Solution strategy

Our problem is how to use relative pushouts to derive, for an arbitrary action calculus, an LTS which yields behavioural congruence. Since we cannot find enough RPOs in the reactive system $\mathbf{ACxt}_0$, we declared in the previous section that we would seek a refined version of it based upon trailing. The first task of this paper is, therefore, to define this refined reactive system, which we shall call $\mathbf{PCxt}_0$. We then have two obligations. First, we have to show that from $\mathbf{PCxt}_0$ we can derive an LTS for $\mathbf{ACxt}_0$ whose associated behavioural equivalence is congruential. Second, which is the main topic of the paper, we have to prove that $\mathbf{PCxt}_0$ possesses enough RPOs. Here we explain in a little more detail how these two obligations will be tackled.

## Deriving a labelled transition system

We have already recalled from [17] the notion of RPO in an arbitrary category. A sister notion defined there is the following:

**Definition (idem pushout)** Let $g_0 f_0 = g_1 f_1$ in any category. Then $g_0, g_1$ is an *idem pushout (IPO) for* $f_0, f_1$ if $g_0, g_1, \mathsf{id}$ is an RPO for $f_0, f_1$. ■

In this case we may say that the equation $g_0 f_0 = g_1 f_1$, or the corresponding commuting square, *is* an IPO. A given pair $f_0, f_1$ may have many IPOs; intuitively, each one is a *a minimal* upper bound, in contrast with a pushout which is *the least* upper bound for $f_0, f_1$, up to isomorphism. In [17] IPOs were used to define an LTS as follows:

**Definition (labelled transition, first kind)** Let $a, a'$ be actions and $F$ a context in a reactive system **A**. Then there is a *labelled transition* $a \xrightarrow{F} a'$ iff there is a reaction rule $(l, r)$ and an IPO $Fa = Dl$ such that $a' = Dr$. ■

If RPOs (and hence IPOs) do not exist in **A**, can we nevertheless derive an LTS for **A** from another reactive system suitably related to **A**? A possible way is as follows:

**Definition (labelled transition, second kind)** Let **A** be a reactive system and $\mathcal{F} : \mathbf{P} \to \mathbf{A}$ a functor. Let $a, a'$ be actions and $F$ a context in **A**. Then there is a *labelled transition* $a \xrightarrow{F} a'$ iff there is a reaction rule $(l, r)$ and context $D$ in **A**, and an IPO $\hat{F}\hat{a} = \hat{D}\hat{l}$ in **P**, such that $\mathcal{F}(\hat{F}) = F$, $\mathcal{F}(\hat{a}) = a$, $\mathcal{F}(\hat{D}) = D$, $\mathcal{F}(\hat{l}) = l$ and $a' = Dr$. ■

This is not enough in general; we cannot expect this definition to yield congruential behaviour relations for an *arbitrary* functor $\mathcal{F}$, even if sufficient RPOs exist in **P**. But when **A** = **ACxt**$_0$, if we take **P** = **PCxt**$_0$ (as we shall define it) then a functor $\mathcal{F}$ exists which does indeed achieve this important property. The definition of $\mathcal{F}$, and a theorem to this effect, will be in Leifer's forthcoming PhD Dissertation [16].

Briefly, **PCxt**$_0$ arises by adding trailing to **ACxt**$_0$ as follows. In contrast with a morphism in **ACxt**$_0$

$$C : (m, n) \to (m', n')$$

whose domain and codomain are arities, a morphism in **PCxt**$_0$

$$\hat{C} : (m, n, V) \to (m', n', V')$$

has as domain and codomain *profiles* of the form $p = (m, n, V)$ where $V$ is a node set. The context $\hat{C}$ contains, in addition to the information contained in $C$, an injective map from $V$ to $V'$ identifying each node of an action (graph) $\hat{a}$ with a node in the action (graph) $\hat{C}\hat{a}$.

## Establishing relative pushouts

Our second obligation, and the focus of the present paper, is to demonstrate that enough RPOs exist in **PCxt**$_0$. We might try to work directly in **PCxt**$_0$, but it is much easier to work in an associated category **GEmb**$_0$, whose objects are action graphs and whose arrows are embeddings of one graph in another. Just as $\mathbf{ACxt}_0$ is a functorial image of $\mathbf{PCxt}_0$, so $\mathbf{PCxt}_0$ is a functorial image of $\mathbf{GEmb}_0$. The analogy goes further, for the objects (graphs) of $\mathbf{GEmb}_0$ arise by adding information to the objects (profiles) of $\mathbf{PCxt}_0$. If $p = (m, n, V)$ is a profile, then a graph $G = (m, n, V, ctr, src)$ with profile $p$ has also a function $ctr$ assigning a control to each node, and a function $src$ assigning a source port to each target port. Thus the overall picture has three levels:

| *Category* | *Objects* | | *Arrows* | |
|---|---|---|---|---|
| $\mathbf{GEmb}_0$ | graphs | $G = (m, n, V, ctr, src)$ | embeddings | $\phi : G \to G'$ |
| $\mathsf{Cod} \downarrow$ | | | | |
| $\mathbf{PCxt}_0$ | profiles | $p = (m, n, V)$ | trailed contexts | $\hat{C} : p \to p'$ |
| $\mathcal{F} \downarrow$ | | | | |
| $\mathbf{ACxt}_0$ | arities | $\alpha = (m, n)$ | contexts | $C : \alpha \to \alpha'$ |

The categories $\mathbf{GEmb}_0$ and $\mathbf{PCxt}_0$ are intimately related by a standard categorical notion; in fact $\mathbf{GEmb}_0$ is isomorphic to a *coslice* of $\mathbf{PCxt}_0$. We shall in due course recall the notion of a coslice $\mathbf{C}'$ of an arbitrary category $\mathbf{C}$; it provides a simple *codomain* functor $\mathsf{Cod}$ between $\mathbf{C}'$ and $\mathbf{C}$, and enables us to reduce the existence of RPOs in $\mathbf{C}$ to the existence of simpler constructions, namely *slice sums*, in $\mathbf{C}'$.

Thus, suppose that our first obligation is met by a suitable functor $\mathcal{F}$. Then, to achieve behavioural congruence in $\mathbf{ACxt}_0$, it is enough to establish the isomorphism mentioned above and then to construct slice sums in $\mathbf{GEmb}_0$. The remainder of the paper is devoted to this task.

To keep the problem tractable we have chosen for this exercise to limit the contexts we admit in $\mathbf{ACxt}_0$, and hence in $\mathbf{PCxt}_0$. Instead of admitting the full power of reflexion in contexts, we exclude contexts which impose a reflexion on their "hole" without a mediating control node. The details are in Section 7. We have no reason to believe the results will not extend to permit these reflexions. However, for some purposes the limitation is irrelevant; for example in the normal representation of the $\pi$-calculus as an action calculus, the actions all have arity of the form $(m, 0)$, for which reflexion is impossible. We take this matter up again in the concluding section.

**Outline** The remainder of the paper is organized as follows. In Section 5 we define action graphs, with illustrations and nomeclature. In Section 6 we define the category $\mathbf{GEmb}_0$ of action graph embeddings; we characterize its isomorphisms, especially node renamings, and the subcategory $\mathbf{GInc}_0$ of inclusion embeddings. In Section 7 we introduce contexts. First we define $\mathbf{PIns}_0$, the category of insertion contexts which correspond to inclusion embeddings. This leads to the definition of $\mathbf{PCxt}_0$, the category of trailed contexts.

In Section 8 we develop the correspondence between the two main categories. After introducing coslice categories, we first define a functor between $\mathbf{GInc}_0$ and the coslice category $\epsilon/\mathbf{PIns}_0$, where $\epsilon = (0, 0, \emptyset)$ is the empty profile. This leads directly to a functor $\mathcal{C} : \mathbf{GEmb}_0 \to \epsilon/\mathbf{PCxt}_0$ from embeddings to a coslice category of contexts. We conclude the section by introducing functions $\mathcal{E}_{G_0,G_1}$ taking contexts $C$ such that $CG_0 = G_1$ and yielding embeddings $\phi : G_0 \to G_1$. In Section 9 we employ the functions $\mathcal{E}_{G_0,G_1}$ to demonstrate that $\mathcal{C}$, which is easily seen to be isomorphic on objects, is in fact an isomorphism of categories.

Section 10 begins with the definition of RPOs and slice sums, and then observes that the RPOs we need in $\mathbf{PCxt}_0$ correspond to slice sums in $\epsilon/\mathbf{PCxt}_0$, and so in $\mathbf{GEmb}_0$ via the isomorphism $\mathcal{C}$. It is shown that these still do not exist in all cases in $\mathbf{GEmb}_0$. A key condition, SLICE, is then defined; it is argued that, in realistic action calculi, the SLICE condition will hold in those cases where slice sums are needed to establish a behavioural congruence. The main result follows: that SLICE is necessary and sufficient for the existence of a slice sum. Thus the goal of the present paper is achieved.

The proof of the last-mentioned result is given in outline here; the full proof will appear in Leifer's forthcoming PhD Dissertation [16].

# 5 Closed shallow action graphs

In this section, after introducing some notation, we define the class of action graphs that are the subject of this paper.

**Notation** We 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 the paper take forms such as $S = \sum_{v\in V} S_v + [m] + [n]$, 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 $j \in [n]$.

We write $X\#Y$ to mean that the sets $X$ and $Y$ are disjoint. In this case we often write their union as $X \uplus Y$, to remind us that they are disjoint. This conventional 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 $f : X \to Z$ and $g : Y \to Z$ are two functions with $X\#Y$, we may write $f \uplus g : X \uplus Y \to Z$ for their combination.

We write $f : X \rightarrowtail Y$, $f : X \twoheadrightarrow Y$ and $f : X \rightleftarrows Y$ for respectively injective, surjective and bijective functions, and $f : X \hookrightarrow Y$ for an injection which is an inclusion. We use "$\circ$" for function composition, $\mathsf{Id}_X$ for the identity function on the set $X$, and $\emptyset_X$ for the empty function from $\emptyset$ to $X$. We often omit all but the inner parentheses in a nested function application, writing for example $f_1 f_0(x)$ for $f_1(f_0(x))$. In a category we usually use juxtaposition for composition and $\mathsf{id}_A$ for the identity on the object $A$.

**Definition 1 (controls)** We fix a set $\mathcal{K}$ of *controls*, equipped with an *arity function* $ar : \mathcal{K} \to \mathbb{N}^2$. We use $K, L, \ldots$ to range over $\mathcal{K}$. For $ar(K) = (m, n)$ we write $K : (m, n)$; in this case we also define $arin(K) \stackrel{\text{def}}{=} m$, $arout(K) \stackrel{\text{def}}{=} n$. ■

**Definition 2 (action graph)** A *(closed, shallow) action graph* $G = (m, n, V, ctr, src)$ comprises an arity $(m, n)$, a set $V$ of *nodes*, a *control* map $ctr : V \to \mathcal{K}$ assigning a control in $\mathcal{K}$ to each node in $V$, and a *source* map $src : T \to S$ assigning a *source (port)* in $S$ to each *target (port)* in $T$, where

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

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

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

We shall write $bind(v, i)$ and $arg(v, j)$ for the ports $bind(v)(i)$ and $arg(v)(j)$. 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 $T^{arg(v)}$ the set of argument targets of $v$. We shall also write for example $S^{bind}$ for $\biguplus_{v \in V} S^{bind(v)}$. With this notation we can represent our partitions as

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

An example of an action graph with arity $(1, 3)$ is shown in Figure 8, 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 not prohibited. 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 control terms do not contain actions. This paper emphasizes the graphical rather than the algebraic aspects of action calculi. We study the closed, shallow graphs as a preliminary to the full graphical presentation of action calculi, since notions such as graph embedding and graph context are more clearly grasped in this simpler setting. From now on we shall usually omit the adjectives "closed" and "shallow", understanding them to qualify all graphs throughout the paper.

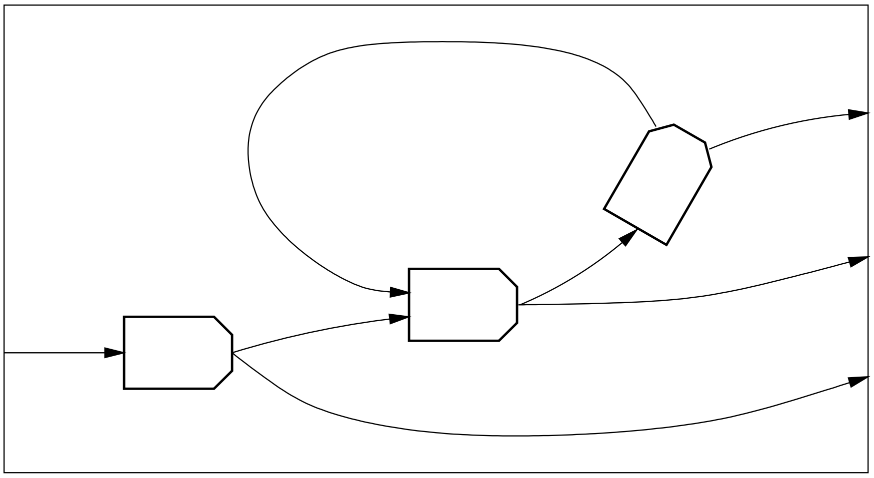


Figure 8: A closed shallow action graph

**Convention** Our graphs will almost always be denoted by $G$ suitably subscripted. We shall not always explicitly list all their primary components $V$, $ctr$, 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, subscripted as necessary to connote the graph's name.

# 6 Embeddings

We now define the category $\mathbf{GEmb}_0$ in which the objects are graphs and the arrows are embeddings. Our main task is to define what we mean by graph embedding and to capture the meaning in a tractable list of axioms. Our approach differs from the standard approach to graph embedding due to Ehrig [5]; this point is taken up in the concluding section.

**Intuition** Two graphs are shown in Figure 9. $G'$ is the graph of Figure 8, and $G$ is a smaller graph which intuitively "occurs" in $G'$. The way it occurs is represented informally by the dotted rectangle. The node $v$ of $G$ is represented by $v'$ in $G'$; the input sources $s_0$ and $s_1$ are represented respectively by $s'_0$ and $s'_1$, and the binding source $s_2$ by $s'_2$. The argument targets $t_0$ and $t_1$ are represented respectively by $t'_0$ and $t'_1$, and the output target $t_2$ by both $t'_2$ and $t''_2$. The other output target of $G$ is not represented in $G'$; the aborted arc (not formally part of $G'$) indicates that the lower arc from $s_2$ in $G$ is discarded by the embedding into $G'$.

Of course several parts of $G'$ don't represent $G$. Also our definition of embedding must allow that a target of $G$ may have any number (including 0) of representatives in $G'$; but we shall insist that every source in $G$ is represented exactly once in $G'$. Two sources may have the same representative, though this is not the case in our example.

We are therefore led to the following definition:

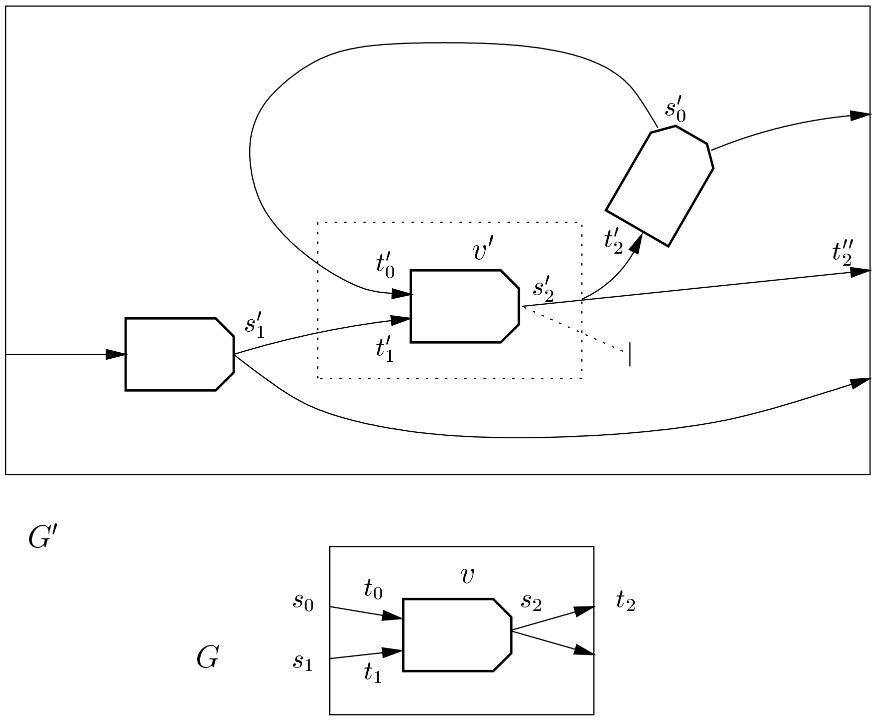



Figure 9: An embedding

---

**Definition 3 (embedding)** For $G_i = (V_i, ctr_i, src_i) \in \mathbf{GEmb}_0$, $i = 0, 1$, a *(loose) embedding* $\phi : G_0 \to G_1$ *of* $G_0$ *in* $G_1$ consists of a triple of functions $(\phi^{\mathsf{V}}, \phi^{\mathsf{S}}, \phi^{\mathsf{T}})$ where

$$
\begin{array}{rl}
\phi^{\mathsf{V}} : V_0 \rightarrowtail V_1 & \text{is an injection of nodes;} \\
\phi^{\mathsf{S}} : S_0 \to S_1 & \text{is a map of sources;} \\
\phi^{\mathsf{T}} : T_1 \rightharpoonup T_0 & \text{is a partial map of targets}
\end{array}
$$

satisfying the following axioms:

$$
\begin{array}{rl}
\text{E-CTR}: & ctr_1 \phi^{\mathsf{V}}(v) = ctr_0(v) \\
\text{E-SRC}: & \phi^{\mathsf{S}} \circ src_0 \circ \phi^{\mathsf{T}} \subseteq src_1 \\
\text{E-BIND}: & {\phi^{\mathsf{S}}}^{-1}\{bind_1(\phi^{\mathsf{V}}(v), i)\} = \{bind_0(v, i)\} \\
\text{E-ARG}: & {\phi^{\mathsf{T}}}^{-1}\{arg_0(v, j)\} = \{arg_1(\phi^{\mathsf{V}}(v), j)\} \\
\text{E-TARGS}: & src_1^{-1} \phi^{\mathsf{S}}(S_0^{bind}) \subseteq \mathsf{Dom}(\phi^{\mathsf{T}})\ .
\end{array}
$$

■

These axioms deserve a little explanation. E-CTR says that $\phi$ respects the control function; E-SRC says that it also respects the source function where $\phi^{\mathsf{T}}$ is defined. E-BIND says not only that $\phi$ respects the binding source functions, but also that $\phi^{\mathsf{S}}$ never identifies (in $G_1$) a binding source of $G_0$ with any other source. This enforces the looseness constraint (see below). E-ARG is similar. E-TARGS says that if a source $s$ in $G_1$ represents a bound source of $G_0$, then all its targets represent targets of $G_0$.

The looseness constraint prevents an embedding where $G_1$ creates a tight loop (such as is formed by the reflexion operator in action calculi) from the back of the $G_0$ to the front. It does not preclude such a loop proceeding via a control, which indeed occurs in Figure 9; thus our embeddings will permit some but not all of the power of reflexion. To match full reflexion is more difficult and is not attempted here. This limitation on embeddings corresponds to the limitation on contexts mentioned in Section 4. We shall usually omit the adjective "loose" in this paper, as it applies to embeddings throughout.

**Proposition 4** *An equivalent set of axioms is gained by replacing* E-TARGS *by the following:*

E-TARGS$'$: If $s_0 \in S_0^{bind}$ and $\phi^{\mathsf{S}}(s_0) = src_1(t_1)$ then $\phi^{\mathsf{T}}(t_1)$ is defined and $s_0 = src_0\phi^{\mathsf{T}}(t_1)$.

