Synthesising Labelled Transitions and Operational Congruences in Reactive Systems, Part 1

James J. Leifer

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Synthesising Labelled Transitions and Operational Congruences in Reactive Systems, Part 1

James J. Leifer

N° 4394

March 2002

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4394--FR+ENG

Synthesising Labelled Transitions and Operational Congruences in Reactive Systems, Part 1

James J. Leifer

Thème 1 — Réseaux et systèmes
Projet Moscova

Rapport de recherche n° 4394 — March 2002 — 54 pages

Abstract: The dynamics of process calculi, e.g. CCS, have often been defined using a labelled transition system (LTS). More recently it has become common when defining dynamics to use *reaction rules* —i.e. unlabelled transition rules— together with a structural congruence. This form, which I call a *reactive system*, is highly expressive but is limited in an important way: LTSs lead more naturally to operational equivalences and preorders. This paper shows how to *synthesise* an LTS for a wide range of reactive systems. A label for an agent (process) 'a' is defined to be any context 'F' which intuitively is just large enough so that the agent 'Fa' ('a' in context 'F') is able to perform a reaction step. The key contribution of my work is the precise definition of "just large enough" in terms of the categorical notion of *relative pushout* (RPO). I then prove that several operational equivalences and preorders (strong bisimulation, weak bisimulation, the traces preorder, and the failures preorder) are congruences when sufficient RPOs exist.

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

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex (France)
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

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

Résumé : La tradition veut que l'on définisse la dynamique des calculs de processus tels que CCS à l'aide d'un système de transitions étiquetées (LTS). Récemment, il est devenu courant d'utiliser plutôt des *règles de réaction* (c'est-à-dire des règles de transitions non étiquetées) associées à une congruence structurelle. Cette présentation, que j'appelle *système réactif*, est très expressive, mais souffre d'une limitation importante : les LTS permettent de définir de manière plus naturelle des équivalences et préordres opérationnels. Cet article montre comment *synthétiser* des LTS pour de larges classes de systèmes réactifs. On prend comme étiquette d'un agent (processus) 'a' n'importe quel contexte 'F' qui, intuitivement, est tout juste assez large pour que l'agent 'Fa' ('a' sous le contexte 'F') puisse effectuer une étape de réaction. Ma principale contribution est une définition précise de "juste assez large" à l'aide de la notion catégorique de somme amalgamée relative ("relative pushout", RPO). Je prouve ensuite que plusieurs équivalences et préordres opérationnels (bisimulation forte, bisimulation faible, préordre des traces, préordre des échecs) sont des congruences lorsqu'il existe assez de RPO.

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

Contents

1 Introduction **2**
1.1 Overview . . . 2
1.2 Historical background and motivation . . . 3
1.3 Contexts as labels . . . 8
1.4 Relative pushouts . . . 10
1.5 Other work . . . 14
1.6 Outline . . . 16
1.7 Acknowledgements . . . 17

2 Operational congruences for reactive systems **18**
2.1 Formalisation of reactive systems . . . 18
2.2 Categorical basis for contextual labels . . . 21
2.3 Labelled transitions and congruence for strong bisimulation . . . 25

3 Further congruence results **29**
3.1 Introduction . . . 29
3.2 Functorial reactive systems . . . 30
3.3 Cutting and pasting portable IPO squares . . . 32
3.4 Strong bisimulation . . . 34
3.5 Weak bisimulation . . . 36
3.6 Traces preorder . . . 39
3.7 Failures preorder . . . 41
3.8 Multi-hole contexts . . . 42

4 Conclusions **48**

Bibliography **50**

1 Introduction

1.1 Overview

This paper is concerned with *process calculi*, which are mathematical models of computation. Process calculi come in many forms, but share several common features. Much as the λ-calculus isolated the canonical features of sequential computation, process calculi do the same for concurrent phenomena, such as non-determinism, synchronisation, and communication. By concentrating on a few core syntactic primitives, process calculi provide a setting in which to reason about these phenomena without the added baggage associated with a full programming language.

There are three critical ingredients beyond syntax that a process calculus may possess. I summarise these now and discuss each in greater detail later.

The first consists of *unlabelled transitions*, or *reactions* as I call them in this paper, which characterise only the internal state changes of an *agent* (a process). Reactions do not require interaction with the surrounding environment.

The second consists of *labelled transitions*, which characterise the state changes that an agent may undergo in concert with the environment. Each transition is associated with a label, which records the interaction with the environment (for example an output on a channel) that enables the transition. A label is thus an *observation* about the behaviour of an agent. Normally, the reactions of an agent are special labelled transitions for which the interaction with the environment is vacuous (so-called τ-transitions).

The third consists of *operational preorders and equivalences*, which characterise when one computation is respectively behaviourally replaceable by, and behaviourally equal to, another. An equivalence is most useful when it is a *congruence*, i.e. is preserved by the syntactic constructions of the calculus; in this case, proving the equivalence of two large agents can be reduced to proving the equivalence of components. Thus congruence is one of the most desirable properties an equivalence may possess. These remarks apply as well to operational preorders.

As I will describe later, much research has concentrated on finding general definitions of operational preorders and equivalences in terms of labelled transitions. For any specific process calculus, it is then a challenge to prove that the preorder or equivalence in question is a congruence. But what happens when there are no labelled transitions, only reactions? The latter situation has become common since reactions provide a clean way of specifying allowable state changes without commitment to particular observables. Reactions are highly expressive but limited in an important way: they do not lead as naturally as labelled transitions do to congruential operational equivalences and preorders.

The central problem addressed by this paper is as follows: From a process calculus with only a reaction relation —called a *reactive system*— can we synthesise a tractable

labelled transition relation? By tractable, I mean two things: the labels come from some small set; and, the labelled transitions readily yield operational preorders and equivalences that are congruences.

My approach is to take a label to be a context (an agent with a hole in it). For any agent a, a label is any context F which intuitively is just large enough so that the agent Fa ("a in context F") is able to perform a reaction. The key contribution of this work is to make precise the definition of "just large enough" in terms of the categorical notion of *relative pushout* (RPO) and to show that several operational equivalences and preorders, namely strong bisimulation, weak bisimulation, the traces preorder, and the failures preorder, are congruences when sufficient RPOs exist.

By showing that the RPO property is a sufficient condition for congruence, this paper brings together two of the main themes of concurrency semantics: the failures preorders of Hoare and Roscoe (developed originally for CSP) and the bisimulation equivalences of Milner and Park (developed originally for CCS). The former are model-theoretic, comparing programs as points in a complete partial order; the latter are operationally-based, comparing programs coinductively. It is striking, therefore, that the same RPO hypothesis serves for my congruence proofs of both.

This paper is the first part of a series of two. The companion paper [Lei02], referred to as Part 2 throughout, proves that a general class of functors does indeed satisfy the axioms of a *functorial reactive system* (an elaboration of the notion of reactive system referred to above). Moreover, it shows that a significant example of such, namely a simplified version of Milner's action calculi, has sufficient RPOs, as required by all the congruence proofs.

In the rest of this section, I discuss some of the history of operational congruences and offer some background as to why reactions have become prominent. I then sketch the solution strategy presented in this paper. I relate my approach to work by other researchers and to material I have jointly published. I conclude with an overview of the structure of the sections.

1.2 Historical background and motivation

It may seem strange at first to investigate the "derivation of operational congruences" since congruence is usually a postulated notion in mathematics: to define a model, one *chooses* which elements are equivalent. But choosing has become difficult for many models of computation, creating a gap between the properties such as reaction that are easy to postulate and those such as equivalence that are useful to have.