**Proof** It is clear that E-TARGS$'$ implies E-TARGS; it remains to show that the given axioms imply E-TARGS$'$. Assume its antecedent. Since $s_0 \in S_0^{bind}$, by E-TARGS $t_0 \stackrel{\text{def}}{=} \phi^{\mathsf{T}}(t_1)$ is defined and therefore by E-SRC

$$\phi^{\mathsf{S}} src_0(t_0) = src_1(t_1) \ ;$$

also $s_0 = bind_0(v_0, i)$ for some $v_0$, whence from E-BIND we have

$$src_1(t_1) = \phi^{\mathsf{S}}(s_0) = bind_1(\phi^{\mathsf{V}}(v_0), i) \ .$$

But E-BIND asserts that $bind_1(\phi^{\mathsf{V}}(v_0), i)$ has an unique preimage under $\phi^{\mathsf{S}}$, hence $s_0 = src_0(t_0)$ as required. ■

**Definition 5 (the category of graphs and embeddings)** The objects of the category $\mathbf{GEmb}_0$ of (closed, shallow) graphs and (loose) embeddings are the graphs defined in Definition 2. The arrows are the embeddings defined in Definition 3. If $\phi : G \to G'$ and $\psi : G' \to G''$ are two embeddings, then their *composition* $\psi\phi : G \to G''$ is defined by

$$\psi\phi \stackrel{\text{def}}{=} (\psi^{\mathsf{V}} \circ \phi^{\mathsf{V}}, \psi^{\mathsf{S}} \circ \phi^{\mathsf{S}}, \phi^{\mathsf{T}} \circ \psi^{\mathsf{T}}) \ .$$

The *identity* embedding $\mathsf{id}_G : G \to G$ is defined by

$$\mathsf{id}_G \stackrel{\text{def}}{=} (\mathsf{Id}_V, \mathsf{Id}_S, \mathsf{Id}_T) \ .$$ ■

This definition is justified by the following two propositions.

**Proposition 6** *If $\phi_i : G_i \to G_{i+1}$ are two embeddings ($i = 0, 1$), then $\phi_1\phi_0$ is an embedding.*

**Proof** It is clear that $\phi_1^{\mathsf{V}} \circ \phi_0^{\mathsf{V}}$ is an injection, since its components are injections. It remains to show that $\phi_1\phi_0$ satisfies the five axioms. The first four are stated in a form which makes the proof easy. Take E-BIND for example; we have

$$\begin{aligned} (\phi_1^{\mathsf{S}} \circ \phi_0^{\mathsf{S}})^{-1}\{bind_2((\phi_1^{\mathsf{V}} \circ \phi_0^{\mathsf{V}})(v), i)\} &= \phi_0^{\mathsf{S}^{-1}} \phi_1^{\mathsf{S}^{-1}} \{bind_2(\phi_1^{\mathsf{V}}\phi_0^{\mathsf{V}}(v), i)\} \\ &= \phi_0^{\mathsf{S}^{-1}} \{bind_1(\phi_0^{\mathsf{V}}(v), i)\} \\ &= \{bind_0(v, i)\} \ . \end{aligned}$$

We now consider E-TARGS$'$, which is easier to verify than E-TARGS. Assume the antecedent for $\phi_1\phi_0$, namely that

$$s_0 \in S_0^{bind} \text{ and } (\phi_1^{\mathsf{S}} \circ \phi_0^{\mathsf{S}})(s_0) = src_2(t_2) \; ;$$

we must prove that $(\phi_0^{\mathsf{T}} \circ \phi_1^{\mathsf{T}})(t_2)$ is defined and $s_0 = src_0(\phi_0^{\mathsf{T}} \circ \phi_1^{\mathsf{T}})(t_2)$.

From the assumptions and E-BIND, we have $s_1 \stackrel{\text{def}}{=} \phi_0^{\mathsf{S}}(s_0) \in S_1^{bind}$ and $\phi_1^{\mathsf{S}}(s_1) = src_2(t_2)$. By E-TARGS$'$ for $\phi_1$ it follows that

$$t_1 \stackrel{\text{def}}{=} \phi_1^{\mathsf{T}}(t_2) \text{ is defined and } s_1 = src_1(t_1) \; .$$

So we now have that $s_0 \in S_0^{bind}$ and $\phi_0^{\mathsf{S}}(s_0) = src_1(t_1)$, which by E-TARGS$'$ for $\phi_0$ implies that

$$t_0 \stackrel{\text{def}}{=} \phi_0^{\mathsf{T}}(t_1) \text{ is defined and } s_0 = src_0(t_0) \; .$$

These two properties yield the required result. ■

**Proposition 7** *For all embeddings $\phi : G \to G'$, $\phi = \phi\,\mathsf{id}_G = \mathsf{id}_{G'}\,\phi$. For all embeddings $\phi_i : G_i \to G_{i+1}$ ($i = 0, 1, 2$), $(\phi_2\phi_1)\phi_0 = \phi_2(\phi_1\phi_0)$. The empty graph* **0** *is an initial object in* $\mathbf{GEmb}_0$.

**Proof** Obvious from the form of the definitions. ■

For any graph $G$ we shall write $\mathbf{0}_G$ for the unique embedding $\mathbf{0} \to G$ whose components are all empty (partial) functions.

We now consider how to characterize the *isomorphisms* in $\mathbf{GEmb}_0$, i.e. those embeddings $\phi : G_0 \to G_1$ for which there exists $\phi^{-1} : G_1 \to G_0$ with $\phi^{-1}\phi = \mathsf{id}_{G_0}$ and $\phi\phi^{-1} = \mathsf{id}_{G_1}$.

**Proposition 8** *Let $G_i = (V_i, ctr_i, src_i) : (m_i, n_i)$ be two action graphs, $i = 0, 1$, and let $\iota^{\mathsf{V}} : V_0 \rightarrowtail V_1$, $\iota^{\mathsf{S}} : S_0 \to S_1$ and $\iota^{\mathsf{T}} : T_1 \rightharpoonup T_0$ be three maps. Then $\iota = (\iota^{\mathsf{V}}, \iota^{\mathsf{S}}, \iota^{\mathsf{T}}) : G_0 \to G_1$ is an isomorphism in $\mathbf{GEmb}_0$ iff its components are bijections satisfying*

$$\begin{array}{rl}
\text{I-CTR}: & ctr_1 \circ \iota^{\mathsf{V}} = ctr_0 \\
\text{I-SRC}: & \iota^{\mathsf{S}} \circ src_0 \circ \iota^{\mathsf{T}} = src_1 \\
\text{I-BIND}: & \iota^{\mathsf{S}} \circ bind_0(v) = bind_1 \iota^{\mathsf{V}}(v) \\
\text{I-ARG}: & \iota^{\mathsf{T}} \circ arg_1 \iota^{\mathsf{V}}(v) = arg_0(v) \; .
\end{array}$$

**Proof** ($\Longrightarrow$) First, the components have to be bijections in order that an inverse

$$\iota^{-1} = (\iota^{\mathsf{V}^{-1}}, \iota^{\mathsf{S}^{-1}}, \iota^{\mathsf{T}^{-1}})$$

may exist. Second, the four conditions follow immediately from the first four embedding axioms.

($\Longleftarrow$) Since the components are bijections, the embedding axioms are deducible from the four given conditions; in particular, E-TARGS follows since $\iota^{\mathsf{T}}$ is total. Hence $\iota$ is an embedding. Now define $\iota^{-1}$ as above. It is clear that both $\iota\iota^{-1}$ and $\iota^{-1}\iota$ are identity embeddings, by the definition of composition; it only remains to show that $\iota^{-1}$ is indeed an embedding. But we know its components are bijections, and hence from I-CTR, I-SRC, I-BIND and I-ARG we can readily deduce the same four properties for $\iota^{-1}$ with subscripts $0, 1$ interchanged. Hence, as for $\iota$, $\iota^{-1}$ is an embedding and the proof is complete. ■

**Corollary 9** *If $\iota : G_0 \to G_1$ is an isomorphism with $G_i : (m_i, n_i)$, then $m_0 = m_1 (= m$ say) and $n_0 = n_1 (= n$ say), and there exist unique permutations $\pi^{in} : [m] \to [m]$ and $\pi^{out} : [n] \to [n]$ such that*

$$in_1 \circ \pi^{in} = \iota^{\mathsf{S}} \circ in_0 \;\; \textit{and} \;\; \iota^{\mathsf{T}} \circ out_1 = out_0 \circ \pi^{out^{-1}} .$$

**Proof** With the notation of the proposition, we deduce from I-BIND that $\iota^{\mathsf{S}}(S_0^{bind}) = S_1^{bind}$; as $\iota^{\mathsf{S}}$ is a bijection it follows that $\iota^{\mathsf{S}}(S_0^{in}) = S_1^{in}$, whence immediately $m_0 = m_1$. Then we may define $\pi^{in}(i) \stackrel{\text{def}}{=} in_1^{-1}\iota^{\mathsf{S}} in_0(i)$, thus uniquely achieving the required condition. A similar argument applies for output targets. ■

**Definition 10 (profiles, permutations and node renamings)** We call a triple $(m, n, V)$ (an arity and a node set) a *profile*. We call a pair $\pi = (\pi^{in}, \pi^{out})$ of bijections $\pi^{in} : [m] \rightleftarrows [m]$ and $\pi^{out} : [n] \rightleftarrows [n]$ an *(arity) permutation (on $m, n$)*, and an injection $\rho : U \rightarrowtail U'$, where $V \subseteq U$, a *node renaming (on $V$)*. ■

We have shown, then, that each isomorphism of $\mathbf{GEmb}_0$ induces a permutation and a node renaming. We now wish to show that the converse is true, in a strong sense:

**Proposition 11** *Let $G : (m, n)$ have node set $V$. Let $\pi$ be a permutation on $m, n$ and $\rho$ a node renaming on $V$. Then there exists a unique pair $G', \iota$ such that $\iota : G \to G'$ is an isomorphism in* $\mathbf{GEmb}_0$ *satisfying*

$$\begin{aligned} \iota^{\mathsf{V}} &= \rho \restriction V : V \rightleftarrows V' \\ \iota^{\mathsf{S}} \circ in &= in' \circ \pi^{in} \\ \iota^{\mathsf{T}} \circ out' &= out \circ \pi^{out^{-1}} \end{aligned}$$

*where $in'$, $out'$ are the input source and output target injections associated with $G'$.*

**Proof** (outline) Let $G' \stackrel{\text{def}}{=} (V', ctr', src') : (m, n)$ and $\iota : G \to G'$ be defined as follows:

$$\begin{aligned} V' &\stackrel{\text{def}}{=} \rho V \\ ctr' &\stackrel{\text{def}}{=} ctr \circ \iota^{\mathsf{V}^{-1}} \\ src' &\stackrel{\text{def}}{=} \iota^{\mathsf{S}} \circ src \circ \iota^{\mathsf{T}} \\ \iota^{\mathsf{V}} &\stackrel{\text{def}}{=} \rho \restriction V \\ \iota^{\mathsf{S}}(s) &\stackrel{\text{def}}{=} \begin{cases} in'(\pi^{in}(i)) & \text{if } s = in(i) \\ bind'(\rho(v), k) & \text{if } s = bind(v, k) \end{cases} \\ \iota^{\mathsf{T}}(t) &\stackrel{\text{def}}{=} \begin{cases} out(\pi^{out^{-1}}(i)) & \text{if } t = out'(i) \\ arg(v, k) & \text{if } t = arg'(\rho(v), k) \end{cases} \end{aligned}$$

The components of $\iota$ are bijections satisfying the conditions in Proposition 8, hence $\iota$ is an isomorphism; they also satisfy the present conditions required of $\iota$. It can be checked that any pair $G', \iota$ satisfying all these conditions will also satisfy the defining equations above. ■

From now on we pay explicit attention to those isomorphisms which do not permute the arity.

**Definition 12 (node isomorphisms and actions)** In the case that $\pi = (\mathsf{Id}_{[m]}, \mathsf{Id}_{[n]})$ is the identity permutation on $m, n$, we call the isomorphism $\iota$ constructed in Proposition 11 a *node isomorphism*, and denote it just by $\rho_G$. Furthermore, we denote the graph $G'$ for which $\rho_G : G \to G'$ by

$$\rho \centerdot G ,$$

and we call the whole isomorphism, or sometimes $\rho \centerdot G$ alone, the *action of $\rho$ upon $G$*. When $G$ is understood, we may write just $\rho$ for the isomorphism $\rho_G$. ■

For future reference let us just record the components of this action.

**Proposition 13** *Let $G = (V, ctr, src) : (m, n)$ be a graph, and $\rho$ be a node renaming on $V$. Then the components of $\rho \centerdot G = (V', ctr', src')$ are related to those of $G$ by the source and target components $\rho^{\mathsf{S}} : S \to S'$ and $\rho^{\mathsf{T}} : T' \to T$ of the isomorphism $\rho : G \to \rho \centerdot G'$ as follows:*

$$\begin{array}{rclrclrcl} V' & = & \rho V & bind'\rho(v) & = & \rho^{\mathsf{S}} \circ bind(v) & in' & = & \rho^{\mathsf{S}} \circ in \\ ctr' \circ \rho & = & ctr & \rho^{\mathsf{T}} \circ arg'\rho(v) & = & arg(v) & \rho^{\mathsf{T}} \circ out' & = & out \; . \\ src' & = & \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}} & & & & & & \end{array}$$

**Proof** Already implicit in Proposition 11. ■

We now consider those embeddings $\phi : G \to G'$ which preserve the names of the nodes of $G$; these will play an important role in the ensuing theory.

**Definition 14 (inclusions)** An embedding $\phi : G \to G'$ is called an *inclusion (embedding)* if $\phi^{\mathsf{V}} : V \hookrightarrow V'$ is an inclusion. ■

It was pointed out after Definition 3 that for any embedding $\phi$, the restriction $\phi^{\mathsf{S}} \restriction S^{bind}$ is an injection. This becomes an inclusion, when $\phi$ is an inclusion embedding. A similar remark applies to the argument targets.

**Proposition 15** *Every embedding $\phi$ can be expressed uniquely as $\phi = \eta\rho$, with $\eta$ an inclusion and $\rho$ a node isomorphism.*

**Proof** Let $\phi : G_0 \to G_1$. The node renaming $\rho = \phi^{\mathsf{V}} : V_0 \rightarrowtail V_1$ yields by Proposition 13 the node isomorphism $\rho : G_0 \to \rho \centerdot G_0$, with $\rho^{\mathsf{V}} = \phi^{\mathsf{V}} : V_0 \rightleftarrows \rho V_0$. Define $\eta : \rho \centerdot G_0 \to G_1 \stackrel{\text{def}}{=} \phi\rho^{-1}$.

$$\begin{tikzcd} G_0 \arrow[r, "\eta"] \arrow[d, "\rho_{G_0}"'] & G_1 \arrow[d, "\rho_{G_1}"] \\ \rho\centerdot G_0 \arrow[r, "\rho\centerdot\eta"] & \rho\centerdot G_1 \end{tikzcd}$$

Figure 10: Action of a renaming upon an inclusion embedding

---

Clearly $\phi = \eta\rho$, and $\eta$ is seen to be an inclusion thus: each node of $\rho V_0$ has the form $v' = \rho^{\mathsf{V}}(v)$, for some $v \in V_0$; so $\eta^{\mathsf{V}}(v') = (\phi\rho^{-1})^{\mathsf{V}}(\rho^{\mathsf{V}}(v)) = (\phi\rho^{-1}\rho)^{\mathsf{V}}(v) = \phi^{\mathsf{V}}(v) = v'$. Moreover, it is straightforward to show that the requirement for $\eta$ to be an inclusion determines $\rho$ uniquely on $V_0$, hence determines $\rho$ uniquely. ∎

It is clear that the identities are inclusions, and the composition of embeddings preserves the inclusion property. This justifies the following definition of a category we shall need later.

**Definition 16 (the category of graphs and inclusions)** We define $\mathbf{GInc}_0$ to be the subcategory of $\mathbf{GEmb}_0$ having all the objects of $\mathbf{GEmb}_0$, and as arrows just those embeddings which are inclusions. ∎

The category $\mathbf{GInc}_0$ contains no isomorphisms except those whose node component is the identity. As we have seen, node renamings $\rho$ provide the opportunity to factor any embedding $\phi$ as $\phi = \eta\rho$. But they have another role in relation to $\mathbf{GInc}_0$; we shall now define the quasi-functorial action of renamings upon $\mathbf{GInc}_0$.

**Definition 17 (node renamings on inclusions)** Let $\eta : G_0 \to G_1$ be an inclusion embedding and $\rho$ a node renaming on $V_1$. Then the *action* of $\rho$ upon $\eta$ is

$$\rho\centerdot\eta : \rho\centerdot G_0 \to \rho\centerdot G_1 \ ,$$

defined by $\rho\centerdot\eta \stackrel{\text{def}}{=} \rho_{G_1}\eta\rho_{G_0}^{-1}$. ∎

**Proposition 18** *The embedding $\rho\centerdot\eta$ is an inclusion. Moreover it preserves identities and composition, i.e. $\rho\centerdot\mathsf{id} = \mathsf{id}$ and $\rho\centerdot(\eta_1\eta_0) = (\rho\centerdot\eta_1)(\rho\centerdot\eta_0)$ whenever the action is defined. Hence $\rho\centerdot(-)$ is a functor upon the full subcategory of $\mathbf{GInc}_0$ whose objects (graphs) have a node set included in $\mathsf{Dom}(\rho)$. Also the diagram in Figure 10 commutes, and thus the unique decomposition of an embedding of the form $\rho_{G_1}\eta$ is given by $\rho_{G_1}\eta = (\rho\centerdot\eta)\rho_{G_0}$.*

**Proof** Straightforward. ∎

# 7 Contexts

We shall now proceed to construct the category $\mathbf{PCxt}_0$ of closed shallow loose contexts, which will have a close correspondence with the embeddings category $\mathbf{GEmb}_0$. The intuition of "context" is well supplied by Figure 9; 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*. (We do not here consider contexts with more than one hole.)

The upper diagram in Figure 11 shows an important variation of the embedding shown in Figure 9. The larger graph $G'$ is unchanged, but $G$ now contains an arc which is pure "wiring", not attached to any control node. Action graphs admit such possibilities, so our notion of context must allow for them. (In the extreme case either a graph or a context may have no nodes but consist only of wiring.) Note in particular that the sources $s_1$ and $s_3$ in $G$ are both represented by $s'_1$ in $G'$, and that $t'_3$ in $G'$ (which represented no target of the old $G$) represents the target $t_3$ of the new $G$.

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 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.

There are two stages in the contruction of $\mathbf{PCxt}_0$. The first stage is to define the category $\mathbf{PIns}_0$ of shallow closed "insertion contexts". They will correspond to inclusion embeddings; in $\mathbf{PIns}_0$ we shall allow composition of insertions only when they have disjoint node sets.

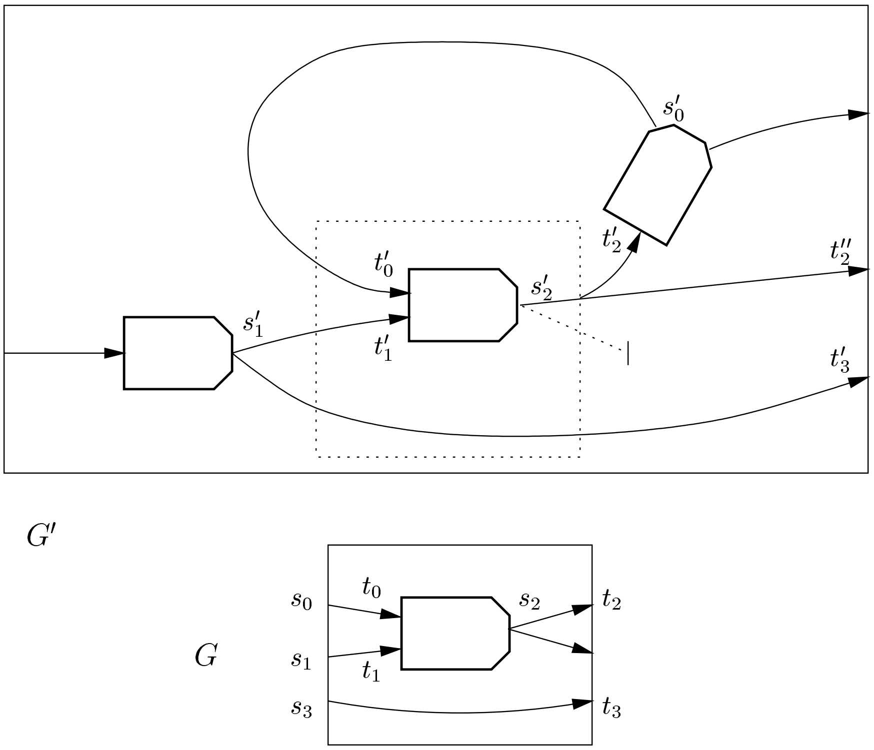
**Definition 19 (insertion contexts)** A *(closed, shallow, loose) insertion context* $A = (V, ctr, src)$ of arity $(m', n')$ to $(m, n)$, written $A : (m', n') \to (m, n)$, comprises a set $V$ of *nodes*, a *control* map $ctr : V \to \mathcal{K}$ assigning a control in $\mathcal{K}$ to each node in $V$, and a *source* map $src : T \to S$ assigning a *source (port)* in $S$ to each *target (port)* in $T$, where

- the *source set* $S \stackrel{\text{def}}{=} \sum_{v \in V}[arout(v)] + [m] + [n']$ 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* $T \stackrel{\text{def}}{=} \sum_{v \in V}[arin(v)] + [n] + [m']$ comprises the *argument* targets for each $v \in V$, the *output* targets indexed by $[n]$, and the *downput* targets indexed by $[m']$.

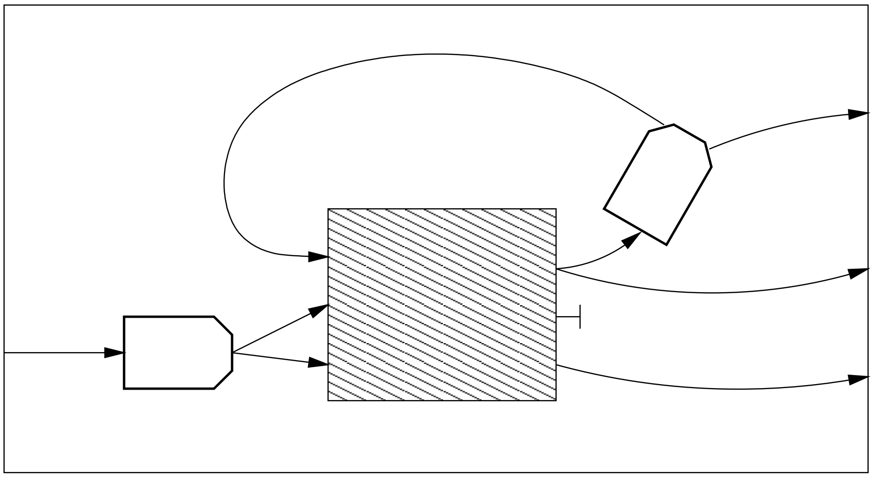
Furthermore the looseness condition must be satisfied (see nomenclature defined below):

$$\text{LOOSE}: \ src(T^{down}) \cap S^{up} = \emptyset \, . \qquad \blacksquare$$

We shall often call an insertion context an *insertion*. Note that an action graph $G$ is just an insertion context $A$ as above in which $m' = n' = 0$.




Another embedding



The corresponding context

Figure 11: An embedding and its associated action context

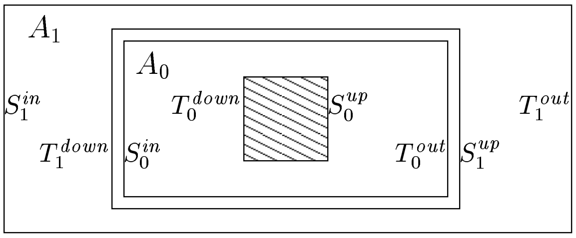



Figure 12: The composition of two insertions

---

The looseness condition, which was mentioned in Section 4, precludes a "tight" loop from the back of the hole to the front, such as is formed by the trace operator in action calculi. It does not preclude such a loop proceeding via a control, which indeed occurs in Figure 11; thus, just as with embeddings, our contexts permit only limited power of reflexion. We shall have to ensure that all contexts which we create – e.g. by composition – satisfy LOOSE. We shall continue to omit the adjectives "closed", "shallow" and "loose" in this paper.

**Nomenclature** As for action graphs, there are induced injections for an insertion as follows:

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

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

$$\begin{array}{rcl} S & = & S^{bind} \uplus S^{in} \uplus S^{up} \\ T & = & T^{arg} \uplus T^{out} \uplus T^{down} \ . \end{array}$$

**Definition 20 (The category of insertions)** The *category* $\mathbf{PIns}_0$ *of insertions* has as objects *profiles* of the form $p = (m, n, U)$, where $U$ is a set. The profile $\epsilon \stackrel{\text{def}}{=} (0, 0, \emptyset)$ is called the *empty* profile. $\mathbf{PIns}_0$ has arrows $A : (m', n', U') \to (m, n, U)$, where $U' \subseteq U$ and the insertion context $A$ has node-set $U \setminus U'$.

The *identity* insertion $\mathsf{id}_{m,n,U} = (\emptyset, \emptyset_{\mathcal{K}}, src)$ has

$$\begin{array}{l} S = \emptyset + [m] + [n] = S^{in} \uplus S^{up} \\ T = \emptyset + [n] + [m] = T^{out} \uplus T^{down} \end{array} \qquad src : \begin{cases} down(i) \mapsto in(i), \ i \in [m] \\ out(j) \mapsto up(j), \ j \in [n] \ . \end{cases}$$

Now let $A_i = (V_i, ctr_i, src_i) : p_i \to p_{i+1}$ be two insertions, where $p_i = (m_i, n_i, U_i)$ $(i = 0, 1)$. Their *composition*, illustrated in Figure 12 by nesting $A_0$ inside $A_1$, is

$$A_1 \oplus A_0 \stackrel{\text{def}}{=} A_2 = (V_2, ctr_2, src_2) : p_0 \to p_2 \ ,$$

where $V_2 \stackrel{\text{def}}{=} V_1 \uplus V_0$ and $ctr_2 \stackrel{\text{def}}{=} ctr_1 \uplus ctr_0$ (thus determining $ar_2 = ar_1 \uplus ar_0$ etc), and

$$\begin{array}{rcl} S_2 & \stackrel{\text{def}}{=} & \sum_{v \in V_2}[arout_2(v)] + [m_2] + [n_0] \quad = \quad (S_1^{bind} \uplus S_0^{bind}) \uplus S_1^{in} \uplus S_0^{up} \\ T_2 & \stackrel{\text{def}}{=} & \sum_{v \in V_2}[arin_2(v)] + [n_2] + [m_0] \quad = \quad (T_1^{arg} \uplus T_0^{arg}) \uplus T_1^{out} \uplus T_0^{down} \ . \end{array}$$

Note (see Figure 12) 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 $\sigma_i : S_i \to S_2$ $(i = 0, 1)$ which describe how sources of $A_0$ and $A_1$ "become" sources of $A_2$:

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

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

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

■

We have adopted the symbol "$\oplus$" for composition in $\mathbf{PIns}_0$, as a reminder that the node sets of two insertions composed in this category are always disjoint. We earlier observed that an action graph $G$ is just an insertion context whose hole has arity $(m', n') = (0, 0)$. If $G$ has node set $V$, we may now identify it with the arrow $G : \epsilon \to (m, n, V)$ of $\mathbf{PIns}_0$.

The above definition of $\mathbf{PIns}_0$ is justified by two propositions.

**Proposition 21** *If $A_0$ and $A_1$ satisfy* LOOSE, *then so does $A_2 = A_1 \oplus A_0$.*

**Proof** We require $t \in T_0^{down} \implies src_2(t) \notin S_0^{up}$. We have $src_2(t) = \sigma_0 src_0(t)$, and we consider cases for $src_0(t)$. By LOOSE for $A_0$ it cannot be in $S_0^{up}$. There are two other cases:

**Case** $src_0(t) \in S_0^{bind}$. Then $src_2(t) = src_0(t) \in S_0^{bind}$.

**Case** $src_0(t) = in_0(i) \in S_0^{in}$. Then $src_2(t) = src_1(t_1)$ where $t_1 = down_1(i) \in T_1^{down}$, whence by LOOSE for $A_1$ we have $src_2(t) \in S_1^{bind} \uplus S_1^{in}$. ■

**Proposition 22** *For any $A$, $\mathsf{id} \oplus A = A = A \oplus \mathsf{id}'$, where $\mathsf{id}'$ and $\mathsf{id}$ are identities on the domain and codomain of $A$. Also composition is associative, i.e. for appropriately matching profiles $A_2 \oplus (A_1 \oplus A_0) = (A_2 \oplus A_1) \oplus A_0$.*

**Proof** (Outline) The identity properties are easy to establish. For the associativity of composition, let $A_i = (V_i, ctr_i, src_i) : p_i \to p_{i+1}$, for $i = 0, 1, 2$, where $p_i = (m_i, n_i, U_i)$. Using L and R to stand for left- and right- association, let

$$\begin{array}{rcl} A_L \stackrel{\text{def}}{=} (A_2 \oplus A_1) \oplus A_0 & = & (V_L, ctr_L, src_L) \\ A_R \stackrel{\text{def}}{=} A_2 \oplus (A_1 \oplus A_0) & = & (V_R, ctr_R, src_R) \ . \end{array}$$

Then it is easy to verify that

$$\begin{array}{rcl} V_L = V_R & = & V_2 \uplus V_1 \uplus V_0 \\ ctr_L = ctr_R & = & ctr_2 \uplus ctr_1 \uplus ctr_0 \ . \end{array}$$

Hence $A_L$ and $A_R$ have the same arity function $ar : V \to \mathbb{N}^2$ where $V = V_L = V_R$, and

$$\begin{array}{rcl} S_L = S_R & = & \sum_{v \in V}[arout(v)] + [m_3] + [n_0] \\ T_L = T_R & = & \sum_{v \in V}[arin(v)] + [n_3] + [m_0] \ . \end{array}$$

It remains to prove that $src_L = src_R$; the detailed case analysis is in the Appendix. ■

Node renamings were defined in Definition 10, their action upon graphs in Definition 12, and their action upon inclusions in Definition 17. To prepare for the definition of $\mathbf{PCxt}_0$ we now define how they act upon insertions.

**Definition 23 (node renaming)** A node renaming $\rho$ on $U$ determines an *action* $\rho \bullet (-)$ upon profiles and insertions, as follows. Upon profiles, assuming $V \subseteq U$:

$$\rho \bullet (m, n, V) \stackrel{\text{def}}{=} (m, n, \rho V) \ .$$

Upon insertions, let $A = (V, ctr, src) : p_0 \to p_1$, with $p_i = (m_i, n_i, U_i)$ and $U_i \subseteq U$; then

$$\rho \bullet A \stackrel{\text{def}}{=} A' : \rho \bullet p_0 \to \rho \bullet p_1$$

where $A' = (V', ctr', src')$ is as follows:

$$\begin{array}{rcl} V' & \stackrel{\text{def}}{=} & \rho V \\ ctr'(\rho v) & \stackrel{\text{def}}{=} & ctr(v), \ \text{thus determining } ar' \text{ etc} \\ S' & \stackrel{\text{def}}{=} & \sum_{v \in V'}[arout'(v)] + [m_1] + [n_0] \\ T' & \stackrel{\text{def}}{=} & \sum_{v \in V'}[arin'(v)] + [n_1] + [m_0] \\ & & \quad \text{thus determining bijections } \rho^{\mathsf{S}} : S \rightleftarrows S' \text{ and } \rho^{\mathsf{T}} : T' \rightleftarrows T \\ src' & \stackrel{\text{def}}{=} & \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}} \ . \end{array}$$

■

**Proposition 24** *If $A$ satisfies* LOOSE *then so does $\rho \bullet A$.*

**Proof** Immediate from $src' \stackrel{\text{def}}{=} \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}}$, since $\rho^{\mathsf{S}}$ and $\rho^{\mathsf{T}}$ are the identity on input sources and output targets respectively. ■

Note in particular that $\rho \bullet A$ is only defined when the codomain of $A$ is a subset of the domain of $\rho$. For what follows we need a few properties of node renamings:

**Proposition 25** *Node renamings satisfy the following, whenever they are defined:*

$$\begin{array}{ll}
(1) & \rho\bullet(\mathsf{id}_{m,n,U}) = \mathsf{id}_{m,n,\rho U} \\
(2) & \rho\bullet(A_1 \oplus A_0) = \rho\bullet A_1 \oplus \rho\bullet A_0 \\
(3) & \mathsf{Id}\bullet A = A \\
(4) & (\rho' \circ \rho)\bullet A = \rho'\bullet(\rho\bullet A) \\
(5) & \rho\bullet A = \rho'\bullet A \qquad \text{if } A \text{ has node set } V \text{ and } \rho \restriction V = \rho' \restriction V \ .
\end{array}$$

Thus, in particular, each node renaming with domain $U$ is a functor on the full subcategory of $\mathbf{PIns}_0$ containing those objects whose node sets are subsets of $U$.

We are now ready for the main definition of this section. As mentioned in Section 4, we shall define a category $\mathbf{PCxt}_0$ of contexts which "trail" the nodes of a graph; this role is played by node renamings. The definition is followed immediately by a justifying proposition.

**Definition 26 (the category of contexts)** The category $\mathbf{PCxt}_0$ of *(closed, shallow, loose) contexts* has profiles as objects, and its arrows – the contexts – take the form

$$(A, \rho) : (m', n', U') \to (m, n, U)$$

where $\rho : U' \rightarrowtail U$ is called a *trail* and $A : (m', n', \rho U') \to (m, n, U)$ is an insertion. The identity contexts are $\mathsf{id}_{m,n,U} \stackrel{\text{def}}{=} (\mathsf{id}_{m,n,U}, \mathsf{Id}_U)$, and the composition of two contexts $C_i = (A_i, \rho_1) : p_i \to p_{i+1}$, for $i = 0, 1$, is defined as follows:

$$C_1\, C_0 \stackrel{\text{def}}{=} (A_1 \oplus \rho_1\bullet A_0, \rho_1 \circ \rho_0) \ . \qquad \blacksquare$$

**Proposition 27** *The identities of* $\mathbf{PCxt}_0$ *satisfy* $\mathsf{id}\, C = C = C\, \mathsf{id}'$ *for appropriate domain and codomain; moreover composition satisfies* $(C_2\, C_1)\, C_0 = C_2\, (C_1\, C_0)$ .

**Proof** For the first, let $C = (A, \rho) : (m', n', U') \to (m, n, U)$; for the second, let $C_i = (A_i, \rho_i)$, $i = 0, 1, 2$. Then:

$$\begin{array}{rcll}
\mathsf{id}\, C & = & (\mathsf{id}, \mathsf{Id}_U)\,(A, \rho) & \\
& = & (\mathsf{id} \oplus \mathsf{Id}_U\bullet A,\ \mathsf{Id}_U \circ \rho) & \\
& = & C & \text{Proposition 25(3) and identity in } \mathbf{PIns}_0 \\
& = & (A \oplus \rho\bullet\mathsf{id},\ \rho \circ \mathsf{Id}_{U'}) & \text{Proposition 25(1) and identity in } \mathbf{PIns}_0 \\
& = & (A, \rho)\,(\mathsf{id}, \mathsf{Id}'_U) & \\
& = & C\, \mathsf{id}' \ . &
\end{array}$$

$$\begin{array}{rcll}
(C_2\, C_1)\, C_0 & = & (A_2 \oplus \rho_2\bullet A_1,\ \rho_2 \circ \rho_1)\,(A_0, \rho_0) & \\
& = & ((A_2 \oplus \rho_2\bullet A_1) \oplus (\rho_2 \circ \rho_1)\bullet A_0,\ (\rho_2 \circ \rho_1) \circ \rho_0) & \\
& = & (A_2 \oplus (\rho_2\bullet A_1 \oplus (\rho_2 \circ \rho_1)\bullet A_0),\ \rho_2 \circ (\rho_1 \circ \rho_0)) & \text{Proposition 22} \\
& = & (A_2 \oplus \rho_2\bullet(A_1 \oplus \rho_1\bullet A_0), \rho_2 \circ (\rho_1 \circ \rho_0)) & \text{Proposition 25(2,4)} \\
& = & C_2\,(C_1\, C_0) \ . &
\end{array}$$

$\blacksquare$

Recall that a graph $G : (m, n)$ with node set $V$ is essentially an insertion context $G : \epsilon \to (m, n, V)$ in $\mathbf{PIns}_0$. The corresponding context in $\mathbf{PCxt}_0$ is

$$(G, \emptyset_V) : \epsilon \to (m, n, V) ,$$

where $\emptyset_V$ stands for the empty renaming with codomain $V$. When working in $\mathbf{PCxt}_0$ we shall often abbreviate $(G, \emptyset_V)$ to $G$.

# 8 Functors from embeddings to contexts

At the end of the previous section we noticed that graphs $(V, \mathit{ctr}, \mathit{src}) : (m, n)$ correspond to contexts $\epsilon \to (m, n, V)$. In this section and the next we show that this correspondence leads to an isomorphism of categories

$$\mathbf{GEmb}_0 \cong \epsilon/\mathbf{PCxt}_0 ,$$

where $\epsilon/\mathbf{PCxt}_0$ is a *coslice* category of $\mathbf{PCxt}_0$; this is a simple general notion that we shall now recall. A consequence of this result is that slice sums in $\mathbf{GEmb}_0$ correspond naturally to relative pushouts in $\mathbf{PCxt}_0$, which is crucial in deriving labelled transition systems.

**Definition 28 (coslice categories)** Let $\mathbf{C}$ be a category and let $C$ be an object of $\mathbf{C}$. Define the *coslice* category $C/\mathbf{C}$ as the following category:

*Objects*: arrows $f : C \to C'$ of $\mathbf{C}$ with fixed domain $C$.

*Arrows*: if $f : C \to C'$ and $g : C \to C''$ are two objects, then arrows $h : f \to g$ are arrows $h : C' \to C''$ of $\mathbf{C}$ such that $hf = g$. ■

**Remark** The categories $C/\mathbf{C}$ and $\mathbf{C}$ are naturally related by a functor $\mathsf{Cod} : C/\mathbf{C} \to \mathbf{C}$ mapping each object $f : C \to C'$ to its codomain $C'$ and each arrow $h : f \to hf$ to itself. See the discussion in Section 4. The functor $\mathsf{Cod}$ is an example of a discrete cofibration.

To establish an isomorphism $\mathcal{C} : \mathbf{GEmb}_0 \xrightarrow{\simeq} \epsilon/\mathbf{PCxt}_0$, we begin by defining an isomorphism $\mathcal{A} : \mathbf{GInc}_0 \xrightarrow{\simeq} \epsilon/\mathbf{PIns}_0$. The definition of $\mathcal{C}$ will then be a straightforward modification of $\mathcal{A}$.

**Definition 29 (functor on inclusions)** The functor $\mathcal{A} : \mathbf{GInc}_0 \to \epsilon/\mathbf{PIns}_0$ is defined on objects by $\mathcal{A}(G) \stackrel{\text{def}}{=} G : \epsilon \to (m, n, V)$, where $G = (V, \mathit{ctr}, \mathit{src}) : (m, n)$. Now let $\eta : G_0 \to G_1$ be an inclusion. Suppose

$$G_i = (V_i, \mathit{ctr}_i, \mathit{src}_i) : (m_i, n_i),\ i = 0, 1$$

with $V_0 \subseteq V_1$. Then

$$\mathcal{A}(\eta) \stackrel{\text{def}}{=} A = (V_A, \mathit{ctr}_A, \mathit{src}_A) : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$$

where $V_A \stackrel{\text{def}}{=} V_1 \setminus V_0$ and $ctr_A \stackrel{\text{def}}{=} ctr_1 \restriction V_A$ (thus determining $ar_A = ar_1 \restriction V_A$ etc), and

$$\begin{array}{rcl} S_A & \stackrel{\text{def}}{=} & \sum_{v \in V_A}[arout_A(v)] + [m_1] + [n_0] \quad = \quad S_A^{bind} \uplus S_1^{in} \uplus S_1^{up} \\ T_A & \stackrel{\text{def}}{=} & \sum_{v \in V_A}[arin_A(v)] + [n_1] + [m_0] \quad = \quad T_A^{arg} \uplus T_1^{out} \uplus T_1^{down} \; . \end{array}$$

Note that $S_1^{up}$ and $T_1^{down}$ are in bijection with $T_0^{out}$ and $S_0^{in}$ respectively. Finally we define $src_A$:

$$src_A(t) \stackrel{\text{def}}{=} \begin{cases} \left\{ \begin{array}{ll} up_1(j) & \text{if } \eta^{\mathsf{T}}(t) \text{ defined, i.e. } \eta^{\mathsf{T}} t = out_0(j), j \in [n_0] \\ src_1(t) & \text{if } \eta^{\mathsf{T}}(t) \text{ undefined} \end{array} \right\} \\ \qquad\qquad\qquad \text{if } t \in T_A^{arg} \uplus T_1^{out} \; ; \\ \eta^{\mathsf{S}} in_0(i) \qquad\quad \text{if } t = down_1(i) \in T_1^{down}, i \in [m_0] \; . \end{cases}$$

■

We now justify this definition by three propositions.