To illustrate how this gap opened, let us look at some strands in the history of process calculi. Starting with the seminal work by E.F. Moore [Moo56] which examined "finite automata from the experimental point of view", theoretical computer scientists have pursued the notion that the *observable behaviour* of a computation is more fundamental than

the form in which the computation is written; in this view, two programs are equivalent when they are indistinguishable by an outside observer. In the case of automata, an observation consists of a *labelled transition* of the form $a \xrightarrow{x} \triangleright a'$: automaton a can input the label x and become a'. (This is a slight simplification of Moore's original notion, which distinguished input from output labels.)

One automaton refines another if the former has a subset of the labelled transitions of the latter, whether or not the two have different syntactic forms. For example, $a \hat{=} x.(y+z)$ and $b \hat{=} x.y + x.z$ both can input the same strings of labels, $\langle\rangle, \langle x\rangle, \langle xy\rangle, \langle xz\rangle$, so are equal (refine each other) despite their syntactic differences. This notion of comparing computations based on their labelled transitions is now known as the *traces preorder*; the interpretation of computations in terms of their traces is the *traces model* (see [Ros98]).

The traces preorder is attractive for its simplicity, but is limited. Some of its inadequacies with respect to particular applications were overcome in the late 1970s by Hoare and Milner in ways that I describe below. The primary weakness of the traces preorder is its inability to handle the subtleties of non-determinism. This is evident if we consider the pair a, b defined earlier. After inputting x, a is in a state that can input either y or z; but b takes a "silent" choice (an internal choice not controllable by external influence) when inputting x and enters one of two states: either a state in which only y can be input or a state in which only z can be input. Even though a and b have identical traces, their non-deterministic behaviour is different.

The problem of handling non-determinism together with causality (the dependence of one transition upon an earlier one) was first addressed by *Petri nets* (see, for example, [Rei85]). Researchers on Petri nets did not originally consider equivalences or preorders for them. However, *event structures*, which are similar to traces but account for the causality and non-determinism of events, are a setting for modelling Petri nets and other causal systems and yield notions of equivalence [WN95]. Because Petri nets do not have a compositional syntax, it is difficult to understand what it means for an equivalence to be a congruence. For this reason, I do not discuss them further and confine my attention to process calculi with compositional syntax.

Different applications for process calculi have different informal requirements for when the patterns of labelled transitions for two agents are the same (in the case of equivalences) or refine one another (in the case of preorders). No one definition can be faithful to all possible requirements. Given that, we can consider schemas parameterised by a labelled transition relation for defining different equivalences and preorders — and thus take labelled transitions to be the *postulated* part of a process calculus from which equivalences and preorders are *derived*. This difference lends labelled transitions their power: a labelled transition relation captures the important interactions between a computation and its environment without making a commitment to a specific equivalence or preorder.

As I will discuss later, many new process calculi for modelling distributed computation do not include axioms for labelled transitions but instead rest on an even simpler base.

Hoare's work on Communicating Sequential Processes (CSP) [Hoa78, Hoa85] and Milner's on Communicating Concurrent Systems (CCS) [Mil80, Mil88] addressed the problem of comparing computations in a way that is sensitive to non-determinism, and hence distinguishes a from b (the examples I gave earlier).

Hoare's approach in CSP was to present every agent as an explicit set-theoretic construction including, amongst other data, *failures*; the latter are traces, each of which is enriched with a set of *refusals* indicating which labels may be refused by the agent after inputting the trace. For example, after inputing x, b can refuse y and can refuse z but a cannot refuse either. Each constructor in CSP (prefixing, parallel composition, choice, etc.) is defined in terms of the manipulation of failures (as a continuous function on the complete partial order of agents). The explicit set-theoretic representation of agents in CSP supports the design of model checkers [Ros94, Ros98] which are, for example, effective in detecting bugs in protocols [Low96]. I concentrate in this paper on the strand of research originating with CCS, but return to the failures preorder in Subsection 3.7 when proving that it is a congruence.

In Milner's CCS, agents have a free syntax and the labelled transition relation is generated by inference rules in the style of Plotkin's structured operational semantics [Plo81], e.g.

$$x.a \xrightarrow{x} \triangleright\, a \qquad\qquad \frac{a \xrightarrow{\alpha} \triangleright\, a'}{a + b \xrightarrow{\alpha} \triangleright\, a'}$$

Remark: Throughout this paper $a, b, c, \ldots$ are used to denote agents even when this contradicts historical conventions. The labels (ranged over by $\alpha \ldots$) comprise input channels $x, y, z, \ldots$ and output channels $\bar{x}, \bar{y}, \bar{z}, \ldots$ and a special distinguished element τ. This terminology is somewhat misleading because no data is actually input or output; they are just complementary flavours. In the π-calculus, which I describe later, the differences are significant.

The key idea in CCS is that of synchronisation. If one agent can output x and the other can input x then their parallel composition can synchronise on x. The result is a τ-transition which records the synchronisation but leaves the name on which it occurs anonymous:

$$\frac{a \xrightarrow{\bar{x}} \triangleright\, a' \quad b \xrightarrow{x} \triangleright\, b'}{a \mid b \xrightarrow{\tau} \triangleright\, a' \mid b'}$$

Milner considered several equivalences for CCS; the main ones were *strong and weak bisimulation*. Both kinds employ a coinductive form of definition which gives a powerful proof

technique for comparing agents (an idea that originates with Park [Par81]) and provide a general way of defining an equivalence parameterised by a labelled transition system. Strong bisimulation relies on no assumptions about the underlying structure of the labels; weak bisimulation requires a distinguished τ-transition. I will discuss both in detail later (Subsection 2.3 and Subsection 3.5) but summarise the two now.

Both strong and weak bisimulation are sensitive to non-determinism and are able to distinguish the pair a, b described earlier in the discussion of the traces preorder. A *strong bisimulation* $\mathcal{S}$ is a relation on agents satisfying the following coinductive property: for all $(a, b) \in \mathcal{S}$ and all labelled transitions $a \xrightarrow{\alpha}\triangleright\, a'$, there exists b' such that $b \xrightarrow{\alpha}\triangleright\, b'$ and $(a', b') \in \mathcal{S}$ (and vice versa). Informally, "if a and b are $\mathcal{S}$-related then whatever a can do, b can do too, and they both end up in $\mathcal{S}$-related states; likewise for whatever b can do". The largest strong bisimulation relation (which is the union of all strong bisimulations) is denoted by $\sim$. The largest weak bisimulation, denoted by $\approx$, is coarser because it is flexible in allowing τ-transitions to be collapsed and expanded when comparing two agents. For example, a agent that inputs x and then immediately outputs y would be related by weak bisimulation to one that inputs x, then has several τ-steps (internal computations), and finally outputs y.

Milner proved that both kinds of bisimulation are congruences (though not with respect to sum for weak bisimulation — see Subsection 3.5 for further discussion). Such proofs require care in CCS and are more difficult in later calculi. This is the price of avoiding CSP-style explicit representations of agents and agent constructors. It is a central theme of this paper to provide mathematical tools for easing the burden of proving that operationally defined equivalences (such as bisimulation) are congruences.

The idea of using a τ-like transition to record a primitive computational step plays a central role in later calculi. These transitions are variously called *reaction rules*, *reduction rules*, *firing rules*, etc.; I shall use *reaction rules* throughout. In CCS, the τ transition relation requires the entire collection of labelled transition rules to generate it. A dramatic simplification was proposed in the Chemical Abstract Machine (CHAM) of Berry and Boudol [BB90, BB92] and used in work [Mil92] on the π-calculus of Milner, Parrow, and Walker [MPW89, MPW92]. These calculi were the first to employ a lightweight quotient of the agent terms, called a structural congruence, in order to make their reaction rules easy to define. I shall confine the discussion to the π-calculus. (CHAM treats the quotient more explicitly with "heating and cooling" rules, though the idea is similar.)