**Proposition 30** *$\mathcal{A}(\eta)$ satisfies* LOOSE.

**Proof** From the definition of $src_A$, if $t \in T_1^{down}$ then $src_A(t) \in \eta^{\mathsf{S}}(S_0^{in}) \subseteq S_A^{bind} \uplus S_1^{in}$. ■

**Proposition 31** *For every two inclusion embeddings $G_0 \stackrel{\eta_0}{\to} G_1 \stackrel{\eta_1}{\to} G_2$, $\mathcal{A}(\eta_1\eta_0) = \mathcal{A}(\eta_1) \oplus \mathcal{A}(\eta_0)$. Moreover for every graph $G$, $\mathcal{A}(\mathsf{id}_G) = \mathsf{id}_G$ and $\mathcal{A}(\mathbf{0}_G) = G$ .*

**Proof** (outline) The full proof is given in the Appendix. To show that $\mathcal{A}(\mathsf{id}_G) = \mathsf{id}_G$ and that $\mathcal{A}(\mathbf{0}_G) = G$ is easy; recall that $\mathbf{0}_G$ is the unique embedding of the empty graph in $G$. To show that $\mathcal{A}(\eta_1\eta_0) = \mathcal{A}(\eta_1) \oplus \mathcal{A}(\eta_0)$ we have to set out the components of the left- and right-hand sides in detail. It then emerges that all components are clearly identical except the source functions; to prove that these agree is routine but requires a detailed case analysis. ■

Combining the two propositions above gives the following corollary:

**Corollary 32** *The construction $\mathcal{A}$ defines a functor* $\mathbf{GInc}_0 \to \epsilon/\mathbf{PIns}_0$.

**Proof** Proposition 30 guarantees that the images of embeddings are indeed arrows of $\mathbf{PIns}_0$, while Proposition 31 ensures the functoriality of the construction $\mathcal{A}$. The one thing that we still have to check is that $\mathcal{A}$ is well-defined in the sense that for every inclusion $\eta : G_0 \to G_1$, we have that $\mathcal{A}(\eta) : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$; in other words, that $\mathcal{A}(\eta) \oplus G_0 = G_1$. To prove this, we use the fact that $\mathcal{A}(\mathbf{0}_G) = G$. In fact we obviously have that $\mathbf{0}_{G_1} = \eta\,\mathbf{0}_{G_0}$; By functoriality we then have that $\mathcal{A}(\mathbf{0}_{G_1}) = \mathcal{A}(\eta) \oplus \mathcal{A}(\mathbf{0}_{G_0})$. Thus we also have that $G_1 = \mathcal{A}(\eta) \oplus G_0$. ■

It is immediately clear that, on objects, $\mathcal{A}$ is a bijection. To prove that it is in fact an isomorphism of categories we then have to show that it is *full and faithful* [18], in other words that for every two graphs $G_0$ and $G_1$ the function

$$\begin{array}{rccc} \mathcal{A}_{G_0,G_1} : & \mathbf{GInc}_0(G_0, G_1) & \longrightarrow & \epsilon/\mathbf{PIns}_0(\mathcal{A}(G_0), \mathcal{A}(G_1)) \\ & \eta & \mapsto & \mathcal{A}(\eta) \end{array}$$

$$\begin{array}{ccc} G_0 & \xrightarrow{\quad\eta\quad} & G_1 \\ & \big\downarrow\,\rho\bullet(-) & \\ \rho\bullet G_0 & \xrightarrow{\quad\rho\bullet\eta\quad} & \rho\bullet G_1 \end{array} \qquad \stackrel{\mathcal{A}}{\longmapsto} \qquad \begin{array}{ccc} \mathcal{A}(G_0) & \xrightarrow{\quad\mathcal{A}(\eta)\quad} & \mathcal{A}(G_1) \\ & \big\downarrow\,\rho\bullet(-) & \\ \rho\bullet\mathcal{A}(G_0) & \xrightarrow{\quad\rho\bullet\mathcal{A}(\eta)\quad} & \rho\bullet\mathcal{A}(G_1) \end{array}$$

Figure 13: How the functor $\mathcal{A}$ treats a node renaming

---

is a bijection. We leave this to the next section where we will make use of inverse functions $\mathcal{D}_{G_0,G_1}$ that we shall define later in this section.

Before extending $\mathcal{A}$ to a functor on all embeddings, we have to show how it interacts with node renaming. The action of renaming on inclusions and on contexts was defined in Definitions 17 and 23 respectively. Figure 13 shows how these two actions are related by the functor $\mathcal{A}$. This relationship is made precise by the following two propositions (recall that $\mathcal{A}$ is the identity on objects).

**Proposition 33** *For any $\eta : G_0 \to G_1$, the context $\rho\bullet\mathcal{A}(\eta)$ is an arrow $\rho\bullet G_0 \to \rho\bullet G_1$ in $\epsilon/$**PIns**$_0$.*

**Proof** This property is an immediate consequence of Proposition 25(2) as

$$\rho\bullet\mathcal{A}(\eta) \oplus \rho\bullet G_0 = \rho\bullet(\mathcal{A}(\eta) \oplus G_0) = \rho\bullet G_1 \ . \qquad \blacksquare$$

**Proposition 34** *The functor $\mathcal{A}$ preserves node renaming, i.e. $\mathcal{A}(\rho\bullet\eta) = \rho\bullet\mathcal{A}(\eta)$ .*

**Proof** See the Appendix. It is straightforward, but lays out quite a lot of data. ■

**Definition 35 (functor on embeddings)** The functor $\mathcal{C} : \mathbf{GEmb}_0 \to \epsilon/\mathbf{PCxt}_0$ from embeddings to contexts is defined on objects, i.e. graphs, essentially as $\mathcal{A}$ is. That is

$$\mathcal{C}(G) \stackrel{\text{def}}{=} \epsilon \xrightarrow{(G,\emptyset_V)} (m, n, V) \ , \text{ where } G = (V, \mathit{ctr}, \mathit{src}) : (m, n) \ .$$

Let $\phi = \eta\rho$ be the unique factorisation of an embedding $\phi$ into an inclusion $\eta$ and a node isomorphism $\rho$; then $\mathcal{C}$ is defined on the embedding $\phi$ by

$$\mathcal{C}(\phi) \stackrel{\text{def}}{=} (\mathcal{A}(\eta), \phi^{\mathsf{V}}) \ . \qquad \blacksquare$$

Fortunately we have already done almost all the work to ensure that:

**Proposition 36** $\mathcal{C} : \mathbf{GEmb}_0 \to \epsilon/\mathbf{PCxt}_0$ *is a functor.*

**Proof** It is easy to check that $\mathcal{C}$ takes identities to identities. To show that it preserves composition, let $\phi = \phi_1\phi_0$. Let $\phi_i = \eta_i\rho_i$ be the unique factorisations of $\phi_1$ and $\phi_0$ assured by Proposition 15. Then $\phi = \eta_1\rho_1\eta_0\rho_0$. But by Proposition 18 we have that $\rho_1\eta_0 = (\rho_1 \centerdot \eta_0)\rho_1$, where $\rho \centerdot \eta_0$ is an inclusion. Thus we have the unique factorisation $\phi = (\eta_1(\rho_1 \centerdot \eta_0))(\rho_1\rho_0)$; it follows that

$$\mathcal{C}(\phi) \;=\; (\mathcal{A}(\eta_1(\rho_1 \centerdot \eta_0)),\; \phi^{\mathsf{V}})\,.$$

On the other hand

$$\begin{array}{rcll}
\mathcal{C}(\phi_1)\mathcal{C}(\phi_0) & = & (\mathcal{A}(\eta_1),\, \phi_1^{\mathsf{V}})\,(\mathcal{A}(\eta_0),\, \phi_0^{\mathsf{V}}) & \text{by definition of } \mathcal{C} \\
& = & (\mathcal{A}(\eta_1) \oplus \phi_1^{\mathsf{V}} \centerdot \mathcal{A}(\eta_0),\, \phi_1^{\mathsf{V}} \circ \phi_0^{\mathsf{V}}) & \text{by definition of composition} \\
& = & (\mathcal{A}(\eta_1) \oplus \rho_1 \centerdot \mathcal{A}(\eta_0),\, \phi^{\mathsf{V}}) & \\
& = & (\mathcal{A}(\eta_1) \oplus \mathcal{A}(\rho_1 \centerdot \eta_0),\, \phi^{\mathsf{V}}) & \text{by Proposition 34} \\
& = & (\mathcal{A}(\eta_1(\rho_1 \centerdot \eta_0)),\, \phi^{\mathsf{V}}) & \text{by Proposition 31} \\
& = & \mathcal{C}(\phi)\,. &
\end{array}$$

We are now left with showing that $\mathcal{C}$ is well defined, i.e., that for every $\phi : G_0 \to G_1$, $\mathcal{C}(G_1) = \mathcal{C}(\phi)\mathcal{C}(G_0)$. To this purpose observe that if $\phi : G_0 \to G_1$ and has decomposition $\phi = \eta\rho$, then we have that $\mathbf{0} = \rho\centerdot\mathbf{0}$ and thus that $\rho\centerdot\mathbf{0}_{G_0} = \mathbf{0}_{\rho \centerdot G_0}$; hence the following diagram commutes:

$$\begin{tikzcd}
\mathbf{0} = \rho\centerdot\mathbf{0} \arrow[r, "\rho\centerdot\mathbf{0}_{G_0}"] \arrow[dr, "\mathbf{0}_{G_1}"'] & \rho\centerdot G_0 \arrow[d, "\eta"] \\
& G_1
\end{tikzcd}$$

We now have the following chain of equalities:

$$\begin{array}{rcll}
\mathcal{C}(G_1) & = & (G_1, \emptyset_{V_1}) & \\
& = & (\mathcal{A}(\mathbf{0}_{G_1}),\, \emptyset_{V_1}) & \text{by Proposition 31} \\
& = & (\mathcal{A}(\eta\,(\rho\centerdot\mathbf{0}_{G_0})),\, \emptyset_{V_1}) & \text{by commutativity of the diagram above} \\
& = & (\mathcal{A}(\eta) \oplus \mathcal{A}(\rho\centerdot\mathbf{0}_{G_0}),\, \emptyset_{V_1}) & \text{by functoriality of } \mathcal{A}\text{, Proposition 31} \\
& = & (\mathcal{A}(\eta) \oplus \rho\centerdot\mathcal{A}(\mathbf{0}_{G_0}),\, \emptyset_{V_1}) & \text{by Proposition 34} \\
& = & (\mathcal{A}(\eta) \oplus \phi^{\mathsf{V}}\centerdot\mathcal{A}(\mathbf{0}_{G_0}),\, \phi^{V}\emptyset_{V_0}) & \\
& = & (\mathcal{A}(\eta) \oplus \phi^{\mathsf{V}}\centerdot G_0,\, \phi^{V}\emptyset_{V_0}) & \text{by Proposition 31} \\
& = & (\mathcal{A}(\eta), \phi^{V})(G_0, \emptyset_{V_0}) & \text{by definition of composition of contexts} \\
& = & \mathcal{C}(\phi)\mathcal{C}(G_0) & \text{by definition of } \mathcal{C}.
\end{array}$$

■

From now on we shall say that $\phi : G_0 \to G_1$ in $\mathbf{GEmb}_0$ is *over* $C : \mathcal{C}(G_0) \to \mathcal{C}(G_1)$ if $\mathcal{C}(\phi) = C$. Similarly in $\mathbf{GInc}_0$, $\eta : G_0 \to G_1$ is *over* $A : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$ if $\mathcal{A}(\eta) = A$.

We now turn to the full and faithfulness problem: Given a context $C : \mathcal{C}(G_0) \to \mathcal{C}(G_1)$, show that there is a unique embedding $\phi : G_0 \to G_1$ such that $\mathcal{C}(\phi) = C$. We begin with the simpler problem: Given an insertion $A : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$, show that there is a unique inclusion $\eta : G_0 \to G_1$ such that $\mathcal{A}(\eta) = A$. We shall achieve this by providing a function $\mathcal{D}_{G_0,G_1}$ which is shown in the next section to be inverse to $\mathcal{A}_{G_0,G_1}$.

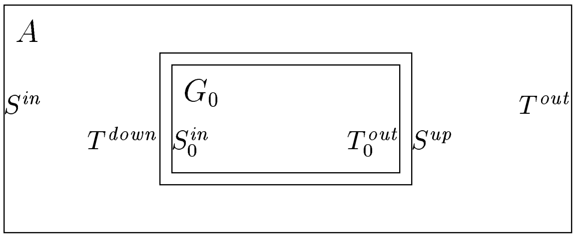



Figure 14: Applying an insertion context to a graph

---

**Definition 37 (inclusion from insertion)** Let $G_0$ and $G_1$ be graphs. We define a function $\mathcal{D}_{G_0,G_1} : \epsilon/\mathbf{PIns}_0(\mathcal{A}(G_0), \mathcal{A}(G_1)) \to \mathbf{GInc}_0(G_0, G_1)$ from insertions to inclusion embeddings.

Let $A : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$ be an insertion, where $G_i = (V_i, ctr_i, src_i) : (m_i, n_i)$ for $i = 0, 1$. Recall that by definition $G_1 = A \oplus G_0$ . Thus if $A = (V, ctr, src)$ then the components of $G_1 = A \oplus G_0 = (V_1, ctr_1, src_1)$ are as follows (a special case of insertion composition, see Figure 14):

$$
\begin{array}{rcl}
V_1 & = & V \uplus V_0 \\
ctr_1 & = & ctr \uplus ctr_0 \\
S_1 & = & (S^{bind} \uplus S_0^{bind}) \uplus S^{in} \\
T_1 & = & (T^{arg} \uplus T_0^{arg}) \uplus T^{out} \\
& \text{and} & \\
src_1(t) & = & \begin{cases} \sigma\, src(t) & \text{if } t \in T^{arg} \uplus T^{out} \\ \sigma_0 src_0(t) & \text{if } t \in T_0^{arg} \end{cases} \\
& \text{where} & \\
\sigma_0(s) & = & \begin{cases} src\, down(i) & \text{if } s = in_0(i) \in S_0^{in} \\ s & \text{if } s \in S_0^{bind} \end{cases} \\
\sigma(s) & = & \begin{cases} \sigma_0 src_0\, out(j) & \text{if } s = up(j) \in S^{up} \\ s & \text{if } s \in S^{bind} \uplus S^{in}\ . \end{cases}
\end{array}
$$

The components of the inclusion embedding $\mathcal{D}_{G_0,G_1}(A) = \theta : G_0 \to G_1$ are then defined by

$$
\begin{array}{rcl}
\theta^{\mathsf{V}} : V_0 \hookrightarrow V_1 & \stackrel{\text{def}}{=} & v \mapsto v \ \ (v \in V_0) \\
\theta^{\mathsf{S}} : S_0 \to S_1 & \stackrel{\text{def}}{=} & \sigma_0 \\
\theta^{\mathsf{T}} : T_1 \rightharpoonup T_0 & \stackrel{\text{def}}{=} & t \mapsto \begin{cases} t & \text{if } t \in T_0^{arg} \\ \left. \begin{cases} \text{undefined} & \text{if } src(t) \in S^{bind} \uplus S^{in} \\ out_0(j) & \text{if } src(t) = up(j) \in S^{up} \end{cases} \right\} & \text{if } t \in T^{arg} \uplus T^{out}\ . \end{cases}
\end{array}
$$

■

This definition is justified by

**Proposition 38** *$\theta = \mathcal{D}_{G_0,G_1}(A)$ is an inclusion embedding.*

**Proof** First, $\theta^{\mathsf{V}}$ is an inclusion map, so it remains to check that the five embedding axioms hold. E-CTR is immediate.

For E-BIND, first note that $\theta^{\mathsf{S}}$ is well defined, i.e. $\sigma_0(s) \in S_1$ for each $s \in S_0$; this is because, in the second case of $\sigma_0$'s definition, $src(down(i)) \in S^{up}$ is impossible by LOOSE for $A$. Then E-BIND follows noting that $\sigma_0(s) \in S_0^{bind}$ iff $s \in S_0^{bind}$.

For E-ARG, note similarly that $\theta^{\mathsf{T}}(t) \in T_0^{arg}$ iff $t \in T_0^{arg}$.

For E-SRC, suppose $\theta^{\mathsf{T}}(t)$ is defined.

**Case** $t \in T^{arg} \uplus T^{out}$ and $src(t) = up(j) \in S^{up}$: then $src_1(t) = \sigma\, up(j) = \sigma_0 src_0\, out(j) = \theta^{\mathsf{S}} src_0 \theta^{\mathsf{T}}(t)$.

**Case** $t \in T_0^{arg}$: then $src_t(t) = \sigma_0 src_0(t) = \theta^{\mathsf{S}} src_0 \theta^{\mathsf{T}}(t)$.

For E-TARGS, we require that if $\theta^{\mathsf{T}}(t)$ is undefined then $src_1(t) \notin S_0^{bind}$. If $\theta^{\mathsf{T}}(t)$ is undefined then $t \in T^{arg} \uplus T^{out}$ and $src(t) \in S^{bind} \uplus S^{in}$. Hence $src_1(t) \in \sigma(S^{bind} \uplus S^{in}) = S^{bind} \uplus S^{in}$, disjoint from $S_0^{bind}$. ■

The last task of this section is to define a function $\mathcal{E}_{G_0,G_1}$ which we intend as an inverse for $\mathcal{C}_{G_0,G_1}$, just as we intend $\mathcal{D}_{G_0,G_1}$ as an inverse for $\mathcal{A}_{G_0,G_1}$. The actual proof that these are indeed inverses will be carried out in the next section.

**Definition 39 (embedding from context)** The function $\mathcal{E}_{G_0,G_1}$ from contexts to embeddings is defined as follows. Let $C = (A, \rho)$ be a context such that $(G_1, \emptyset_{V_1}) = (A, \rho)(G_0, \emptyset_{V_0})$. In other words $C$ is an arrow $\mathcal{C}(G_0) \to \mathcal{C}(G_1)$ in $\epsilon/\mathbf{PCxt}_0$. Then, by definition, $G_1 = A \oplus \rho \centerdot G_0$. Moreover we also have $\rho : G_0 \to \rho \centerdot G_0$, and $\mathcal{D}_{\rho \centerdot G_0, G_1}(A) : \rho \centerdot G_0 \to G_1$.We therefore define

$$\mathcal{E}_{G_0,G_1}(C) : G_0 \to G_1 \stackrel{\text{def}}{=} \mathcal{D}_{\rho \centerdot G_0, G_1}(A)\, \rho\,. \qquad ■$$

Thus $\mathcal{E}$ is built from $\mathcal{D}$, just as $\mathcal{C}$ is built from $\mathcal{A}$.

# 9 Functor inverses

In this section we demonstrate that the function

$$\mathcal{C}_{G_0,G_1} : \mathbf{GEmb}_0(G_0, G_1) \to \epsilon/\mathbf{PCxt}_0(\mathcal{C}(G_0), \mathcal{C}(G_1))$$

and the function

$$\mathcal{E}_{G_0,G_1} : \epsilon/\mathbf{PCxt}_0(\mathcal{C}(G_0), \mathcal{C}(G_1)) \to \mathbf{GEmb}_0(G_0, G_1)$$

are inverses of each other. This implies that $\mathcal{C}_{G_0,G_1}$ is a bijection and thus that $\mathcal{C}$ is an isomorphism of categories. This main result depends upon the analogous property that the function $\mathcal{A}_{G_0,G_1}$ and the function $\mathcal{D}_{G_0,G_1}$ invert one another. We prove the latter first.

For the remainder of this section, when no confusion arises, we will omit for readability the subscript $G_0, G_1$ from $\mathcal{C}_{G_0,G_1}$, $\mathcal{E}_{G_0,G_1}$, $\mathcal{A}_{G_0,G_1}$ and $\mathcal{D}_{G_0,G_1}$.