Structural congruence is an equivalence relation $\equiv$ on agents that allows the parts of an agent to rearrange themselves, e.g.

$$a \mid b \equiv b \mid a \qquad a \mid (b \mid c) \equiv (a \mid b) \mid c \qquad \cdots$$

The reaction relation $\longrightarrow\!\triangleright$, which characterises the primitive computational step (namely communication of a name over a channel), is simple to define; it is the smallest relation

satisfying the rule

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

that respects structural congruence

$$\frac{a \equiv a' \quad a' \longrightarrow\!\triangleright\, b' \quad b' \equiv b}{a \longrightarrow\!\triangleright\, b}$$

and is closed under all non-guarding contexts C (e.g. $c \mid -$; see Subsection 2.1 for further discussion)

$$\frac{a \longrightarrow\!\triangleright\, b}{C[a] \longrightarrow\!\triangleright\, C[b]}\ .$$

Thus the agent $\bar{x}\langle y\rangle \mid (c \mid x(z).b)$ has a reaction even though the parts needed to enable the reaction — namely $\bar{x}\langle y\rangle$ and $x(z).b$ — are not adjacent:

$$\bar{x}\langle y\rangle \mid (c \mid x(z).b) \quad \equiv \quad (\bar{x}\langle y\rangle \mid x(z).b) \mid c \quad \longrightarrow\!\triangleright \quad \{y/z\}b \mid c\ .$$

The ease with which reaction rules are defined in this style facilitated an outpouring of new process calculi for modelling encrypted communication [AG97], secure encapsulation [SV99], agent migration [CG98, Sew98, FGL$^+$96] and so on. Each isolates a computational phenomenon and presents it via a reaction rule together with a structural congruence over some syntax. Here are two examples of reaction rules:

- In Cardelli and Gordon's ambient calculus, one ambient may move inside another (like a packet through a firewall):

$$y[in\ x.a \mid b] \mid x[c] \longrightarrow\!\triangleright\, x[y[a \mid b] \mid c]\ .$$

- In Sewell and Vitek's Box-π calculus, a message may move from inside to outside a wrapper and is decorated with the wrapper's name while doing so:

$$y[\bar{x}^{\uparrow}v \mid b] \longrightarrow\!\triangleright\, \bar{x}^{\bar{y}}v \mid y[b]\ .$$

In the π-calculus the communication of names along channels presented subtleties in the design of a labelled transitions system and of equivalences (of which several are now studied [SW01]); the proofs of congruence for these equivalences require care. For many of the newer calculi, such as those listed above, the problem of choosing appropriate labelled transitions and proving that bisimulation is a congruence is difficult, and, in many cases, not attempted. In all these cases, experimentation is costly: a slight modification of the syntax or reaction rules often causes the labelled transition relation to change and breaks congruence proofs, forcing them to be reconstructed.

So the gap first described at the beginning of this subsection has widened as the data defining a typical process calculus have changed. Labelled transitions are no longer postulated primitives of a calculus but instead are teased out from the four fundamental components:

- syntax (agents and agent contexts);
- structural congruence;
- set of reactive (non-guarding) contexts;
- reaction rules.

I call a process calculus containing these components a *reactive system*.

In this way, a reactive system closely resembles instances of the λ-calculus [Bar84]. The latter consist of a simple syntax, a structural congruence based on α-conversion, a set of reactive contexts (known as "evaluation contexts" [FF86]) chosen to force strategies such as call-by-name or call-by-value, and a reaction rule based on β-reduction.

There is however an important difference which renders the problem of finding useful equivalences for process calculi more difficult: their reaction relations are usually neither confluent nor normalising. As a result, equivalences for λ-calculi, such as those based on normal forms or on termination properties [Bar84, Lév78], do not provide viable routes to follow. Therefore we need to consider equivalences based on bisimulation or other techniques that make no assumptions about confluence or normalisation properties. Yet bisimulation requires labelled transitions, which are not provided in a reactive system, and proofs of congruence, which can be difficult and fragile, as already discussed. The next subsection outlines the strategy for *synthesising* labelled transitions which is the basis for the work in this paper.

1.3 Contexts as labels

We wish to answer two questions about an arbitrary reactive system consisting of agents (whose syntax may be quotiented by a structural congruence) and a reaction relation $\longrightarrow$ (generated by reaction rules):

1. Can we *synthesise* a labelled transition relation $\xrightarrow{\lambda}$ where λ comes from a small set of labels that intuitively reflect how an agent interacts with its environment?
2. Under what general conditions is strong bisimulation (and, more generally, other preorders and equivalences) over $\xrightarrow{\lambda}$ a congruence?

We can begin to address question 1 by considering CCS. Let a, b range over agents, C, D, F range over contexts (agents with a hole), and x range over names. The usual labelled

transitions $\xrightarrow{\lambda}\triangleright$ for $\lambda ::= \bar{x} \mid x \mid \tau$ reflect an agent's *capability* to engage in some behaviour, e.g. $\bar{x}.a \mid b$ has the labelled transition $\xrightarrow{\bar{x}}\triangleright$ because $\bar{x}.a \mid b$ can perform an output on x. However, if we shift our emphasis from characterising the capabilities of an agent to the *contexts* that enable the agent to react, then we gain an approximate answer to question 1 by choosing the contexts themselves as labels; namely we define

$$a \xrightarrow{F}\triangleright a' \qquad \text{iff} \qquad Fa \longrightarrow\triangleright a' \tag{1}$$

for all contexts F. (We denote context composition and application by juxtaposition throughout. We regard a context with a trivial hole as an agent, so application is a special case of composition.) Instead of observing that $\bar{x}.a \mid b$ "can do an $\bar{x}$" we might instead see that it "interacts with an environment that offers to input on x, i.e. reacts when placed in the context $- \mid x$". Thus, $\bar{x}.a \mid b \xrightarrow{-|x}\triangleright a \mid b$.

The definition of labelled transition in (1) is attractive when applied to an arbitrary reactive system because it in no way depends upon the presence or absence of structural congruence. Furthermore, it is generated entirely from the reaction relation $\longrightarrow\triangleright$ (question 1); and, strong bisimulation over the synthesised labelled transition relation $\dot{\longrightarrow}\triangleright$ is a congruence, (question 2). The proof of the latter is straightforward: let C be an arbitrary context and suppose $a \sim b$; we show that $Ca \sim Cb$. Suppose $Ca \xrightarrow{F}\triangleright a'$; by definition, $FCa \longrightarrow\triangleright a'$, hence $a \xrightarrow{FC}\triangleright a'$. Since $a \sim b$, there exists b' such that $b \xrightarrow{FC}\triangleright b'$ and $a' \sim b'$. Hence $Cb \xrightarrow{F}\triangleright b'$, as desired. The other direction follows by symmetry.

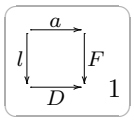
Nonetheless, the definition in (1) is unsatisfactory: the label F comes from the set of *all* contexts — not the "small set" asked for in question 1 — thus making strong bisimulation proofs intolerably heavy. Also, the definition fails to capture its intended meaning, namely that $a \xrightarrow{F}\triangleright a'$ holds when a *requires* the context F in order that a reaction is enabled in Fa: there is nothing about the reaction $Fa \longrightarrow\triangleright a'$ that forces all of F — or indeed any of F — to be used. In particular, if $a \longrightarrow\triangleright a'$ then for all contexts F that preserve reaction, $Fa \longrightarrow\triangleright Fa'$, hence $a \xrightarrow{F}\triangleright Fa'$; thus a has many labelled transitions that reflect nothing about the interactions of a itself.

Let us unpack (1) to understand in detail where it goes wrong. Consider an arbitrary reactive system equipped with a set Reacts of reaction rules; the reaction relation $\longrightarrow\triangleright$ contains Reacts and is preserved by all contexts, as formalised by the following axiom and inference rule:

$$l \longrightarrow\triangleright r \quad \text{if } (l,r) \in \mathsf{Reacts} \qquad\qquad \frac{a \longrightarrow\triangleright a'}{Ca \longrightarrow\triangleright Ca'} \quad .$$

Expanding (1) according to this definition of $\longrightarrow\triangleright$ we have:

$$\begin{aligned} a \xrightarrow{F}\triangleright a' \qquad &\text{iff} \qquad Fa \longrightarrow\triangleright a' \\ &\text{iff} \qquad \exists (l,r) \in \mathsf{Reacts}, D. \quad Fa = Dl \;\&\; a' = Dr \quad . \end{aligned} \tag{2}$$



The requirement $Fa = Dl$ in (2) is rendered by a commuting square (as shown in Figure 1) in some category whose arrows are the agents and contexts of the reactive system. This requirement reveals the flaw described earlier: nothing in (2) forces F and D to be a "small upper bound" on a and l. The next subsection explores the challenges involved in making precise what "small upper bound" means.

1.4 Relative pushouts

For several years I tried to make precise what "small" means. I was mired in detailed arguments about specific examples of contexts for which I would search for a "dissection lemma", having roughly the following form: given $Fa = Dl$, there exists a "maximum" C, such that for some F' and D' we have $F'a = D'l$, $F = CF'$ and $D = CD'$. In other words, dissection peels off as much as possible from the outside of F and D, decomposing $Fa = Dl$ as $CF'a = CD'l$. If the dissection of $Fa = Dl$ peels off nothing, i.e. produces the decomposition $\mathsf{id}\, Fa = \mathsf{id}\, Dl$, where id is the identity context, then F and D are "a small upper bound". But the problem then remains: when is C the "maximum" possible context that can be peeled off?

In this subsection I look at some examples of dissection in order to illustrate just how subtle the problem is. These examples motivate a solution that I introduce here and present in detail later in this paper.

For some particular classes of contexts, it is easy to understand what to do. Consider this example of two compositions $Fa = Dl$ in a free term algebra:

$$\Big(\alpha\langle\alpha'\langle\beta\langle-\rangle\rangle\rangle\Big) \circ \gamma \quad = \quad \Big(\alpha\langle\alpha'\langle-\rangle\rangle\Big) \circ \Big(\beta\langle\gamma\rangle\Big) \quad = \quad \alpha\langle\alpha'\langle\beta\langle\gamma\rangle\rangle\rangle\,.$$

(For the sake of clarity, I use $\circ$ for composition in this example.) The maximum context C that can be peeled off is $\alpha\langle\alpha'\langle-\rangle\rangle$. Nothing more can be peeled off because $C = D$. The only other possibilities are $-$ and $\alpha\langle-\rangle$, neither of which is as big as C. So C it is! Of course I am arguing informally here, but it is straightforward to make this line of thinking precise.

Free term algebras are the simplest setting in which to consider dissection exactly because they are *free*. When passing to a syntax quotiented by a non-trivial structural congruence, the question becomes difficult, especially when compounded with naming structure such as in the π-calculus. Consider a typical π-calculus agent such as $a \mathrel{\hat{=}} (\nu u)(\bar{x}\langle u\rangle \mid x(z).\bar{y}\langle z\rangle)$. Notice that a contains a parallel constructor $|$, which is associative and commutative and has identity $\mathbf{0}$; also, u and z are bound, so are subject to α-conversion. Furthermore, the name x is used twice (which requires attention when substituting another name for it) and is discarded after the reaction $a \longrightarrow\!\!\triangleright\, a'$, where $a' \mathrel{\hat{=}} (\nu u)(\bar{y}\langle u\rangle)$.

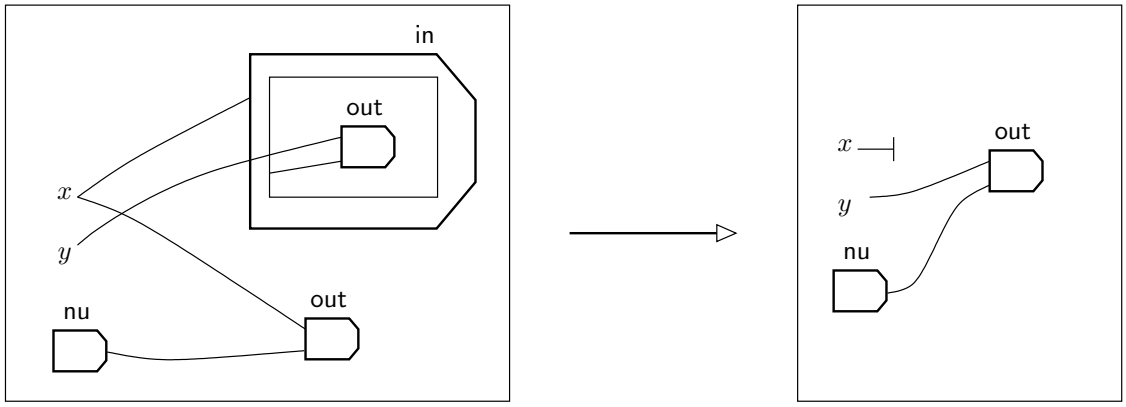


Figure 2: A graphical representation of a π-calculus reaction $(\nu u)(\bar{x}\langle u\rangle \mid x(z).\bar{y}\langle z\rangle) \longrightarrow\!\triangleright (\nu u)(\bar{y}\langle u\rangle)$

This structure demands careful treatment and makes dissection for the π-calculus difficult: for term algebra, one can incrementally peel off a top-level function symbol, but for the π-calculus, there is no notion of a "top-level" constructor. The structural congruence of the π-calculus equates so many different syntactic representations of the same agent that it is difficult to understand where to begin. Without any naming structure, parallel composition becomes easier to handle, as shown by Sewell [Sew]. As I explain in Subsection 1.5, it is the treatment of names that distinguishes the dissection results in this paper from his.

A possible approach is to abandon tree-like syntax and to think in terms of graph-like syntax that automatically quotients out many ignorable differences. Even if a dissection result could be proved for some graph-theoretic representation of the π-calculus, it would not necessarily generalise smoothly to other calculi. As a result, I studied dissection for Milner's action calculi [Mil96], which are a family of reactive systems. The syntax of action calculi is sufficiently rich to embrace process calculi such as π-calculus, the λ-calculus, and the ambient calculus. Action calculi are introduced in Subsection 4.2 in Part 2; here I confine my attention to a few salient features of their graphical form.

Consider the example shown in Figure 2; it illustrates a pair of *action graphs* —agents in an action calculus— that represent the π-calculus reaction $a \longrightarrow\!\triangleright a'$ given earlier. An action graph consists of nodes (rectangles with two blunt corners) and arcs:

- Nodes are labelled with *controls* from a *control signature* of primitives. Each action calculus may have a different control signature, such as $\{\mathsf{nu}, \mathsf{in}, \mathsf{out}\}$ for a simple π-calculus without replication or choice, or $\{\mathsf{ap}, \mathsf{lam}\}$ (application and λ-abstraction) for the λ-calculus.