**Proposition 40** *Let $A : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$ be an insertion context. Then $\mathcal{A}(\mathcal{D}(A)) = A$.*

**Proof** First we construct $\theta = \mathcal{D}(A) : G_0 \to G_1$ as in Definition 37. We now apply the definition of $\mathcal{A}$, Definition 29, to give us

$$A' = \mathcal{A}(\theta) = (V', ctr', src') : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$$

whose first two components are readily found to agree with those of $A$. Furthermore:

$$\begin{array}{rcccl} S' = S & = & \sum_{v \in V}[arout(v)] + [m_1] + [n_0] & = & S^{bind} \uplus S^{in} \uplus S^{up} \\ T' = T & = & \sum_{v \in V}[arin(v)] + [n_1] + [m_0] & = & T^{arg} \uplus T^{out} \uplus T^{down} \; . \end{array}$$

We also have the source function of $A'$:

$$src'(t) \stackrel{\text{def}}{=} \begin{cases} \left\{ \begin{array}{ll} up(j) & \text{if } \theta^{\mathsf{T}}(t) \text{ defined, i.e. } \theta^{\mathsf{T}} t = out_0(j), j \in [n_0] \\ src_1(t) & \text{if } \theta^{\mathsf{T}}(t) \text{ undefined} \end{array} \right\} \\ \qquad\qquad\qquad \text{if } t \in T^{arg} \uplus T^{out} \; ; \\ \theta^{\mathsf{S}} in_0(i) \qquad\quad \text{if } t = down(i) \in T^{down}, i \in [m_0] \; . \end{cases}$$

It therefore remains to prove that $src' = src$. There are three cases:

**Case** $t \in T^{arg} \uplus T^{out}$ and $src(t) \in S^{bind} \uplus S^{in}$. Then by definition $\theta^{\mathsf{T}}(t)$ is undefined, hence $src'(t) \stackrel{\text{def}}{=} \sigma\, src(t) = src(t)$.

**Case** $t \in T^{arg} \uplus T^{out}$ and $src(t) = up(j) \in S^{up}$. Then by definition $\theta^{\mathsf{T}}(t) = out_0(j)$, hence $src'(t) \stackrel{\text{def}}{=} up(j) = src(t)$.

**Case** $t = down(i) \in T^{down}$. Then $src'(t) \stackrel{\text{def}}{=} \theta^{\mathsf{S}} in_0(i) = \sigma_0 in_0(i) = src\, down(i) = src(t)$.

This concludes the proof that $A' = A$, and hence that $\mathcal{A}(\mathcal{D}(A)) = A$. ∎

**Proposition 41** *Let $\eta : G_0 \to G_1$ be an inclusion embedding. Then $\mathcal{D}(\mathcal{A}(\eta)) = \eta$.*

**Proof** The argument is somewhat detailed and can be found in the Appendix. ∎

Proposition 40 and Proposition 41 say that $\mathcal{D}$ is both a left and a right inverse for $\mathcal{A}$, and thus that $\mathcal{A}$ is full and faithful.. Since, as we said earlier, $\mathcal{A}$ is a bijection on objects, this is enough to conclude the following:

**Proposition 42** *The functor $\mathcal{A} : \mathbf{GInc}_0 \to \epsilon/\mathbf{PIns}_0$ is an isomorphism of categories.*

We shall now complete the work of the section by lifting the foregoing results to full embeddings and full contexts.

**Proposition 43** *Let $C : \mathcal{C}(G_0) \to \mathcal{C}(G_1)$ be an arbitrary context. Then $\mathcal{C}(\mathcal{E}(C)) = C$.*

**Proof** Let $C = (A, \rho)$ and $\phi = \mathcal{E}(C)$. Then $\phi = \eta\,\rho$ where $\eta = \mathcal{D}_{\rho \bullet G_0, G_1}(A)$ and $\rho = \phi^{\mathsf{V}}$. Hence

$$\begin{array}{rcll} \mathcal{C}(\mathcal{E}(C)) & = & \mathcal{C}(\eta\,\rho) & \\ & = & (\mathcal{A}(\eta),\, \phi^{\mathsf{V}}) & \text{by definition of } \mathcal{C} \\ & = & (A,\, \phi^{\mathsf{V}}) & \text{by Proposition 40} \\ & = & (A,\, \rho) & \\ & = & C\,. & \end{array}$$

■

**Proposition 44** *Let $\phi : G_0 \to G_1$ be an arbitrary embedding. Then $\mathcal{E}(\mathcal{C}(\phi)) = \phi$.*

**Proof** Let the unique factorisation of $\phi$ into an inclusion and a node renaming be $\phi = \eta\,\rho$, where $\rho : V_0 \rightarrowtail V_1 = \phi^{\mathsf{V}}$. Then $\mathcal{C}(\phi) = (\mathcal{A}(\eta),\, \rho)$ by definition. So

$$\begin{array}{rcll} \mathcal{E}(\mathcal{C}(\phi)) & = & \mathcal{D}_{\rho \bullet G_0, G_1}(\mathcal{A}(\eta))\,\rho & \text{by definition} \\ & = & \eta\,\rho & \text{by Proposition 41} \\ & = & \phi\,. & \end{array}$$

■

Finally, exactly as we did for $\mathcal{A}$, we deduce that

**Theorem 45** *The functor $\mathcal{C} : \mathbf{GEmb}_0 \to \epsilon/\mathbf{PCxt}_0$ is an isomorphism of categories.*

# 10 Relative pushouts for contexts

The last section concluded by stating that the category of graphs and graph embeddings $\mathbf{GEmb}_0$ is isomorphic to the coslice category $\epsilon/\mathbf{PCxt}_0$. In this section we use this result to show that $\mathbf{PCxt}_0$ possesses enough RPOs to ensure that a wide class of LTSs derived for action calculi will yield behavioural congruences.

We shall first define RPOs and prove a few results about them. For this pure category theory we denote objects by $A, B, C, \ldots$ and arrows by $f, g, h, \ldots$. When we come to apply the results we shall revert to our applied notation.

**Definition 46 (relative pushout)** In any category $\mathbf{C}$, let $f_i, h_i$ $(i = 0, 1)$ be arrows such that $h_0 f_0 = h_1 f_1$ (see Figure 15). A triple $g_0, g_1, g$ is called a *relative pushout (RPO) for $f_0, f_1$ wrt $h_0, h_1$* if $g_0 f_0 = g_1 f_1$ and $g g_i = h_i$, and moreover for any other triple $g'_0, g'_1, g'$ such that $g'_0 f_0 = g'_1 f_1$ and $g' g'_i = h_i$, there is an unique mediating arrow $z$ such that $z g_i = g'_i$ and $g' z = g$. ■

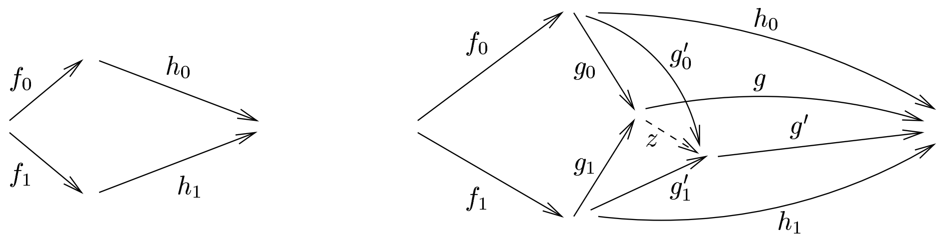



Figure 15: A relative pushout $g_0, g_1, g$

---

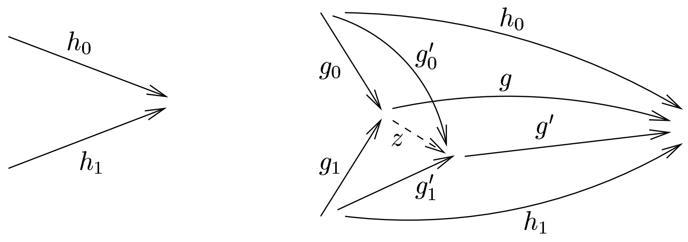



Figure 16: A slice sum $g_0, g_1, g$

---

A closely related concept is the following:

**Definition 47 (slice sum)** In any category $\mathbf{C}$, let $h_0, h_1$ be a pair of arrows with a common codomain (see Figure 16). A triple $g_0, g_1, g$ for which $gg_i = h_i$ $(i = 0, 1)$ is a *slice sum of* $h_0, h_1$ if, for any other triple $g_0', g_1', g'$ such that $gg_i' = h_i'$, there is an unique mediating arrow $z$ such that $zg_i = g_i'$ and $g'z = g$. ∎

**Remark** RPOs and slice sums are pushouts and coproducts in slice categories (the obvious dual notion to that of coslice, cf. Definition 28). We make no specific use of this observation.

**Nomenclature** If $h_0, h_1$ is a pair of arrows as in Figure 16, we call a triple $g_0, g_1, g$ for which $gg_i = h_i$ a *candidate (slice sum)* for $h_0, h_1$.

If $C$ is an object of $\mathbf{C}$, slice sums in the coslice category $C/\mathbf{C}$ correspond to RPOs in $\mathbf{C}$ for pairs of arrows with domain $C$:

**Proposition 48** *Let* $\mathbf{C}$ *be a category. Let* $C, C_0, C_1, C_2$ *be four objects of* $\mathbf{C}$*, and* $f_i : C \to C_i$*,* $h_i : C_i \to C_2$ $(i = 0, 1)$ *four arrows of* $\mathbf{C}$*. If we regard* $f_0$ *and* $f_1$ *as objects of* $C/\mathbf{C}$ *and* $h_i : f_i \to h_i f_i$ *as arrows of* $C/\mathbf{C}$*, respectively, the triple* $g_0, g_1, g$ *is a slice sum of* $h_0, h_1$ *iff it is an RPO in* $\mathbf{C}$ *for* $f_0, f_1$ *wrt* $h_0, h_1$*.*

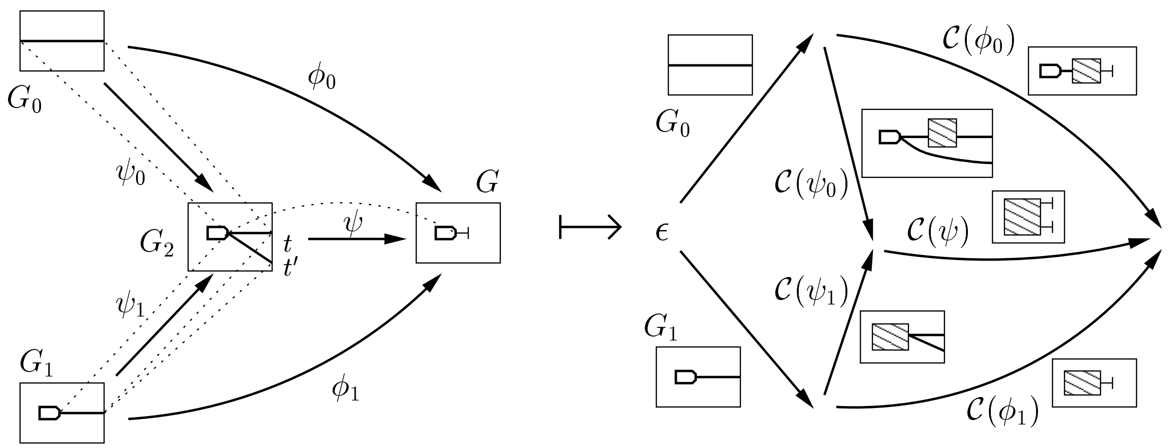



Figure 17: A slice sum in $\mathbf{GEmb}_0$ becomes a relative pushout in $\mathbf{PCxt}_0$

---

**Proof** The proof is a trivial consequence of the definitions of slice sum and RPO applied to the categories $C/\mathbf{C}$ and $\mathbf{C}$ respectively. ∎

This proposition relates RPOs in $\mathbf{PCxt}_0$ to slice sums in $(m, n, U)/\mathbf{PCxt}_0$. In particular we are interested in RPOs for pairs of contexts out of the empty profile $\epsilon$ (actions), so we focus on slice sums in $\epsilon/\mathbf{PCxt}_0$. As shown in the previous section, $\mathcal{C} : \mathbf{GEmb}_0 \to \epsilon/\mathbf{PCxt}_0$ is an isomorphism of categories, so preserves slice sums.

We shall use these facts in Corollary 57 to derive RPOs for $\mathbf{PCxt}_0$ from slice sums in $\mathbf{GEmb}_0$. We now illustrate this derivation for a simple example in Figure 17. The left diagram shows a slice sum $\psi_0, \psi_1, \psi$ for $\phi_0, \phi_1$ in $\mathbf{GEmb}_0$, which becomes an RPO in $\mathbf{PCxt}_0$ via $\mathcal{C}$. (Recall that an arrow $\mathcal{C}(\phi)$ of $\epsilon/\mathbf{PCxt}_0$ is also an arrow of $\mathbf{PCxt}_0$.) Note that $\mathcal{C}(\psi_0)\, G_0 = \mathcal{C}(\psi_1)\, G_1 = G_2$ and $\mathcal{C}(\psi)\, \mathcal{C}(\psi_i) = \mathcal{C}(\psi\, \psi_i) = \mathcal{C}(\phi)$, thus the proposed RPO commutes correctly.

The source and target maps of $\psi_0$, $\psi_1$ and $\psi$ are shown as dotted lines. For example the embedding of $G_0$'s single arc into $G_2$ is represented by the hole in the context $\mathcal{C}(\psi_0)$. Since $G$ has no targets, the target map $\psi^{\mathsf{T}}$ is empty; this corresponds to the discarding of the targets $t$ and $t'$, represented by the aborted arcs in $\mathcal{C}(\psi)$. Just as in the action $b$ of Figure 4 in Example 2 (arithmetic), the extra arc of $G_2$ connecting the control node to $t'$ may seem at first superfluous. But it is essential, since a competing candidate can have a target like $t'$, sourced by the control node, that is mapped back to $G_1$ but not to $G_0$. This distinguishes it from $t$, which is mapped to both. By E-TARGS there are no other possibilities (such as a target mapped to $G_0$ but not to $G_1$). Thus $G_2$ contains precisely the targets it needs to be "as good as" any candidate. The formal construction of $G_2$ is given later in Theorem 55.

Our task now is to establish slice sums in $\mathbf{GEmb}_0$. They are not always present! Figure 18 shows a case with no controls – only "wiring" – in which there is no slice sum. $G_0$ is a graph with a single input source $s_0$ and output target $t_0$, with $s_0 = src_0(t_0)$; $G_1$ is similar. $G$ has a single source but no target. $H$ and $H'$ are candidate slice sums – one with a single arc and one with two sources

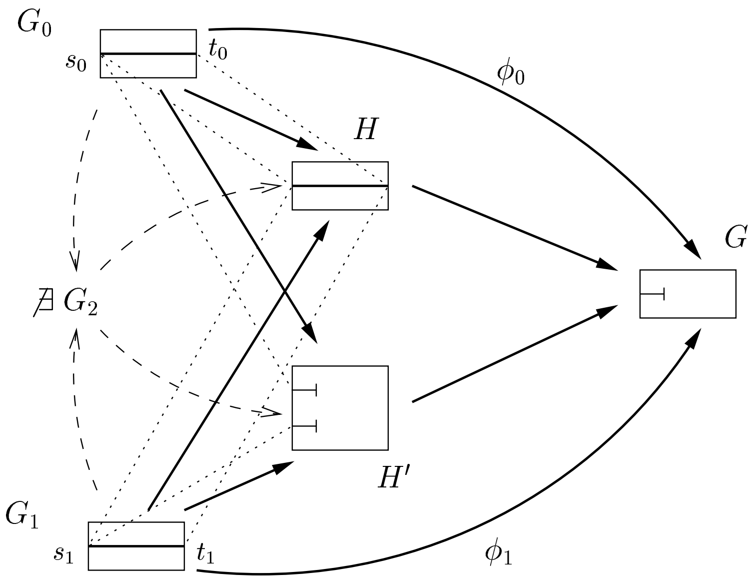



Figure 18: A case where no slice sum exists

and no arcs. The source and targets maps of the embeddings into $H$ and $H'$ are shown as dotted lines. These maps are not shown for embeddings into $G$ (they are simple: all sources go to the single source in $G$, and $G$ has no target to map). Now any slice sum $G_2$ must possess embeddings from $G_i$ and to $H$ and $H'$ as shown by dashed lines, making the diagram commute. But no such embeddings can exist. The reader may enjoy proving this.

However, slice sums exist for many pairs of embeddings. We define below a condition SLICE, which characterises exactly these pairs, i.e., is a necessary and sufficient condition for the existence of slice sums. Details of the definitions and proofs which we outline in the remainder of this section will appear in Leifer's forthcoming PhD Dissertation [16]; here we give the main definitions, and state the auxiliary results and main theorem with outlined proofs.

First we define a key equivalence on which SLICE depends.

**Definition 49 (source coalescing)** Let $\phi_i : G_i \to G$ $(i = 0, 1)$ be a pair of embeddings. The *source coalescing* of $\phi_0, \phi_1$ is the smallest equivalence relation $\equiv_\phi$ on $S_0 + S_1$ such that $src_0\phi_0^{\mathsf{T}}(t) \equiv_\phi src_1\phi_1^{\mathsf{T}}(t)$ for all $t \in \mathsf{Dom}\,\phi_0^{\mathsf{T}} \cap \mathsf{Dom}\,\phi_1^{\mathsf{T}}$. ∎

An important property of $\equiv_\phi$ is that it relates sources that must be equated in any candidate, as shown by the following proposition:

**Proposition 50** *Let $\psi_0, \psi_1, \psi$ be any candidate for $\phi_0, \phi_1$ and let $s_i$ be a source of $G_i$ for $i = 0, 1$. If $s_0 \equiv_\phi s_1$ then $\psi_0^{\mathsf{S}}(s_0) = \psi_1^{\mathsf{S}}(s_1)$.*

**Definition 51 (SLICE)** Let $\phi_i$ be as in the previous definition. The SLICE condition holds for

$\phi_0, \phi_1$ if for all $s_i \in S_i^{in}$ and $t_i \in T_i^{out}$ $(i = 0, 1)$, if $s_i = src_i(t_i)$ and $\phi_0^{\mathsf{S}}(s_0) = \phi_1^{\mathsf{S}}(s_1)$ then $s_0 \equiv_\phi s_1$. ■

The reader may like to check that SLICE fails for the pair $\phi_0, \phi_1$ in Figure 18.

The SLICE condition is implied by a striking property which is asymmetric between $G_0$ and $G_1$:

**Definition 52 (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. ■

**Proposition 53** *If $G_1$ is output-controlled, then any pair of embeddings $\phi_i : G_i \to G$ $(i = 0, 1)$ satisfies* SLICE.

Our interest in this fact arises because we are mainly concerned to find slice sums, or RPOs, when one of the graphs is a redex. In all cases we have met, the required reaction relation $\longrightarrow$ can be achieved with reaction rules whose redexes are output-controlled. The reason appears to be as follows. Let $(l, r)$ be a reaction rule whose redex $l$ in not output-controlled. Consider all arcs $src(t) = s$ in $l$ where $s$ is an input source and $t$ an output target. If the contractum $r$ possesses all these arcs, then it can be shown that the alternative rule $(l', r')$ with all such arcs removed yields exactly the same reactions; hence the rule can be replaced by $(l', r')$ for the purpose of deriving an LTS. If on the other hand $r$ does not contain such an arc then the rule $(l, r)$ appears to generate non-deterministic behaviour of a kind not normally required. So it appears that if the argument below showing the existence of slice sums can be generalised to richer action calculi (deep, open, etc.) then congruential LTSs can be derived when we need them since the requirement that redexes are output-controlled is tolerable.

We confine our argument to the category $\mathbf{GInc}_0$ of inclusions, returning to $\mathbf{GEmb}_0$ in Corollary 56. Let us fix a pair $\eta_i : G_1 \to G$ $(i = 0, 1)$ of inclusions. To show that SLICE is sufficient for $\eta_0, \eta_1$ to have a slice sum, we could give a direct construction. However, we also wish to show that SLICE is necessary, and therefore require a second candidate to argue that no slice sum exists when SLICE fails. To factor the work of constructing both candidates, we now give a general way to lift a triple of partial maps $\theta_0^{\mathsf{T}}, \theta_1^{\mathsf{T}}, \theta^{\mathsf{T}}$ to a candidate $\theta_0, \theta_1, \theta$. Only those triples that satisfy certain conditions (omitted here) can be so lifted; we call them *target scaffolds (for $\eta_0, \eta_1$)*.

We first show how to make a candidate from a target scaffold. Then we prove SLICE's sufficiency by exhibiting a particular target scaffold $\theta_0^{\mathsf{T}}, \theta_1^{\mathsf{T}}, \theta^{\mathsf{T}}$, lifting it to a candidate, and, subject to SLICE, proving that this candidate is a slice sum. Finally we prove SLICE's necessity by exhibiting a second target scaffold, $\hat{\theta}_0^{\mathsf{T}}, \hat{\theta}_1^{\mathsf{T}}, \hat{\theta}^{\mathsf{T}}$, lifting it to a candidate, and showing that the negation of SLICE implies that no putative slice sum is better than both $\theta, \theta_0, \theta_1$ and $\hat{\theta}, \hat{\theta}_0, \hat{\theta}_1$.