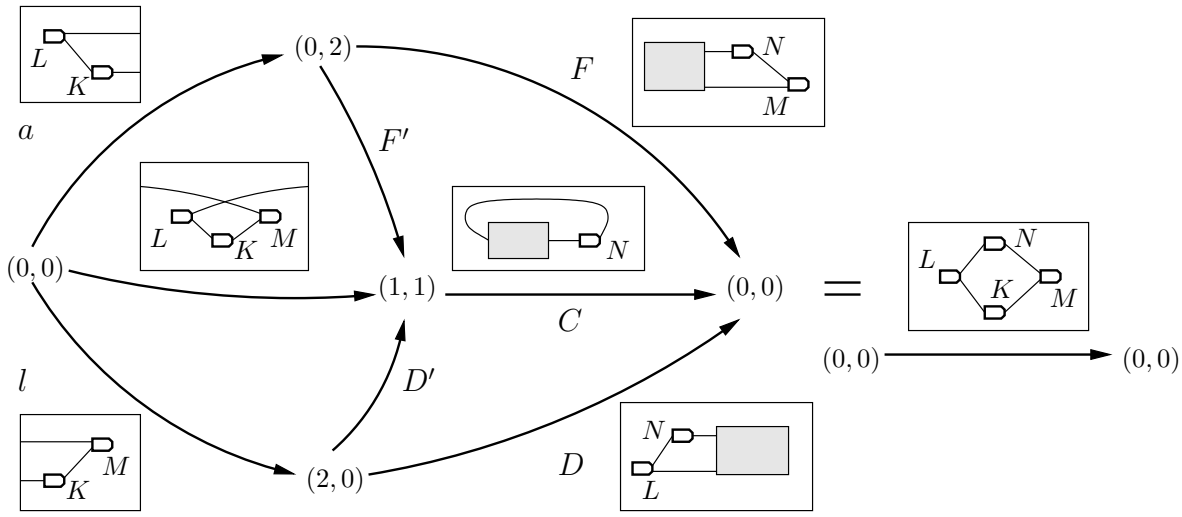


Figure 3: A surprising dissection involving reflexion

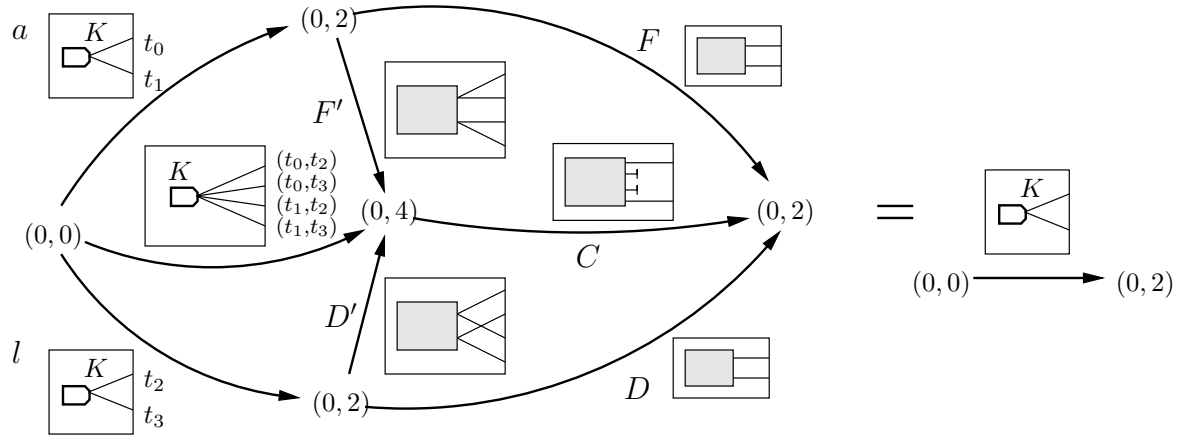


Figure 4: A surprising dissection involving forking

- Arcs represent names. They connect *source ports* to *target ports*. The source ports are arrayed on the left interface of an action graph (such as is shown for the action graph nested inside in) and on the right side of nodes (e.g. nu); they also include free names (e.g. y). The target ports are arrayed on the right interface of action graphs and the left side of nodes. Arcs may be forked from a source port (such as the arcs from x in the LHS) and may be discarded (such as the arc from x in the RHS).

How can we perform dissection on action graphs? It is the wiring structure (the arcs) of an action graph that makes dissection so difficult. Consider the example of a composition

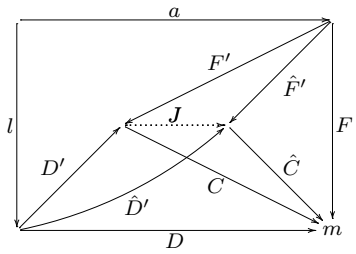


Figure 5: A relative pushout

$Fa = Dl$ shown in Figure 3. (This example is based on one first suggested by Philippa Gardner.) The contexts F and D are just action graphs with a hole (shown as a shaded rectangle). The composition Fa is formed by placing a inside the hole of F and joining up the corresponding arcs. By reasoning about the controls it is possible to see which ones can be peeled off: F has N, M and D has N, L, so C might have N (the intersection of the controls in F and D). Indeed, the C shown does have just one control. But what arcs should C have? Does the loop shown in C make C "maximal" in some sense? Should C have other forked and discarded arcs? How can we choose?

The example in Figure 4 is even more curious: F and D have no nodes at all — they are pure wiring contexts. Since $F = D$ and $a = l$, one might expect that the "maximal" common part that can be peeled off of $Fa = Dl$ is $F = D$ itself. This is not true! The triple F', D', C provides a better dissection in a way that I will make precise below.

The point of these examples is to demonstrate that informal reasoning about dissection is difficult when dealing with contexts containing wiring: There is almost no way to decide just by looking at the arcs themselves whether one dissection is better than another.

Out of this murkiness finally emerged a clearer way. I came upon an *abstract* characterisation, called a *relative pushout*, of which dissections are "best". By abstract, I mean that the characterisation makes no use of any specific properties of contexts except their composability, thus can be instantly cast as a category theoretic property.

Consider the outside commuting square in Figure 5 which shows $Fa = Dl$. A *relative pushout* (RPO) for this square is a triple F', D', C satisfying two properties: first, the triple is a *candidate*, i.e. $F'a = D'l$ and $CF' = F$ and $CD' = D$; second, for any other candidate $\hat{F}', \hat{D}', \hat{C}$, there exists a *unique mediating* J making all the triangles commute (as shown).

RPOs are the key contribution of this paper. It is by working with them, rather than trying to come up with *ad hoc* ways of dissecting contexts in specific examples, that we gain two important advantages:

- RPOs are abstract: as I said before they do not rely on any specific properties of contexts except composability. By defining labelled transitions in terms of RPO constructions, it is possible carry out proofs of congruence (see Section 2 and Section 3) for several operational equivalences and preorders (strong bisimulation, weak bisimulation, the traces preorder, and the failures preorder), for which the proofs *do not depend on any specific properties of contexts except the existence of RPOs*. Thus these results are applicable to any reactive system which possesses sufficient RPOs.

- RPOs provide a unifying discipline for analysing contexts in specific examples. The fuzziness about which dissection is "best" in the examples of action graphs shown earlier disappears: we want the candidate triple F', D', C' from which there is a unique mediator to any other candidate. By this requirement, the candidates shown in Figure 3 and Figure 4 are "best". (There may be many best candidates, but all of them are isomorphic to each other by standard categorical reasoning.) The problem of finding RPOs for graphs is non-trivial, as shown by the lengthy proofs in Chapter 6 of [Lei01], but one is sustained by the unambiguity of the task: there is no vagueness about the properties required of an RPO.

Thus, the notion of an RPO does not solve the problem of finding a dissection, but it makes the problem well-defined and provides a reward for the effort by virtue of the congruence proofs that rely on the existence of sufficient RPOs.

1.5 Other work

This subsection brings together some of the important related work. There is a large collection of literature about process calculi, some of which I referred to in previous subsections. I will concentrate on those pieces of research that most closely impinge on the specific problems that I address in this paper.

The idea of finding conditions under which a labelled transition relation yields an operational congruence has been thoroughly studied in work on *structural operational semantics* (SOS) [GV92, TP97]. The principle is to postulate *rule formats*, conditions on an *inductive* presentation of a labelled transition relation that ensure that operational equivalences (e.g. weak bisimulation [Blo93]) are congruences. There is a fundamental difference between this problem and the one I am looking at. The work on SOS presumes that a labelled transition relation is *already given*: the problem is to show that *if* it satisfies a particular format *then* the congruences follow. My work takes reaction rules as primitive, not labelled transitions: I aim to *synthesise* a labelled transition relation from the reaction rules for which operational equivalences (and preorders) are congruences. In my case, the synthesised labelled transition relation is not inductively presented. It is an open question whether it can be inductively presented and, further, whether this presentation satisfies

some well-known rule format from SOS theory: this possibility is discussed in greater detail in Section 5 of Part 2, which covers future work.

The problem of deriving operational congruences for process calculi from a reaction relation and not from a labelled transition relation is studied in the work on *barbed bisimulation* [MS92], *insensitivity observation* [HY95], and *testing equivalence* [DH84]. The first two construct equivalences by augmenting bisimulation over the reaction relation with observations about related states. For example, in the former, the observations are barbs which detect the ability of an agent to perform an output on a channel. The last (testing) compares two agents by their ability to satisfy the same "may" and "must" tests.

Thus all three depend on primitive observations (not just reactions) though these can be simpler than labelled transitions. The main obstacle to their use is the fact that they do not yield congruences but instead need to be "closed up" by all contexts. For example, it is straightforward to prove barbed equivalence in particular cases but difficult to show barbed congruence because of the heavy quantification over all contexts. Work by Fournet [Fou98] and by Fournet and Gonthier [FG98] ease this burden with techniques that allow a proof of barbed congruence to be broken into pieces, each of which may be carried out using other congruence relations.

Jeffrey and Rathke [JR99] used contexts as the basis for the labels of an LTS in the case of the ν-calculus (a variant of the λ-calculus with fresh name creation). They did not synthesise uniformly these labels from a reaction relation but they were guided by the intuition that the labels are small contexts that enable a reaction. These labels give them the right observational power to obtain useful congruences based on bisimulation.

Sewell's work [Sew] is the closest to mine of all the material I have cited here. He studied the problem of deriving contextual labels for a family of reactive systems (parameterised by arbitrary reaction rules) whose syntax consists of free terms quotiented by a structural congruence for parallel composition. He defined labelled transitions by explicitly reasoning about how a label overlaps with a redex. From this definition, he proved that bisimulation is a congruence by appealing to his specific dissection results (not motivated by RPOs). There are three important differences in approach:

- He dealt with multi-hole redexes which capture the full uniformity of metavariables, thus leading to lighter labelled transitions than I can synthesise with my present RPO technology. I discuss possible remedies for this in Section 5 in Part 2.
- He invented explicitly his definition of labelled transition and his statement of dissection, both of which are complex, without employing RPOs (which I worked with later). As as result, his proof of congruence is not easily generalisable to other syntactic families and other operational equivalences. It seems likely, however, that his dissection results imply the existence of RPOs for the class of reactive systems he considered, and it would be worth trying to recast them so as to make this precise.

- He confined his attention to free term algebras with parallel composition and did not handle wiring structure such as is shown in Figure 4. I discuss wiring in Section 4 in Part 2 — and show detailed proofs involving it in Chapter 6 of [Lei01] where the discipline of seeking RPOs guides the dissection involved. Given the complexity of Sewell's definition of labelled transition and of his statement of dissection, it is difficult to see how these could be generalised to embrace wiring without the benefit of the notion of RPOs or other universal constructions.

The idea of RPOs first appears in [LM00], published jointly with Milner. It contains congruence results for strong bisimulation and weak bisimulation for reactive system (thus partly overlapping with Section 2 in the present paper) not the functorial reactive system considered here in Section 3.

The presentation of action calculi contexts in Section 4 of Part 2 intersects with that of [CLM00], though the latter work does not contain any use of functorial reactive systems, a substantial innovation of the present paper and Part 2.

Milner's recent study of *bigraphical reactive systems* [Mil01] attacks some of the open problems brought out in the present paper and in Part 2. I have inserted pointers to his work in the discussion about the role of precategories in functorial reactive systems (Subsection 3.7 in Part 2) and in the description of future work related to nested graphs, multi-hole contexts and metavariables in reactions, free names and binding, and summation (all in Section 5 in Part 2).

In this paper, I have made no use of the *double pushout* techniques developed in graph rewriting [Ehr79]. These are a way to describe the occurrence of a subgraph —especially a redex— in a graph. To avoid confusion, I should emphasise that *relative* pushouts play quite a different role. In my work, subgraph occurrences are handled by embeddings and contexts; the nature of the graphs (with forked wiring) seems to require a specific approach. But it would be useful to examine in the future how the embeddings relate to the double pushout construction, and how graph-theoretic representations of the syntax of the π-calculus and other calculi formed by quotienting out structural congruence compare to similar work in graph rewriting [CM91, Kön99].

The proofs contained in this paper and in Part 2 are sometimes omitted for the sake of brevity. In all cases I refer to the full proof in my Ph.D. [Lei01].

1.6 Outline

The subsequent sections of this paper are organised in the following way:

Section 2: I make precise the notion of a reactive system and give a definition of labelled transition in terms of *idem pushouts* (IPO), a sister notion of RPOs. I then present a series of results using simple categorical reasoning that shows how to manipulate

IPOs and RPOs. The main theorem then follows by direct use of the categorical results from the section: if sufficient RPOs exist then strong bisimulation is a congruence. I conclude by reproving the same theorem in a cleaner way by isolating two lemmas which give derived inference rules for labelled transitions.

Section 3: I generalise the definition of reactive system by enriching it so as to comprise two categories with a functor $\mathcal{F}$ between them. The idea is that the downstairs category is the one in which one wishes to consider agents and contexts, but for which enough RPOs might not exist. The upstairs category does have RPOs but at the cost of extra intensional information in the arrows and objects. By refining the definition of labelled transition so that it relates arrows downstairs in terms of IPO properties of their preimages upstairs, I obtain congruence results for strong bisimulation, weak bisimulation, the traces preorder, and the failures preorder. Finally, I propose added structure needed in a functorial reactive system to cater explicitly for RPOs that yield multi-hole contexts. I conclude by proving that strong bisimulation is a congruence here as well.

Section 4: This section reviews some of the accomplishments of this paper.

Part 2 continues this thread of work. It shows that a general class of functors satisfies the axioms of functorial reactive systems, from which I argue that a subclass of Milner's action calculi with sufficient RPOs is indeed an instance. Part 2 concludes with a list of open questions, whose solutions are critical to the application of this work to a wide variety of process calculi.

1.7 Acknowledgements

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

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

2 Operational congruences for reactive systems

2.1 Formalisation of reactive systems

In this subsection I investigate how to give reactive systems, which were introduced informally in Subsection 1.2, a precise category-theoretic definition. The goal is to include appropriate categorical structure so that it is possible to synthesise labelled transitions and prove that several operational equivalences and preorders are congruences. To that end, I first study *relative pushouts* (RPOs) and *idem pushouts* (IPOs), which are universal constructions related to pushouts. I then show how to synthesise labelled transitions from a set of reaction rules, with IPOs playing the central role. Finally I prove by categorical reasoning that if sufficiently many RPOs exist then strong bisimulation is a congruence. The next section considers richer notions of equivalences and reactive systems, proving that the former are congruences.