**Lemma 54** *Every target scaffold for $\eta_0, \eta_1$ consisting of a set $T_2$ and a triple of partial maps $\zeta_0^{\mathsf{T}}, \zeta_1^{\mathsf{T}}, \zeta^{\mathsf{T}}$, where $\zeta^{\mathsf{T}} : T \rightharpoonup T_2$ and $\zeta_i^{\mathsf{T}} : T_2 \rightharpoonup T_i$, yields a graph $G_2$ and a candidate $\zeta_0, \zeta_1, \zeta$ with $\zeta : G \to G_2$ and $\zeta_i : G_2 \to G_i$.*

**Proof** (sketch) We find it convenient to express our construction of $G_2$ up to bijection of its source and target sets, rather than in the exact form of Definition 2. Let $G_2 \stackrel{\text{def}}{=} (V_2, ctr_2, src_2) : (m_2, n_2)$ where the parts are defined as follows. First we consider the vertices and targets of $G_2$:

$$\begin{array}{lcll} V_2 & \stackrel{\text{def}}{=} & V_0 \cup V_1 & \\ ctr_2(v) & \stackrel{\text{def}}{=} & ctr(v) & \text{for } v \in V_2 \\ arg_2(v,k) & \stackrel{\text{def}}{=} & \zeta^{\mathsf{T}} arg(v,k) & \text{for } v \in V_2 \\ n_2 & \stackrel{\text{def}}{=} & |T_2| - |\mathsf{Im}\, arg_2| & . \end{array}$$

Notice that $V_0$ and $V_1$ are not necessarily disjoint: since $\eta_0^{\mathsf{V}}$ and $\eta_1^{\mathsf{V}}$ are inclusions, $V_0 \cup V_1$ contains the vertices of $G_0$ and $G_1$ with appropriate overlap.

Now we consider the sources and source maps. Take $\doteq$ to be the least equivalence relation on $S_0 + S_1$ satisfying the following conditions:

$$\begin{array}{rcll} s_0 & \doteq & s_1 & \text{for } s_i \in S_i^{bind},\ s_{1-i} \in S_{1-i}, \text{ and } \eta_0^{\mathsf{S}}(s_0) = \eta_1^{\mathsf{S}}(s_1) \\ src_0 \zeta_0^{\mathsf{T}}(t) & \doteq & src_1 \zeta_1^{\mathsf{T}}(t) & \text{for } t \in \mathsf{Dom}\, \zeta_0^{\mathsf{T}} \cap \mathsf{Dom}\, \zeta_1^{\mathsf{T}} \quad . \end{array}$$

Let $[\,]_{\doteq} : S_0 + S_1 \to (S_0 + S_1)/\doteq$ map sources to their equivalence classes. Then, let:

$$\begin{array}{lcll} S_2 & \stackrel{\text{def}}{=} & (S_0 + S_1)/\doteq & \\ bind_2(v,k) & \stackrel{\text{def}}{=} & [bind_i(v,k)]_{\doteq} & \text{if } v \in V_i \\ m_2 & \stackrel{\text{def}}{=} & |S_2| - |\mathsf{Im}\, bind_2| & \\ src_2(t) & \stackrel{\text{def}}{=} & [src_i \zeta_i^{\mathsf{T}}(t)]_{\doteq} & \text{if } t \in \mathsf{Dom}\, \zeta_i^{\mathsf{T}} \\ \zeta_i^{\mathsf{S}}(s) & \stackrel{\text{def}}{=} & [s]_{\doteq} & \text{for } s \in S_i \\ \zeta^{\mathsf{S}}([s]_{\doteq}) & \stackrel{\text{def}}{=} & \eta_i^{\mathsf{S}}(s) & \text{for } s \in S_i \quad . \end{array}$$

From the conditions defining target scaffolds it can be verified that $G_2$ is a graph, that $\zeta_0, \zeta_1, \zeta$ are all well-defined inclusion embeddings, and that together they form a candidate for $\eta_0, \eta_1$. ∎

We now come to the main applied theorem of the paper.

**Theorem 55** *A pair of inclusion embeddings $\eta_i : G_i \to G$ ($i = 0, 1$) satisfies* SLICE *if and only if there exists a triple $\theta_0, \theta_1, \theta$ of inclusion embeddings forming a slice sum for $\eta_0, \eta_1$ in* **GInc**$_0$.

**Proof** (outline: for full details see Leifer [16]).

**Sufficiency:** We exhibit a target scaffold consisting of a set $T_2$ and a triple $\theta_0^{\mathsf{T}}, \theta_1^{\mathsf{T}}, \theta^{\mathsf{T}}$, which we lift to a slice sum. Let:

$$\begin{array}{lcl}
T_2 & \stackrel{\text{def}}{=} & A + O_0 + O_{0,1} + O_1 \\
A & \stackrel{\text{def}}{=} & \{arg(v,k) \mid v \in V_0 \cup V_1\} \\
O_{0,1} & \stackrel{\text{def}}{=} & \{(t_0,t_1) \in T_0^{out} \times T_1^{out} \mid \eta_0^{\mathsf{S}} src_0(t_0) = \eta_1^{\mathsf{S}} src_1(t_1)\} \\
O_i & \stackrel{\text{def}}{=} & \{t_i \in T_i^{out} \mid \eta_i^{\mathsf{S}}\, src_i(t_i) \notin \eta_{1-i}^{\mathsf{S}}(S_{1-i}^{bind})\} \\
\theta_i^{\mathsf{T}}(t) & \stackrel{\text{def}}{=} & \begin{cases} \eta_i^{\mathsf{T}}(t) & \text{if } t \in A \\ t & \text{if } t \in O_i \\ t_i & \text{if } t = (t_0,t_1) \in O_{0,1} \end{cases} \\
\theta^{\mathsf{T}}(t) & \stackrel{\text{def}}{=} & \begin{cases} t & \text{if } t \in A \\ \eta_i^{\mathsf{T}}(t) & \text{if } t \notin A \text{ and } t \in \mathsf{Dom}\,\eta_i^{\mathsf{T}} \setminus \mathsf{Dom}\,\eta_{1-i}^{\mathsf{T}} \\ (\eta_0^{\mathsf{T}}(t), \eta_1^{\mathsf{T}}(t)) & \text{if } t \notin A \text{ and } t \in \mathsf{Dom}\,\eta_0^{\mathsf{T}} \cap \mathsf{Dom}\,\eta_1^{\mathsf{T}}\ . \end{cases}
\end{array}$$

It can be verified that $\theta_0^{\mathsf{T}}, \theta_1^{\mathsf{T}}, \theta^{\mathsf{T}}$ is a target scaffold, so can be lifted to a candidate $\theta_0, \theta_1, \theta$. We also claim that SLICE implies that $\theta_0, \theta_1, \theta$ is a slice sum.

At this point the reader may like to refer to the remark following the proof, which illustrates this construction with reference to examples.

**Necessity:** Construct a second candidate for $\eta_0, \eta_1$ as follows. Let

$$\begin{array}{lcl}
\hat{T}_2 & \stackrel{\text{def}}{=} & \mathsf{Dom}\,\eta_0^{\mathsf{T}} \cup \mathsf{Dom}\,\eta_1^{\mathsf{T}} \\
\hat{\theta}^{\mathsf{T}} : T \rightharpoonup \hat{T} & \stackrel{\text{def}}{=} & t \mapsto t \qquad \text{for } t \in \hat{T} \\
\hat{\theta}_i^{\mathsf{T}} : \hat{T} \rightharpoonup T_i & \stackrel{\text{def}}{=} & \eta_i^{\mathsf{T}} \restriction \hat{T}\ .
\end{array}$$

Then $\hat{\theta}_0^{\mathsf{T}}, \hat{\theta}_1^{\mathsf{T}}, \hat{\theta}^{\mathsf{T}}$ is a target scaffold, hence can be extended to a candidate $\hat{\theta}_0, \hat{\theta}_1, \hat{\theta}$. This candidate has a useful property which we state without proof: if $s_i$ is an input source of $G_i$ for $i = 0,1$ then $s_0 \not\equiv_\eta s_1$ implies $\hat{\theta}_0^{\mathsf{S}}(s_0) \neq \hat{\theta}_1^{\mathsf{S}}(s_1)$.

Suppose that the SLICE condition does not hold, i.e. there are $s_i \in S_i^{in}$ and $t_i \in T_i^{out}$ such that $src_i(t_i) = s_i$, $\eta_0^{\mathsf{S}}(s_0) = \eta_1^{\mathsf{S}}(s_1)$, and $s_0 \not\equiv_\eta s_1$. Suppose for contradiction that there is a graph $G_3$ and a slice sum triple $\zeta_0, \zeta_1, \zeta$ such that $\zeta : G_3 \to G$ and $\zeta_i : G_i \to G_3$ for $i = 0,1$. By the definition of slice sum, there exists mediating inclusions $\chi : G_3 \to G_2$ and $\xi : G_3 \to \hat{G}_2$.

As observed above, $s_0 \not\equiv_\eta s_1$ implies $\hat{\theta}_0^{\mathsf{S}}(s_0) \neq \hat{\theta}_1^{\mathsf{S}}(s_1)$, hence $\xi^{\mathsf{S}}\zeta_0^{\mathsf{S}}(s_0) \neq \xi^{\mathsf{S}}\zeta_1^{\mathsf{S}}(s_1)$, hence $\zeta_0^{\mathsf{S}}(s_0) \neq \zeta_1^{\mathsf{S}}(s_1)$. However, $(t_0,t_1) \in O_{0,1}$, so $\zeta_i^{\mathsf{S}}(s_i) = \zeta_i^{\mathsf{S}} src_i(t_i) = \zeta_i^{\mathsf{S}} src_i \eta_i^{\mathsf{T}}(t_0,t_1) = \zeta_i^{\mathsf{S}} src_i \zeta_i^{\mathsf{T}} \chi^{\mathsf{T}}(t_0,t_1) = src_3 \chi^{\mathsf{T}}(t_0,t_1)$ by I-SRC for $\zeta_i$; so $\zeta_0^{\mathsf{S}}(s_0) = \zeta_1^{\mathsf{S}}(s_1)$, a contradiction. ∎

**Remark** The construction of $T_2$ in the first part of this proof appears complex, so the reader may wish to pattern-match it against the target sets of the RPO examples given in Section 3. As we will see shortly (Corollary 57) there is a tight correspondence: $T_2$ is exactly the target set of an RPO action, as we now illustrate.

*Example 2 (arithmetic)*: Consider the action graphs $G_0 = a$ and $G_1 = l_1$ in Figure 4, embedded in $b'$. The targets of $G_2 = b$, the RPO graph, correspond exactly to $T_2$; in particular, the extra arc in $b$ connecting S to the top output target is generated by the set $O_1$ in the definition of $T_2$. Similarly, $t' \in O_1$ in Figure 17.

*Example 3 (wiring)*: Consider the identical action graphs $G_0 = a$ and $G_1 = b$ in Figure 5, embedded in an identical graph. The targets $\{(t_0, t_2), \ldots, (t_1, t_3)\}$ of $K$ in $G = Ca = Db$ are generated by the set $O_{0,1}$ (a subset of the product of the targets from $G_0$ and $G_1$) in the definition of $T_2$.

**Corollary 56** *The embeddings $\phi_i : G_i \to G$ $(i = 0, 1)$ satisfy property* SLICE *if and only if there exist a graph $G_2$ and embeddings $\psi_i : G_i \to G_2$ $(i = 0, 1)$ and $\psi : G_2 \to G$ forming a slice sum for $\phi_0, \phi_1$ in* $\mathbf{GEmb}_0$.

**Proof** (outline)
($\Longrightarrow$) By Proposition 15 $\phi_i = \eta_i \rho_i$ $(i = 0, 1)$, where $\rho_i$ is an isomorphism and $\eta_i$ an inclusion. The pair $\eta_0, \eta_1$ has property SLICE, since isomorphism does not disturb it; therefore, by Theorem 55, in $\mathbf{GInc}_0$ there is a slice sum $\theta_0, \theta_1, \theta$ for $\eta_0, \eta_1$. It is then not hard to show that $\theta_0\rho_0, \theta_1\rho_1, \theta$ is a slice sum for $\phi_0, \phi_1$ in $\mathbf{GEmb}_0$; one needs the fact that if $\chi'$ and $\chi'\chi$ are inclusions then so is $\chi$.

($\Longleftarrow$) The argument can be essentially reversed. Let $\psi_0, \psi_1, \psi$ be a slice sum for $\phi_0, \phi_1$ in $\mathbf{GEmb}_0$. Let $\psi = \theta\tau$, with $\theta$ an inclusion and $\tau$ an iso. Then setting $\theta_i \stackrel{\text{def}}{=} \tau\psi_i\rho_i^{-1}$, $(i = 0, 1)$, we can show that $\theta_0, \theta_1, \theta$ is a slice sum for the pair $\eta_0, \eta_1$, which therefore has property SLICE. Hence also the pair $\phi_0, \phi_1$ has property SLICE. ■

We are finally ready to deduce the existence of RPOs in the category $\mathbf{PCxt}_0$ of contexts, which is the goal set for the paper in Section 4. Recall that in $\mathbf{PCxt}_0$ a context $C : \epsilon \to (m, n, V)$, whose domain is the empty profile, is a pair $(G, \emptyset_V)$ consisting of a graph $G$ with arity $(m, n)$ and the empty renaming into $V$; we abbreviate this pair to $G$.

**Corollary 57** *Let $C_0G_0 = C_1G_1 = G$ in* $\mathbf{PCxt}_0$*, where $G_1$ is output-controlled. Then there exists an RPO for $G_0, G_1$ wrt $C_0, C_1$.*

**Proof** Let $\eta_i = \mathcal{E}_{G_i,G}(C_i)$, for $i = 0, 1$. Since $G_1$ is output-controlled the pair $\eta_0, \eta_1$ satisfies property SLICE by Proposition 53, hence possesses a slice sum by Corollary 56. The isomorphism $\mathcal{C} : \mathbf{GEmb}_0 \to \epsilon/\mathbf{PCxt}_0$ transforms any such slice sum into a slice sum of $C_0 : \mathcal{C}(G_0) \to \mathcal{C}(G)$ and $C_1 : \mathcal{C}(G_1) \to \mathcal{C}(G)$, and hence, by Proposition 48, into an RPO in $\mathbf{PCxt}_0$ for $G_0, G_1$ with respect to $C_0, C_1$. ■

# 11 Related and future work

**Related studies** Many connections with existing and ongoing work can be made. As mentioned in the introduction, the algebraic theory of graphs originated by Ehrig [5] employs double pushouts; these are a way to describe the occurrence of a subgraph – especially a redex – in a graph. In our theory, subgraph occurrences are handled by embeddings and contexts; the nature of our graphs (with forked wiring) seems to require a specific approach. But it would be useful to examine more closely how the embeddings relate to the double pushout construction. (To avoid confusion, we should emphasize that our *relative* pushouts play quite a different rôle.)

There are other graphical models of interaction, with different ways of wiring up components. We have already mentioned Lafont's interaction nets and Hasegawa's sharing graphs in Section 3; closely related are the equational term graphs of Klop and Ariola [1], and we believe the current work should apply quite smoothly to them. Models which allow many-to-many arcs between nodes are the fusion systems of Gardner and Wischik [8], and the process structures of Honda [11]. It is less clear how our theory will apply here; for example, the embeddings axioms may turn out to be simpler or more complex, and slice sums may turn out to exist more often (even always) or less often. So our theory offers a good means for comparison of rival models.

Among other studies of behavioural congruence, the work on structured operational semantics (SOS) and bisimilarity should be mentioned. About a decade ago much work was done to determine conditions on an LTS defined by SOS which ensure that the associated bisimilarity equivalence is a congruence. This is well reported by Groote and Vaandrager [9], and has recently been understood in categorical terms by Turi and Plotkin [24]. In that work the LTS is *given* inductively on the structure of terms. Our work is complementary; we *construct* an LTS. We are therefore offered the challenge to see whether our LTS can be *presented* inductively.

Finally, since our work yields congruential bisimilarities using categorical notions, we would like to see a connection with the categorical derivation of bisimilarity from open maps, following Joyal, Nielsen and Winskel [14].

**Future work** The work reported here must be developed in several directions, in order to demonstrate that it covers calculi which are used in practice. The present paper is a pilot study, applying the concept of RPO to yield labelled transition systems in a case which is tractable but retains several aspects of practical calculi. We excluded nested actions and free names; but we included much of the complexity of wiring (copy, discard and reflexion) which are important in such calculi.

The extension to include free names is not hard; the notion of context has to be generalised to allow them to bind names, but for example the central theorem on existence of RPOs remains essentially unchanged. However, free names do not give us much more in practice unless we also add nested actions. Work is in progress in this direction; the problems appear not to be basic, but the details are more complicated. The third main constraint imposed in this paper was the limitation on reflexivity; we allowed only "loose" contexts (see Section 7). As we observed, this is no limitation for some applications, but it obscures the theory; we shall not fully understand embeddings and contexts until we have either removed this limitation or found it necessary.

Thus, having shown in this pilot study that the RPO method is plausible, we are encouraged to seek new ideas which will keep the theory tractable when the present limitations are removed. We intend next to study *linear action graphs* – those with no copy and discard – but with full nesting and reflexion; linearity greatly simplifies (for example) the concept of embedding and hence the existence of RPOs, making it easier to concentrate upon other aspects. It may then be possible to retrieve sharing (copy and discard) by some sort of quotient. As part of this exercise we shall also consider multi-hole contexts, which Sewell [23] has shown to play an important part in deriving LTSs with simple labels. A prime example is term rewriting, the simplest case of a reactive system with nested components.

Finally we must see whether, for existing calculi, our method recovers known LTSs and congruences or discovers interesting new ones. In many cases, e.g. the $\pi$-calculus, we must wait until the theory is worked out for nested actions. But there are examples which need no nesting; one such is the action calculus for Petri nets presented in [20].

An ultimate goal for the theory of mobile interactive systems is to find a mathematical model at a level of abstraction comparable with that provided by domain theory for functional computation. This is a distant goal; we believe that a thorough structural understanding of behavioural congruence is a prerequisite which is more attainable. The present work is a step in that direction.

**Acknowledgments** The authors thank Philippa Gardner, Georges Gonthier, Martin Hyland, Ole Jensen, Jean-Jacques Lévy, Andrew Pitts, Peter Sewell and Lucian Wischik for valuable discussions and comments. Cattani was supported by EPSRC Research Grant GR/L62290, and Leifer by an NSF Graduate Research Fellowship and a Trinity College Senior Rouse Ball Studentship.

# References

[1] Ariola, Z.M. and Klop, J.W., Equational term graph rewriting. Fundamentae Informaticae, 26(3,4), pp. 207–240, 1996.

[2] Berry, G. and Boudol, G., The chemical abstract machine. Theor. Comp. Sci. 96, pp. 217–248, 1992.

[3] Cardelli, L. and Gordon, A.D., Mobile ambients. Foundations of System Specification and Computational Structures, Lecture Notes in Comp. Sci. 1378, pp. 140–155, Springer-Verlag, 1998.

[4] Corradini, A. and Montanari, U., An algebra of graphs and graph rewriting. Proc. Fourth Biennial Conf. on Category Theory and Computer Science, Lecture Notes in Comp. Sci. 530, pp. 236–260, Springer-Verlag, 1991.

[5] Ehrig., H., Introduction to the algebraic theory of graph grammars. Proc. first international Workshop on Graph Grammars and their application to Computer Science and Biology, Lecture Notes in Comp. Sci. 73, pp. 1–69, Springer-Verlag, 1979.

[6] Fournet, C. and Gonthier, G., A hierarchy of equivalences for asynchronous calculi. Proc. Int. Conf. on Automata, Languages and Programming, pp. 844–855, Springer-Verlag, 1998.

[7] Gardner, P., Closed action calculi. Theor. Comp. Sci. 228(1,2), pp. 77–103, 1999.

[8] Gardner P. and Wischik, L., Explicit fusions, Proc. MFCS, 2000.

[9] Groote, J.F. and Vaandrager, F.W., Structural operational semantics and bisimulation as a congruence. Information and Computation 100(2), pp. 202–260, 1992.

[10] Hasegawa, M., Models of sharing graphs (a categorical semantics of let and letrec). Distinguished Dissertation Series, Springer-Verlag, 1999.

[11] Honda, K., Process structures. Proc. TPPP '94, Lecture Notes in Comp. Sci. 907, pp. 25–44, Springer-Verlag, 1995.

[12] Honda, K. and Tokoro, M., An object calculus for asynchronous communication. Proc. ECOOP 1991, ed. P.America, Lecture Notes in Comp. Sci. 512, pp. 133–147, Springer-Verlag, 1991.

[13] Joyal, A., Street, R. and Verity, D., Traced monoidal categories and models of cyclic lambda calculi. TLCA '97, Lecture Notes in Comp. Sci. , Springer-Verlag 1210, pp. 196–213, 1997.

[14] Joyal, A., Nielsen, M. and Winskel, G., Bisimulation from open maps. Information and Computation 127(2), pp. 164–185, 1996.

[15] Lafont, Y., Interaction nets. Proc. 17th ACM Symposium on Principles of Programming Languages, pp. 95–108, 1990.

[16] Leifer. J.J., forthcoming PhD dissertation, University of Cambridge, 2000.

[17] Leifer, J.J. and Milner, R., Deriving bisimulation congruences for reactive systems. Proc. CONCUR 2000. Available from: http://www.cl.cam.ac.uk/∼jjl21/ .

[18] Mac Lane, S., Categories for the Working Mathematician. Volume 5 of Graduate Texts in Mathematics, Springer-Verlag, 1971.

[19] Milner, R., Flowgraphs and Flow Algebras. Journal of ACM, 26,4, pp. 794–818, 1979.

[20] Milner, R., Calculi for interaction. Acta Informatica 33(8), pp. 707–737, 1996.

[21] Milner, R., Parrow, J. and Walker, D., A calculus of mobile processes, Parts 1 and 2. Information and Computation 100(1), pp. 1–77, 1992.

[22] Milner, R. and Sangiorgi, D., Barbed bisimulation. Proc. 19th Int. Conf. on Automata, Languages and Programming, Lecture Notes in Comp. Sci. 623, pp. 685–695, Springer-Verlag, 1992.

[23] Sewell, P., From rewrite rules to bisimulation congruences. Proc. CONCUR'98, Lecture Notes in Comp. Sci. 1466, pp. 269–284, Springer-Verlag, 1998. [Revised version to appear in a special issue of Theor. Comp. Sci.]

[24] Turi, D. and Plotkin, G., Towards a mathematical operational semantics. Proc. LICS'97, IEEE Press, pp. 280–291, 1997.

## APPENDIX

This Appendix contains the full proofs of Propositions 22, 31, 34 and 41.

**Proposition 22** *For any insertion $A$, $A \circledcirc \mathsf{id} = \mathsf{id}' \circledcirc A = A$, where* $\mathsf{id}$ *and* $\mathsf{id}'$ *are identities on the domain and codomain of $A$. Also composition of insertions is associative, i.e. for appropriately matching profiles $A_2 \circledcirc (A_1 \circledcirc A_0) = (A_2 \circledcirc A_1) \circledcirc A_0$.*

**Proof** The identity properties are easy to establish. For the associativity of composition, let $A_i = (V_i, ctr_i, src_i) : p_i \to p_{i+1}$, with $p_i = (m_i, n_i, U_i)$, for $i = 0, 1, 2$. Using L and R to stand for left- and right-association, let

$$\begin{array}{rcl} A_L \stackrel{\text{def}}{=} (A_2 \circledcirc A_1) \circledcirc A_0 & = & (V_L, ctr_L, src_L) \\ A_R \stackrel{\text{def}}{=} A_2 \circledcirc (A_1 \circledcirc A_0) & = & (V_R, ctr_R, src_R) \ . \end{array}$$

Then it is easy to verify that

$$\begin{array}{rcl} V_L = V_R & = & V_2 \uplus V_1 \uplus V_0 \\ ctr_L = ctr_R & = & ctr_2 \uplus ctr_1 \uplus ctr_0 \ . \end{array}$$

Hence $A_L$ and $A_R$ have the same arity functions $ar : V \to \mathbb{N}^2$ etc, and

$$\begin{array}{rcccl} S_L = S_R & = & \sum_{v \in V_2 \uplus V_1 \uplus V_0} [arout(v)] + [m_3] + [n_0] & = & (S_2^{bind} \uplus S_1^{bind} \uplus S_0^{bind}) \uplus S_2^{in} \uplus S_0^{up} \\ T_L = T_R & = & \sum_{v \in V_2 \uplus V_1 \uplus V_0} [arin(v)] + [n_3] + [m_0] & = & (T_2^{arg} \uplus T_1^{arg} \uplus T_0^{arg}) \uplus T_2^{out} \uplus T_0^{down} \ . \end{array}$$

It remains to prove that $src_L = src_R$.

As a further abbreviation, define $A_\ell \stackrel{\text{def}}{=} A_2 \circledcirc A_1$ and $A_r \stackrel{\text{def}}{=} A_1 \circledcirc A_0$, so that

$$A_L = A_\ell \circledcirc A_0 \ \text{ and } \ A_R = A_2 \circledcirc A_r \ .$$

So there are four source functions, each with two auxiliary functions:

For $A_r \stackrel{\text{def}}{=} A_1 \circledcirc A_0$:

$$\begin{array}{rcl} src_r(t) & = & \begin{cases} \sigma_{r1} src_1(t) & \text{if } t \in T_1^{arg} \uplus T_1^{out} \\ \sigma_{r0} src_0(t) & \text{if } t \in T_0^{arg} \uplus T_0^{down} \end{cases} \\ \sigma_{r1}(s) & = & \begin{cases} s & \text{if } s \in S_1^{bind} \uplus S_1^{in} \\ \sigma_{r0} src_0\, out_0(j) & \text{if } s = up_1(j) \in S_1^{up} \end{cases} \\ \sigma_{r0}(s) & = & \begin{cases} s & \text{if } s \in S_0^{bind} \uplus S_0^{up} \\ src_1\, down_1(i) & \text{if } s = in_0(i) \in S_0^{in} \ . \end{cases} \end{array}$$

For $A_\ell \stackrel{\text{def}}{=} A_2 \circledcirc A_1$:

$$\begin{array}{rcl} src_\ell(t) & = & \begin{cases} \sigma_{\ell 2} src_2(t) & \text{if } t \in T_2^{arg} \uplus T_2^{out} \\ \sigma_{\ell 1} src_1(t) & \text{if } t \in T_1^{arg} \uplus T_1^{down} \end{cases} \\ \sigma_{\ell 2}(s) & = & \begin{cases} s & \text{if } s \in S_2^{bind} \uplus S_2^{in} \\ \sigma_{\ell 1} src_1\, out_1(j) & \text{if } s = up_2(j) \in S_2^{up} \end{cases} \\ \sigma_{\ell 1}(s) & = & \begin{cases} s & \text{if } s \in S_1^{bind} \uplus S_1^{up} \\ src_2\, down_2(i) & \text{if } s = in_1(i) \in S_1^{in} \ . \end{cases} \end{array}$$

For $A_R \stackrel{\text{def}}{=} A_2 \oplus A_r$:

$$
\begin{aligned}
src_R(t) &= \begin{cases} \sigma_{R2}\, src_2(t) & \text{if } t \in T_2^{arg} \uplus T_2^{out} \\ \sigma_{Rr}\, src_r(t) & \text{if } t \in T_1^{arg} \uplus T_0^{arg} \uplus T_0^{down} \end{cases} \\
\sigma_{R2}(s) &= \begin{cases} s & \text{if } s \in S_2^{bind} \uplus S_2^{in} \\ \sigma_{Rr}\, src_r\, out_1(j) & \text{if } s = up_2(j) \in S_2^{up} \end{cases} \\
\sigma_{Rr}(s) &= \begin{cases} s & \text{if } s \in S_1^{bind} \uplus S_0^{bind} \uplus S_0^{up} \\ src_2\, down_2(i) & \text{if } s = in_1(i) \in S_1^{in}\ . \end{cases}
\end{aligned}
$$

For $A_L \stackrel{\text{def}}{=} A_\ell \oplus A_0$:

$$
\begin{aligned}
src_L(t) &= \begin{cases} \sigma_{L\ell}\, src_\ell(t) & \text{if } t \in T_2^{arg} \uplus T_1^{arg} \uplus T_2^{out} \\ \sigma_{L0}\, src_0(t) & \text{if } t \in T_0^{arg} \uplus T_0^{down} \end{cases} \\
\sigma_{L\ell}(s) &= \begin{cases} s & \text{if } s \in S_2^{bind} \uplus S_1^{bind} \uplus S_2^{in} \\ \sigma_{L0}\, src_0\, out_1(j) & \text{if } s = up_1(j) \in S_1^{up} \end{cases} \\
\sigma_{L0}(s) &= \begin{cases} s & \text{if } s \in S_0^{bind} \uplus S_0^{up} \\ src_\ell\, down_1(i) & \text{if } s = in_0(i) \in S_0^{in}\ . \end{cases}
\end{aligned}
$$

We now prove that $src_R(t) = src_L(t)$ by considering the cases $t \in T_0^{arg} \uplus T_0^{down}$, $t \in T_1^{arg}$, and $t \in T_2^{out} \uplus T_2^{arg}$. (The subsidiary case analysis is somewhat detailed due to the wiring connection which each composition can make.)

**Case 1** $t \in T_0^{arg} \uplus T_0^{down}$. Then $src_R(t) = \sigma_{Rr}\, src_r(t) = \sigma_{Rr}\sigma_{r0}\, src_0(t)$ and $src_L(t) = \sigma_{L0}\, src_0(t)$. Let $s_0 = src_0(t)$.

- **1.1** $s_0 \in S_0^{bind} \uplus S_0^{up}$. Then $\sigma_{Rr}\sigma_{r0}(s_0) = \sigma_{Rr}(s_0) = s_0$, and $\sigma_{L0}(s_0) = s_0$.
- **1.2** $s_0 = in_0(i) \in S_0^{in}$. Then $\sigma_{Rr}\sigma_{r0}(s_0) = \sigma_{Rr}\, src_1\, down_1(i)$, and $\sigma_{L0}(s_0) = src_\ell\, down_1(i) = \sigma_{\ell 1}\, src_1\, down_1(i)$. Let $s_1 = src_1\, down_1(i)$.
  - **1.2.1** $s_1 \in S_1^{bind}$. Then $\sigma_{Rr}(s_1) = s_1$ and $\sigma_{\ell 1}(s_1) = s_1$.
  - **1.2.2** $s_1 = in_1(i) \in S_1^{in}$. Then $\sigma_{Rr}(s_1) = src_2\, down_2(i)$ and $\sigma_{\ell 1}(s_1) = src_2\, down_2(i)$.
  - **1.2.3** $s_1 \in S_1^{up}$. Impossible by LOOSE.

**Case 2** $t \in T_1^{arg}$. Then $src_R(t) = \sigma_{Rr}\, src_r(t) = \sigma_{Rr}\sigma_{r1}\, src_1(t)$ and $src_L(t) = \sigma_{L\ell}\, src_\ell(t) = \sigma_{L\ell}\sigma_{\ell 1}\, src_1(t)$. Let $s_1 = src_1(t)$.

- **2.1** $s_1 \in S_1^{bind}$. Then $\sigma_{Rr}\sigma_{r1}(s_1) = \sigma_{Rr}(s_1) = s_1$ and $\sigma_{L\ell}\sigma_{\ell 1}(s_1) = \sigma_{L\ell}(s_1) = s_1$ and
- **2.2** $s_1 = in_1(i) \in S_1^{in}$. Then $\sigma_{Rr}\sigma_{r1}(s_1) = \sigma_{Rr}(s_1) = src_2\, down_2(i)$ and $\sigma_{L\ell}\sigma_{\ell 1}(s_1) = \sigma_{L\ell}\, src_2\, down_2(i) = src_2\, down_2(i)$, since $src_2\, down_2(i) \in S_2^{in} \uplus S_2^{bind}$ by LOOSE.
- **2.3** $s_1 = up_1(i) \in S_1^{up}$. Then $\sigma_{Rr}\sigma_{r1}(s_1) = \sigma_{Rr}\sigma_{r0}\, src_0\, out_0(j)$ and $\sigma_{L\ell}\sigma_{\ell 1}(s_1) = \sigma_{L\ell}(s_1) = \sigma_{L0}\, src_0\, out_0(j)$. Let $s_0 = src_0\, out_0(j)$. Then the argument proceeds exactly as in the subcases of Case 1.

**Case 3** $t \in T_2^{arg} \uplus T_2^{out}$. Then $src_R(t) = \sigma_{R2}\, src_2(t)$ and $src_L(t) = \sigma_{L\ell}\, src_\ell(t) = \sigma_{L\ell}\sigma_{\ell 2}\, src_2(t)$. Let $s_2 = src_2(t)$.

**3.1** $s_2 \in S_2^{bind} \uplus S_2^{in}$. Then $\sigma_{R2}(s_2) = s_2$ and $\sigma_{L\ell}\sigma_{\ell 2}(s_2) = \sigma_{L\ell}(s_2) = s_2$.

**3.2** $s_2 = up_2(j) \in S_2^{up}$. Then $\sigma_{R2}(s_2) = \sigma_{Rr} src_r\, out_1(j) = \sigma_{Rr}\sigma_{r1} src_1\, out_1(j)$, and $\sigma_{L\ell}\sigma_{\ell 2}(s_2) = \sigma_{L\ell}\sigma_{\ell 1} src_1\, out_1(j)$. Let $s_1 = src_1\, out_1(j)$. Then the argument proceeds exactly as in the subcases of Case 2.

This concludes the proof that $src_R = src_L$, and hence that composition is associative in $\mathbf{PIns}_0$. ■

**Proposition 31** *For every two inclusion embeddings $G_0 \overset{\eta_1}{\to} G_1 \overset{\eta_2}{\to} G_2$, $\mathcal{A}(\eta_2\eta_1) = \mathcal{A}(\eta_2) \oplus \mathcal{A}(\eta_1)$. Moreover for every graph $G$, $\mathcal{A}(\mathsf{id}_G) = \mathsf{id}_G$ and $\mathcal{A}(\mathbf{0}_G) = G$.*

**Proof** Consider two inclusion embeddings

$$\eta_1 : G_0 \to G_1 \text{ and } \eta_2 : G_1 \to G_2$$

where

$$G_k = (V_k, ctr_k, src_k) : (m_k, n_k)\ ,\ k = 0, 1, 2$$

with $V_k \subseteq V_{k+1}$. We must first set out the components of

$$\mathcal{A}(\eta_k) \overset{\text{def}}{=} A_k = (V_{Ak}, ctr_{Ak}, src_{Ak})\ , k = 1, 2\ .$$

From Definition 29 they are as follows:

$$\begin{array}{rcl}
V_{Ak} & = & V_k \setminus V_{k-1} \\
ctr_{Ak} & = & ctr_k \restriction V_{Ak}\ , \qquad \text{hence } ar_{Ak} = ar_k \restriction V_{Ak} \text{ etc} \\
S_{Ak} & = & \sum_{v \in V_{Ak}} [arout_k(v)] + [m_k] + [n_{k-1}] \ \ = (S_k^{bind} \setminus S_{k-1}^{bind}) \uplus S_k^{in} \uplus S_k^{up} \\
T_{Ak} & = & \sum_{v \in V_{Ak}} [arin_k(v)] + [n_k] + [m_{k-1}] \ \ = (T_k^{arg} \setminus T_{k-1}^{arg}) \uplus T_k^{out} \uplus T_k^{down}\ ,
\end{array}$$

and the source function of $A_k$ is

$$src_{Ak}(t) \ = \ \left\{ \begin{array}{ll}
\left\{ \begin{array}{ll} up_k(j) & \text{if } \eta_k^{\mathsf{T}}(t) \text{ defined, i.e. } \eta^{\mathsf{T}} t = out_{k-1}(j), j \in [n_{k-1}] \\ src_k(t) & \text{if } \eta_k^{\mathsf{T}}(t) \text{ undefined} \end{array} \right\} & \\
 & \text{if } t \in (T_k^{arg} \setminus T_{k-1}^{arg}) \uplus T_k^{out} \\
\eta^{\mathsf{S}} in_{k-1}(i) & \text{if } t = down_k(i) \in T_k^{down}, i \in [m_{k-1}]\ .
\end{array} \right.$$

Next, we consider

$$A \overset{\text{def}}{=} A_2 \oplus A_1 = (V_A, ctr_A, src_A) : \mathcal{A}(G_0) \to \mathcal{A}(G_2)\ .$$

We derive the components of $A$ from Definition 20:

$$\begin{array}{rcl}
V_A & = & V_{A2} \uplus V_{A1} \ = \ V_2 \setminus V_0 \\
ctr_A & = & ctr_2 \restriction V_A \text{ , thus determining } ar_A = ar_2 \restriction V_A \text{ etc} \\
S_A & = & \sum_{v \in V_A} [arout_A(v)] + [m_2] + [n_0] \ = \ (S_2^{bind} \setminus S_0^{bind}) \uplus S_2^{in} \uplus S_1^{up} \\
T_A & = & \sum_{v \in V_A} [arin_A(v)] + [n_2] + [m_0] \ = \ (T_2^{arg} \setminus T_0^{arg}) \uplus T_2^{out} \uplus T_1^{down}\ ,
\end{array}$$

and for the source function of $A$:

$$
\begin{array}{rcl}
src_2(t) & = & \begin{cases} \sigma_2 src_{A2}(t) & \text{if } t \in (T_2^{arg} \setminus T_1^{arg}) \uplus T_2^{out} \\ \sigma_1 src_{A1}(t) & \text{if } t \in (T_1^{arg} \setminus T_0^{arg}) \uplus T_1^{down} \end{cases} \\
\text{where} & & \\
\sigma_1(s) & = & \begin{cases} s & \text{if } s \in (S_1^{bind} \setminus S_0^{bind}) \uplus S_1^{up} \\ src_{A2} down_2(i) & \text{if } s = in_1(i) \in S_1^{in} \ ; \end{cases} \\
\text{and} & & \\
\sigma_2(s) & = & \begin{cases} s & \text{if } s \in (S_2^{bind} \setminus S_1^{bind}) \uplus S_2^{in} \\ \sigma_1 src_{A1} out_1(j) & \text{if } s = up_2(j) \in S_2^{up} \ . \end{cases}
\end{array}
$$

Finally, setting $\eta \stackrel{\text{def}}{=} \eta_2\eta_1$, we consider

$$B \stackrel{\text{def}}{=} \mathcal{A}(\eta) = (V_B, ctr_B, src_B) : \mathcal{A}(G_0) \to \mathcal{A}(G_2) \ .$$

We derive the components of $B$ from Definition 29:

$$
\begin{array}{rcl}
V_B & = & V_2 \setminus V_0 \\
ctr_B & = & ctr_2 \restriction V_B \text{ , thus determining } ar_B = ar_2 \restriction V_B \text{ etc} \\
S_B & = & \sum_{v \in V_B} [arout_B(v)] + [m_2] + [n_0] \ = \ (S_2^{bind} \setminus S_0^{bind}) \uplus S_2^{in} \uplus S_1^{up} \\
T_B & = & \sum_{v \in V_B} [arin_B(v)] + [n_2] + [m_0] \ = \ (T_2^{arg} \setminus T_0^{arg}) \uplus T_2^{out} \uplus T_1^{down} \ ,
\end{array}
$$

and for the source function of $B$:

$$
src_B(t) \ = \ \begin{cases} \left\{ \begin{cases} up_1(j) & \text{if } \eta^{\mathsf{T}}(t) \text{ defined, i.e. } \eta^{\mathsf{T}} t = out_0(j), j \in [n_0] \\ src_2(t) & \text{if } \eta^{\mathsf{T}}(t) \text{ undefined} \end{cases} \right\} \\ \qquad\qquad\qquad \text{if } t \in (T_2^{arg} \setminus T_0^{arg}) \uplus T_2^{out} \ ; \\ \eta^{\mathsf{S}} in_0(i) \qquad\quad \text{if } t = down_1(i) \in T_1^{down}, i \in [m_0] \ . \end{cases}
$$

Now we must prove that $A = B$. It is clear from the foregoing that $V_A = V_B$ and $ctr_A = ctr_B$. We now verify by a case analysis that indeed $src_A(t) = src_B(t)$ for all $t$. We take separately the cases that $t$ is in $T_1^{down}$, $T_1^{arg} \setminus T_0^{arg}$ and $(T_2^{arg} \setminus T_1^{arg}) \uplus T_2^{out}$.

**Case 1** $t = down_1(i) \in T_1^{down}$. Then $src_A(t) = \sigma_1 src_{A1}(t) = \sigma_1(s)$ and $src_B(t) = \eta_2^{\mathsf{S}}(s)$ where $s = \eta_1^{\mathsf{S}} in_0(i)$. There are three cases:

**1.1** $s \in S_1^{bind} \setminus S_0^{bind}$. Then $\eta_2^{\mathsf{S}}(s) = s$ by E-BIND, and $\sigma_1(s) = s$, hence result.