This paper employs only basic category theory, such as simple universal constructions, slices (and coslices), monoidal structures, and functors. Full explanations appear in the classic work by Mac Lane [Mac71], or in [BW01], which is online.

Throughout this section, I use lowercase roman letters $a, b, \ldots$ for *agents* (processes) and uppercase roman letters $C, D, \ldots$ for *agent contexts* (process contexts). Both are arrows in a category of contexts (as explained below). Juxtaposition is used for categorical composition. Other notation is explained as it comes up.

How do we get at the essence of reactive systems, i.e. how do we find mathematical structure that is simple enough to get general results about congruences and rich enough to encompass significant examples? The key ingredients of a reactive system were shown in Subsection 1.2 and are recalled here:

- syntax (agents and agent contexts);
- structural congruence;
- set of reactive (non-guarding) contexts;
- reaction rules.

For each, I outline the mathematical design space and explain the decisions I have taken.

syntax: Since contexts are composable, I take them to be the arrows of some category $\mathbf{C}$. This presents an immediate question. Are the objects of $\mathbf{C}$ agents or sorts?

Following the former route leads to a problem: If we think of a context C as an embedding of an agent a into an agent a', i.e. " $C : a \rightarrow a'$ ", then there is no easy way to apply C to a different agent. For example, we cannot state the congruence

property of an equivalence $\sim$: if $a \sim b$ for agents a and b then C "applied to" a and C "applied to" b are $\sim$-equivalent for all contexts C. (In Chapter 6 of [Lei01] I discuss embeddings, which are critical in proving the existence of universal constructions for categories of graphs.) Alternatively, if the objects are the sorts (or "types" or "arities") of the contexts, then agents are a special subclass of contexts. In particular, agents are contexts with a null hole, i.e. contexts whose domain is some distinguished object 0. Now the congruence property of $\sim$ is neatly rendered: if $a \sim b$ for all arrows $a, b : 0 \rightarrow m$, then $Ca \sim Cb$ for all arrows C with domain m.

For concreteness, consider as an example a category of contexts for some Algol-like programming language. The objects of the category could comprise the usual types: *bool*, *int*, *cmd*. Then we have the following examples of arrows:

$$\begin{array}{lcll} C_0 & \hat{=} & \textbf{if } - \textbf{ then } x := 0 \textbf{ else skip} & : \mathit{bool} \rightarrow \mathit{cmd} \\ C_1 & \hat{=} & 14 < - & : \mathit{int} \rightarrow \mathit{bool} \\ C_0C_1 & = & \textbf{if } 14 < - \textbf{ then } x := 0 \textbf{ else skip} & : \mathit{int} \rightarrow \mathit{cmd}\ . \end{array}$$

Another example is of a category of linear multi-hole term algebra contexts over some signature Σ. These are considered in more detail in Subsection 3.8. The objects are natural numbers. The arrows $m \rightarrow n$ are n-tuples of terms over $\Sigma \cup \{-_1, \ldots, -_m\}$, where each symbol $-_i$ is used exactly once. For example, if $\Sigma = \{\alpha, \alpha', \beta, \gamma\}$, where α and α' are constants, β is a 1-place function symbol, and γ is a 2-place function symbol, then:

$$\begin{array}{lcll} C_0 & \hat{=} & \langle \gamma\langle -_2, \alpha'\rangle, \alpha, \beta\langle -_1\rangle\rangle & : 2 \rightarrow 3 \\ C_1 & \hat{=} & \langle \alpha, \beta\langle \alpha'\rangle\rangle & : 0 \rightarrow 2 \\ C_0C_1 & = & \langle \gamma\langle \beta\langle \alpha'\rangle, \alpha'\rangle, \alpha, \beta\langle \alpha\rangle\rangle & : 0 \rightarrow 3\ . \end{array}$$

structural congruence: The main decision here is whether to make structural congruence explicit or implicit. The simplest solution (which is the one taken in this paper) is leave it implicit in the definition of arrow equality — thus the arrows are structural equivalence classes of contexts. Consequently, certain categories (such as those of graph contexts, see Section 4 in Part 2) do not have enough universal constructions to give the desired congruence results. In these cases, we are forced to look for the universal constructions in less quotiented categories and then exhibit functors with special properties back to the fully quotiented categories (see Subsection 3.2).

set of reactive contexts: This is modelled by a set $\mathbf{D}$ of arrows. Since reactive contexts are composable and identity contexts are reactive, I take $\mathbf{D}$ to be a subcategory of $\mathbf{C}$. Furthermore, decomposing reactive contexts yields reactive contexts, so $D_1D_0 \in \mathbf{D}$ implies $D_1, D_0 \in \mathbf{D}$.

For example, in the call-by-value λ-calculus [Plo75], the reactive contexts consist of all compositions of the following contexts:

$$- \qquad \mathsf{ap}(v, -) \qquad \mathsf{ap}(-, a)$$

where v is any value (closed abstraction) and a is any term. In the π-calculus considered in [Mil92], the reactive contexts consist of all compositions of following contexts (closed under structural congruence):

$$- \qquad (\nu x)(-) \qquad - \mid a$$

where x is any name and a is any process. Reactive contexts correspond to *evaluation contexts* of Felleisen and Friedman [FF86].

reaction rules: These are given by a set Reacts of redex-contractum pairs of agents (l, r) with common codomain, i.e. $(l, r) \in \mathsf{Reacts}$ implies that there is an object m of $\mathbf{C}$ such that $l, r : 0 \rightarrow m$. For simplicity, I consider redexes and contractums that are pure agents, not agents with meta-variables (i.e. contexts). Thus, to define the reactions of CCS, we let

$$\mathsf{Reacts} \mathrel{\hat{=}} \left\{ \left(\bar{x}.a \mid x.b \ , \ a \mid b \right) \ \ / \ \ x \text{ is a name and } a, b \text{ are agents} \right\}$$

rather than:

$$\mathsf{Reacts} \mathrel{\hat{=}} \left\{ \left(\bar{x}.-_1 \mid x.-_2 \ , \ -_1 \mid -_2 \right) \ \ / \ \ x \text{ is a name} \right\} \ .$$

I use / throughout this paper for set comprehensions. The latter approach maintains the maximum uniformity present in rules and is considered in detail by Sewell in [Sew]; however, that approach is complex and would require future work to adapt it to the categorical setting of this paper (see Section 5 in Part 2).