**1.2** $s \in S_1^{up}$: impossible by LOOSE.

**1.3** $s = in_1(i') \in S_1^{in}$. Then $\sigma_1(s) = src_{A2} down_2(i') = \eta_2^{\mathsf{S}}(s)$, hence result.

**Case 2** $t \in T_1^{arg} \setminus T_0^{arg}$. Then as $\eta_2^{\mathsf{T}}(t) = t$ by E-ARG, since $\eta_2$ is an inclusion, hence $\eta^{\mathsf{T}}(t) = \eta_1^{\mathsf{T}}\eta_2^{\mathsf{T}}(t) = \eta_1^{\mathsf{T}}(t)$.

**2.1** $\eta_1^{\mathsf{T}}(t)$ defined. Then $\eta_1^{\mathsf{T}}(t) = out_0(j), j \in [n_0]$. So $src_A(t) = \sigma_1 src_{A1}(t) = \sigma_1 up_1(j) = up_1(j) = src_B(t)$.

**2.2** $\eta_1^{\mathsf{T}}(t)$ undefined. Then $src_A(t) = \sigma_1 src_1(t)$ while $src_B(t) = src_2(t) = \eta_2^{\mathsf{S}} src_1 \eta_2^{\mathsf{T}}(t)$ by E-SRC, $= \eta_2^{\mathsf{S}} src_1(t)$. Let $s = src_1(t)$. By E-TARGS $s \notin S_0^{bind}$, since $\eta_1^{\mathsf{T}}(t)$ is undefined, so there are two cases:

$$
\begin{array}{ccc}
G_0 \xrightarrow{\ \eta\ } G_1 & & \mathcal{A}(G_0) \xrightarrow{\ \mathcal{A}(\eta)\ } \mathcal{A}(G_1) \\
\Big\downarrow \rho\boldsymbol{\cdot}(-) & \overset{\mathcal{A}}{\longmapsto} & \Big\downarrow \rho\boldsymbol{\cdot}(-) \\
\rho\boldsymbol{\cdot}G_0 \xrightarrow{\ \rho\boldsymbol{\cdot}\eta\ } \rho\boldsymbol{\cdot}G_1 & & \rho\boldsymbol{\cdot}\mathcal{A}(G_0) \xrightarrow{\ \rho\boldsymbol{\cdot}\mathcal{A}(\eta)\ } \rho\boldsymbol{\cdot}\mathcal{A}(G_1)
\end{array}
$$

Figure 19: How the functor $\mathcal{A}$ treats a renaming

---

**2.2.1** $s = in_1(i) \in S_1^{in}$, $i \in [m_1]$. Then $\sigma_1(s) = src_{A2}down_2(i) = \eta_2^{\mathsf{S}} in_1(i) = \eta_2^{\mathsf{S}}(s)$, hence result.

**2.2.2** $s \in S_1^{bind} \setminus S_0^{bind}$. Then $\sigma_1(s) = s$, and $\eta_2^{\mathsf{S}}(s) = s$ by E-BIND, hence result.

**Case 3** $t \in (T_2^{arg} \setminus T_1^{arg}) \uplus T_2^{out}$.

**3.1** $\eta^{\mathsf{T}}(t) = \eta_1^{\mathsf{T}}\eta_2^{\mathsf{T}}(t)$ defined. Then $\eta_2^{\mathsf{T}}(t) = out_1(j_1)$, $j_1 \in [n_1]$, and $src_A(t) = \sigma_2 up_2(j_1) = \sigma_1 src_{A1}\eta_2^{\mathsf{T}}(t)$. but $\eta_1^{\mathsf{T}}\eta_2^{\mathsf{T}}(t) = out_0(j_0)$ say, $j_0 \in [n_0]$; hence $src_A(t) = \sigma_1 up_1(j_0) = up_1(j_0)$. But $src_B(t) = up_1(j_0)$ also.

**3.2** $\eta_2^{\mathsf{T}}(t)$ defined, $\eta_1^{\mathsf{T}}\eta_2^{\mathsf{T}}(t)$ undefined. Then $src_A(t) = \sigma_1 src_{A1}\eta_2^{\mathsf{T}}(t)$ as above, $= \sigma_1 src_1\eta_2^{\mathsf{T}}(t)$; on the other hand $src_B(t) = src_2(t) = \eta_2^{\mathsf{S}} src_1 \eta_2^{\mathsf{T}}(t)$ by E-SRC. Now let $s = src_1\eta_2^{\mathsf{T}}(t)$; note that $s \notin S_0^{bind}$ since $s = src_{A1}\eta_2^{\mathsf{T}}(t)$.

**3.2.1** $s = in_1(i) \in S_1^{in}$, $i \in [m_1]$. Then $src_A(t) = \sigma_1(s) = src_{A2}down_2(i) = \eta_2^{\mathsf{S}} in_1(i) = \eta_2^{\mathsf{S}}(s) = src_B(t)$.

**3.2.2** $s \in S_1^{bind} \setminus S_0^{bind}$. Then $src_A(t) = \sigma_1(s) = s$. Also $src_B(t) = \eta_2^{\mathsf{S}}(s) = s$ by E-BIND.

**3.3** $\eta_2^{\mathsf{T}}(t)$ undefined, hence $\eta^{\mathsf{T}}(t)$ undefined. Then $src_A(t) = \sigma_2 src_{A2}(t) = \sigma_2 src_2(t)$, $= src_2(t)$ since $src_{A2}(t) = src_2(t) \in (S_2^{bind} \setminus S_1^{bind}) \uplus S_2^{in}$. Also $src_B(t) = src_2(t)$.

Now consider $\mathcal{A}(\mathsf{id}_G) = (V, ctr, src)$, where $G : (m, n)$. We find easily that $V = \emptyset$, $ctr = \emptyset_{\mathcal{K}}$, and

$$
\begin{array}{rcccl}
S_A & = & \emptyset + [m] + [n] & = & S^{in} \uplus S^{up} \\
T_A & = & \emptyset + [n] + [m] & = & T^{out} \uplus T^{down} \ ,
\end{array}
$$

and that

$$
src_A(t) = \begin{cases} up(j) & \text{if } t = out(j) \in T^{out}, j \in [n] \\ in(i) & \text{if } t = down(i) \in T^{down}, i \in [m] \end{cases}
$$

which is indeed an identity in $\mathbf{PIns}_0$.

Finally, using the definition of $\mathcal{A}$, it can be easily verified that $\mathcal{A}(\mathbf{0}_G) = G$. ∎

**Proposition 34** *The functor $\mathcal{A}$ preserves renaming, i.e. $\mathcal{A}(\rho\bullet\eta) = \rho\bullet\mathcal{A}(\eta)$ .*

**Proof** We must set out the data involved. It will help to refer to Figure 19. First, we have two graphs $G_k : (m_k, n_k)$, $k = 0,1$ and their renamed versions $\rho\bullet G_k : (m_k, n_k)$, $k = 0,1$. Their components are

$$\begin{array}{rcl} G_k &=& (V_k, ctr_k, src_k) \\ S_k &=& S_k^{bind} \uplus S_k^{in} \\ T_k &=& T_k^{arg} \uplus T_k^{out} \\ \rho\bullet G_k &=& (V_k', ctr_k', src_k') \\ V_k' &=& \rho V_k \\ ctr_k' \circ \rho &=& ctr_k \qquad\qquad S_k' \;=\; {S'}_k^{bind} \uplus S_k^{in} \\ src_k' &=& \rho_k^{\mathsf{S}} \circ src_k \circ \rho_k^{\mathsf{T}} \qquad T_k' \;=\; {T'}_k^{arg} \uplus T_k^{out}\ . \end{array}$$

In each case ($k = 0,1$) the node isomorphism $\rho_k : G_k \to \rho\bullet G_k$ has the following components:

$$\begin{array}{l} \rho_k = (\rho_k^{\mathsf{V}}, \rho_k^{\mathsf{S}}, \rho_k^{\mathsf{T}}) \\ \rho_k^{\mathsf{V}} \;\; : V_k \rightleftarrows V_k' \;\; = \rho \restriction V_k \\ \rho_k^{\mathsf{S}} \;\; : S_k \rightleftarrows S_k' \;\; : \begin{cases} bind_k(v,i) \mapsto bind_k'(\rho v, i) \\ in_k(i) \mapsto in_k(i) \end{cases} \\ \rho_k^{\mathsf{T}} \;\; : T_k' \rightleftarrows T_k \;\; : \begin{cases} arg_k'(\rho v, j) \mapsto arg_k(v,j) \\ out_k(j) \mapsto out_k(j)\ . \end{cases} \end{array}$$

Next, corresponding to the inclusion $\eta : G_0 \to G_1$ we have the context $\mathcal{A}(\eta) : \mathcal{A}(G_0) \to \mathcal{A}(G_1)$, and its transform (cf. Proposition 33) $\rho\bullet\mathcal{A}(\eta) : \rho\bullet\mathcal{A}(G_0) = \rho\bullet G_0 \to \rho\bullet G_1 = \rho\bullet\mathcal{A}(G_1)$, with components as follows:

$$\begin{array}{rcl} \mathcal{A}(\eta) &=& (V, ctr, src) \\ V &=& V_1 \setminus V_0 \\ ctr &=& ctr_1 \restriction V \\ S &=& (S_1^{bind} \setminus S_0^{bind}) \uplus S_1^{in} \uplus S^{up} \qquad (S^{up} \rightleftarrows T_0^{out}) \\ T &=& (T_1^{arg} \setminus T_0^{arg}) \uplus T_1^{out} \uplus T^{down} \qquad (T^{down} \rightleftarrows S_0^{in}) \end{array}$$

$$src(t) \;=\; \begin{cases} \left.\begin{cases} up(j) & \text{if } \eta^{\mathsf{T}}(t) \text{ defined, i.e. } = out_0(j) \\ src_1(t) & \text{if } \eta^{\mathsf{T}}(t) \text{ undefined} \end{cases}\right\} & \text{if } t \in (T_1^{arg} \setminus T_0^{arg}) \uplus T_1^{out} \\ \eta^{\mathsf{S}} in_0(i) \qquad\qquad \text{if } t = down_(i) \in T^{down}\ . \end{cases}$$

$$\begin{array}{rcl} \rho\bullet\mathcal{A}(\eta) &=& (V', ctr', src') \\ V' &=& \rho V_1 \setminus \rho V_0 \\ ctr' \circ \rho &=& ctr \\ S' &=& ({S'}_1^{bind} \setminus {S'}_0^{bind}) \uplus S_1^{in} \uplus S^{up} \\ T' &=& ({T'}_1^{arg} \setminus {T'}_0^{arg}) \uplus T_1^{out} \uplus T^{down} \\ src' &=& \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}}\ , \end{array}$$

where the bijections between sources and targets of the two contexts are as follows:

$$\begin{array}{l} \rho^{\mathsf{S}} \;\; : S \rightleftarrows S' \;\; : \begin{cases} bind_1(v,i) \mapsto bind_1'(\rho v, i) & (v \in V_1 \setminus V_0) \\ in_1(i) \mapsto in_1(i) \\ up(j) \mapsto up(j) \end{cases} \\ \rho_k^{\mathsf{T}} \;\; : T' \rightleftarrows T \;\; : \begin{cases} arg_1'(\rho v, j) \mapsto arg_1(v,j) & (v \in V_1 \setminus V_0) \\ out_1(j) \mapsto out_1(j) \\ down(i) \mapsto down(i)\ . \end{cases} \end{array}$$

These bijections correspond with the components of the node isomorphism $\rho_1 : G_1 \to \rho\text{▪}G_1$ in a simple way which we now record as a lemma:

**Lemma** *If* $s \in (S_1^{bind} \setminus S_0^{bind}) \uplus S_1^{in}$ *then* $\rho^{\mathsf{S}}(s) = \rho_1^{\mathsf{S}}(s)$.
*If* $t \in ({T'}_1^{arg} \setminus {T'}_0^{arg}) \uplus T_1^{out}$ *then* $\rho^{\mathsf{T}}(t) = \rho_1^{\mathsf{T}}(t)$.

The last required data is the inclusion $\theta \stackrel{\text{def}}{=} \rho\text{▪}\eta : G_0 \to G_1$. We recall that $\theta = \rho_1 \eta {\rho_0}^{-1}$. its image under $\mathcal{A}$ has the following components:

$$\begin{array}{rcl}
\mathcal{A}(\rho\text{▪}\eta) & = & (V'', ctr'', src'') \ : \\
V'' & = & \rho V_1 \setminus \rho V_0 \\
ctr'' & = & ctr'_1 \restriction V'' \\
S'' & = & ({S'}_1^{bind} \setminus {S'}_0^{bind}) \uplus S_1^{in} \uplus S^{up} \\
T'' & = & ({T'}_1^{arg} \setminus {T'}_0^{arg}) \uplus T_1^{out} \uplus T^{down}
\end{array}$$

$$src''(t) \ = \ \begin{cases} \left.\begin{cases} up(j) & \text{if } \theta^{\mathsf{T}}(t) \text{ defined, i.e. } = out_0(j) \\ src'_1(t) & \text{if } \theta^{\mathsf{T}}(t) \text{ undefined} \end{cases}\right\} & \text{if } t \in ({T'}_1^{arg} \setminus {T'}_0^{arg}) \uplus T_1^{out} \\ \theta^{\mathsf{S}} in_0(i) \qquad \text{if } t = down(i) \in T^{down} \ . \end{cases} .$$

We now proceed to prove that $\mathcal{A}(\rho\text{▪}\eta) = \rho\text{▪}\mathcal{A}(\eta)$. It is evident from above that $V'' = V'$, $S'' = S'$ and $T'' = T'$. It is simple to check that $ctr'' = ctr'$. It only remains to prove that $src' = src''$, which we do by considering the cases for an argument $t \in T'$. Note that $src''$ and $src$ are defined directly, while $src' = \rho^{\mathsf{S}} \circ src \circ \rho^{\mathsf{T}}$.

**Case** $t \in ({T'}_1^{arg} \setminus {T'}_0^{arg}) \uplus T_1^{out}$. Now let $t_1 = \rho^{\mathsf{T}}(t), = \rho_1^{\mathsf{T}}(t)$ also by the Lemma. We have on the one hand $src'(t) = \rho^{\mathsf{S}} src(t_1)$, and on the other-hand $src''(t)$ depends upon the value (and definedness) of $\theta^{\mathsf{T}}(t)$. But $\theta^{\mathsf{T}}(t) = {\rho_0^{\mathsf{T}}}^{-1} \eta^{\mathsf{T}} \rho_1^{\mathsf{T}}(t), = {\rho_0^{\mathsf{T}}}^{-1} \eta^{\mathsf{T}}(t_1)$. We may therefore proceed according to the value of $\eta^{\mathsf{T}}(t_1)$:

**Case** $\eta^{\mathsf{T}}(t_1)$ defined, $= out_0(j)$, $j \in [n_0]$. Then $src'(t) = \rho^{\mathsf{S}} src(t_1) = \rho^{\mathsf{S}} up(j) = up(j)$; on the other hand $\theta^{\mathsf{T}}(t) = {\rho_0^{\mathsf{T}}}^{-1} out_0(j) = out_0(j)$, whence $src''(t) = up(j)$ also.

**Case** $\eta^{\mathsf{T}}(t_1)$ undefined. Then $src'(t) = \rho^{\mathsf{S}} src(t_1) = \rho^{\mathsf{S}} src_1(t_1) = \rho_1^{\mathsf{S}} src_1(t_1)$ by the Lemma, since in this case we can check that $src(t_1) \in ({S'}_1^{bind} \setminus {S'}_0^{bind}) \uplus S_1^{in}$. On the other hand $src''(t) = src'_1(t) = \rho_1^{\mathsf{S}} src_1 \rho_1^{\mathsf{T}}(t), = \rho_1^{\mathsf{S}} src_1(t_1)$ also.

**Case** $t = down(i) \in T^{down}$, $i \in [m_0]$. Then $\rho^{\mathsf{T}}(t) = t$, so $src'(t) = \rho^{\mathsf{S}} src(t) = \rho^{\mathsf{S}} \eta^{\mathsf{S}} in_0(i)$, $= \rho_1^{\mathsf{S}} \eta^{\mathsf{S}} in_0(i)$ by the Lemma, whose condition again holds. On the other hand $src''(t) = \theta^{\mathsf{S}} in_0(i) = \rho_1^{\mathsf{S}} \eta^{\mathsf{S}} {\rho_0^{\mathsf{S}}}^{-1} in_0(i), = \rho_1^{\mathsf{S}} \eta^{\mathsf{S}} in_0(i)$ also.

This concludes the proof that $src' = src''$, and that $\mathcal{A}(\rho\text{▪}\eta) = \rho\text{▪}\mathcal{A}(\eta)$ . ∎

**Proposition 41** *Let $\eta : G_0 \to G_1$ be an inclusion embedding. Then $\mathcal{D}(\mathcal{A}(\eta)) = \eta$.*

**Proof** Let $G_i : (m_i, n_i) = (V_i, ctr_i, src_i)$, for $i = 0, 1$. Then from Definition 29 the components of $\mathcal{A}(\eta)$ are as follows:

$$\begin{array}{rcl} \mathcal{A}(\eta) & = & (V, ctr, src) \\ & & \text{where} \\ V & = & V_1 \setminus V_0 \\ ctr & = & ctr_1 \restriction V \\ S & = & (S_1^{bind} \setminus S_0^{bind}) \uplus S_1^{in} \uplus S^{up} \\ T & = & (T_1^{arg} \setminus T_0^{arg}) \uplus T_1^{out} \uplus T^{down} \end{array}$$

and

$$src(t) = \begin{cases} \left.\begin{cases} up(j) & \text{if } \eta^{\mathsf{T}}(t) \text{ defined, i.e. } = out_0(j), j \in [n_0] \\ src_1(t) & \text{if } \eta^{\mathsf{T}}(t) \text{ undefined} \end{cases}\right\} & \text{if } t \in T^{arg} \uplus T_1^{out} \\ \eta^{\mathsf{S}} in_0(i) & \text{if } t = down_(i) \in T^{down}, i \in [m_0] \,. \end{cases}$$

Let $\theta : G_0 \to G_1$ be $\mathcal{D}(\mathcal{A}(\eta))$. Our task is to prove that $\theta = \eta$. From Definition 37 the components of $\theta$ are

$$\begin{array}{rcl} \theta^{\mathsf{V}} : V_0 \hookrightarrow V_1 & = & \mathsf{Id}_{V_0} \\ \theta^{\mathsf{S}} : S_0 \hookrightarrow S_1 & = & \sigma_0 \\ \theta^{\mathsf{T}} : T_1 \rightharpoonup T_0 & : & t \mapsto \begin{cases} t & \text{if } t \in T_0^{arg} \\ \left.\begin{cases} out_0(j) & \text{if } src(t) = up(j) \in S^{up} \\ \text{undefined} & \text{if } src(t) \in S^{bind} \uplus S^{in} \end{cases}\right\} & \text{if } t \in T^{arg} \uplus T^{out} \,. \end{cases} \end{array}$$

Now since $\eta$ is an inclusion, first we have $\eta^{\mathsf{V}} = \mathsf{Id}_{V_0}$, hence $\theta^{\mathsf{V}} = \eta^{\mathsf{V}}$. Next we have that $\eta^{\mathsf{S}} \restriction S_0^{bind}$ and $\eta^{\mathsf{T}} \restriction T_0^{arg}$ are identities, and this agrees with $\theta^{\mathsf{S}}$ and $\theta^{\mathsf{T}}$ as defined. It remains to show that

$$\begin{array}{ll} \theta^{\mathsf{S}}(s) = \eta^{\mathsf{S}}(s) & \text{for } s \in S_0^{in} \,, \text{ and} \\ \theta^{\mathsf{T}}(t) = \eta^{\mathsf{T}}(t) & \text{for } t \in T_1^{arg} \uplus T^{out} \,. \end{array}$$

For the former, let $s = in_0(i) \in S_0^{in}$; then $\theta^{\mathsf{S}}(s) = \sigma_0(s) = src\, down(i) = \eta^{\mathsf{S}}(s)$.

For the latter, let $t \in T^{arg} \uplus T_1^{out}$; then there are two cases:

**Case** $\eta(t)$ defined, i.e. $= out_0(j)$, $j \in [n_0]$. Then $src(t) = up(j)$, whence $\theta(t) = out_0(j)$ also.

**Case** $\eta(t)$ undefined. Then $src(t) = src_1(t) \in S^{bind} \uplus S^{in}$ since the alternative, $src_1(t) \in S_0^{bind}$, contradicts E-TARGS; so $\theta^{\mathsf{T}}(t)$ is undefined also.

This concludes the proof that $\mathcal{D}(\mathcal{A}(\eta)) = \eta$. ∎