Distilling the structures described in the past paragraphs yields the following definition of a reactive system and a reaction relation:

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

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

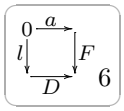
Definition 2.2 (reaction relation) Given a reactive system $\mathbf{C}$, the *reaction relation* $\longrightarrow\!\!\triangleright \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0,m)^2$ contains pairs of agents with common codomain and is defined by lifting the reaction rules through all reactive contexts:

$$a \longrightarrow\!\!\triangleright a' \qquad \text{iff} \qquad \exists (l,r) \in \mathsf{Reacts}, D \in \mathbf{D}. \quad a = Dl \;\&\; a' = Dr\ . \qquad \blacksquare$$

We now have enough definitions to make precise the first approximation for labelled transitions given in (1). The only change is that we think of composing arrows rather than "applying contexts" and we are careful about which contexts are reactive (by writing $D \in \mathbf{D}$ below):

Definition 2.3 (labelled transition — first approximation)

$$\begin{aligned} a \xrightarrow{F}\!\!\triangleright\, a' \qquad &\text{iff} \qquad Fa \longrightarrow\!\!\triangleright\, a' \\ &\text{iff} \qquad \exists (l,r) \in \mathsf{Reacts}, D \in \mathbf{D}. \quad Fa = Dl \;\&\; a' = Dr\ . \end{aligned} \qquad \blacksquare$$

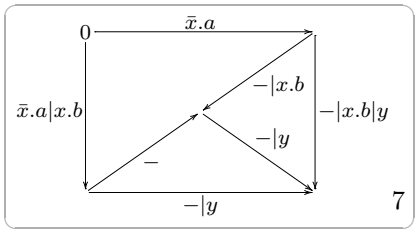


The commuting square to the right renders the equality $Fa = Dl$. As I argued in Subsection 1.2, there may be "junk" in F and D, i.e. parts of F and D that do not contribute to the reaction. For example, in a category of CCS contexts, the outside square in Figure 7 commutes. So, by the naive definition of labelled transitions given above,

$$\bar{x}.a \xrightarrow{-|x.b|y}\!\!\triangleright\, a \mid b \mid y \tag{3}$$

But the y in the label is superfluous. Is there a general condition on Figure 6 that prevents this labelled transition, but still allows the following:

$$\bar{x}.a \xrightarrow{-|x.b}\!\!\triangleright\, a \mid b \quad ?$$



Informally, the condition would state that there is no lesser upper bound in Figure 6 for a, l than F, D. In Figure 7 there clearly *is* a lesser upper bound, as illustrated by the triple of arrows inside the square. In the following subsections I render this condition in terms of categorical constructions and incorporate it in a new definition of labelled transitions. I then show that strong bisimulation is a congruence with respect to this labelled transition relation. A variety of preorders and other operational equivalences are discussed in the next section.

2.2 Categorical basis for contextual labels

The goal of this subsection is to find a tractable definition of a labelled transition relation, one which readily leads to congruential equivalences and preorders and moreover facilitates proofs concerning these relations. Intuitively, the labels represent just the information exchanged between an agent and its environment in order to make a reaction.

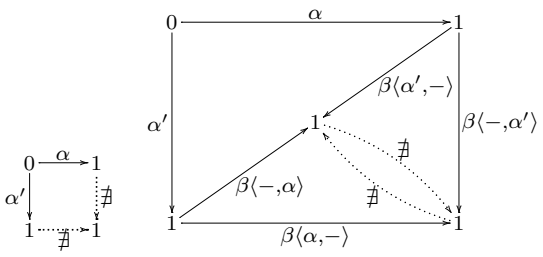
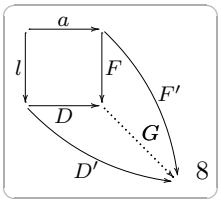


Figure 9: Non-existence of pushouts



Following the intuitions of the previous subsections concerning Figure 6, the natural question is as follows. How can F, D be forced to contain no "junk"? A possible solution is to require that F and D are a "least upper bound" for a and l. The typical way to formulate this is to state that the square in Figure 8 is a *pushout*, i.e. has the property: $Fa = Dl$, and for every F' and D' satisfying $F'a = D'l$ there exists a unique G such that $GF = F'$ and $GD = D'$, as shown here.

Unfortunately, pushouts rarely exist in the categories that interest us. Consider, for example, a category of term contexts over a signature Σ; its objects consist of 0 and 1; its arrows $0 \longrightarrow 1$ are terms over Σ; its arrows $1 \longrightarrow 1$ are one-hole contexts over Σ; there are no arrows $1 \longrightarrow 0$ and exactly one arrow $\mathsf{id}_0 : 0 \longrightarrow 0$. Now, if Σ contains only constant symbols, say $\Sigma = \{\alpha, \alpha'\}$, then there is no pushout completing Figure 9(1) because there are no contexts other than the identity. If we introduce a 2-place function symbol β into Σ, we can construct an upper bound for α and α' but still no pushout (Figure 9(2)).

A more refined approach is to assert that F and D are a "minimal upper bound" — informally, an upper bound for which there are no lesser upper bounds. Before defining this notion in terms of *idem pushouts* (IPOs), I give a more basic construction, namely that of *relative pushouts* (RPOs). The latter, unlike pushouts, exist in many categories of agent contexts.

The plan for the rest of this subsection is to develop a sequence of propositions that will serve as a basis for the proofs of congruence by categorical reasoning given in this paper.

Because RPOs and IPOs are categorical constructions independent of reactive systems, I shall work in this subsection with an arbitrary category $\mathbf{C}$ whose arrows and objects I denote by f, g, h, k, x, y, z and m, n; in pictures I omit labels on the objects when possible.

Definition 2.4 (RPO) In any category $\mathbf{C}$, consider a commuting square (Figure 10(1)) consisting of $g_0 f_0 = g_1 f_1$. An *RPO* is a triple h_0, h_1, h satisfying two properties:

commutation: $h_0 f_0 = h_1 f_1$ and $h h_i = g_i$ for $i = 0, 1$ (Figure 10(2));

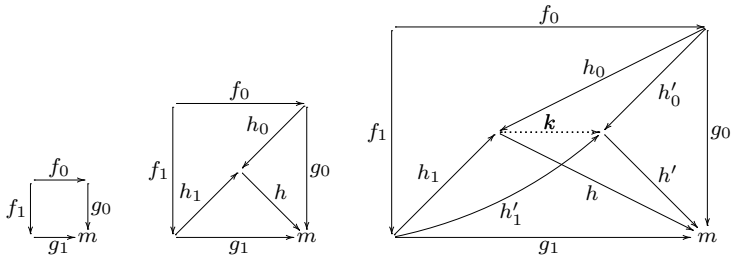


Figure 10: Construction of an RPO

universality: for any h'_0, h'_1, h' satisfying $h'_0 f_0 = h'_1 f_1$ and $h' h'_i = g_i$ for $i = 0, 1$, there exists a unique mediating arrow k such that $h'k = h$ and $kh_i = h'_i$ (Figure 10(3)). ■

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

An RPO for Figure 10(1) is just a pushout in the slice category of $\mathbf{C}$ over m. Thus an RPO is a standard combination of categorical constructions — though it is not commonly used in category theory and its application to reactive systems is novel.

In Part 2, I illustrate the existence of RPOs for categories of graphs. For concreteness, though, it is worth examining now the example of an RPO and another candidate shown in Figure 11. The arrows are in a category of term algebra contexts over the signature $\{\alpha, \beta, \gamma, \delta\}$, where α is a constant and β, γ, δ are 1-place function symbols. The RPO triple $-, \beta\langle - \rangle, \delta\langle \gamma\langle - \rangle\rangle$ adds just the minimal extra bit of context $\beta\langle - \rangle$ to α in order to get an upper bound for $\beta\langle\alpha\rangle$ and α; the arrow $\delta\langle\gamma\langle - \rangle\rangle$ then provides the extra junk necessary to recover the upper bound provided by the surrounding square. The reader may enjoy checking that the candidate triple $\gamma\langle - \rangle, \gamma\langle\beta\langle - \rangle\rangle, \delta\langle - \rangle$ is the only other non-trivial one possible and that the mediating dotted arrow $\gamma\langle - \rangle$ is unique.

A square is called an IPO if it has an RPO of a special kind:

Definition 2.5 (IPO) The commuting square in Figure 10(1) is an IPO if the triple g_0, g_1, id_m is an RPO. ■

The difference between a pushout and an IPO is clearest in a partial order category: a pushout is a least upper bound (i.e. less than any other upper bound) and an IPO is a minimal upper bound (i.e. not greater than any other upper bound). IPOs form the basis of our abstract definition of labelled transition and their existence follows from that of RPOs as shown by the following lemma:

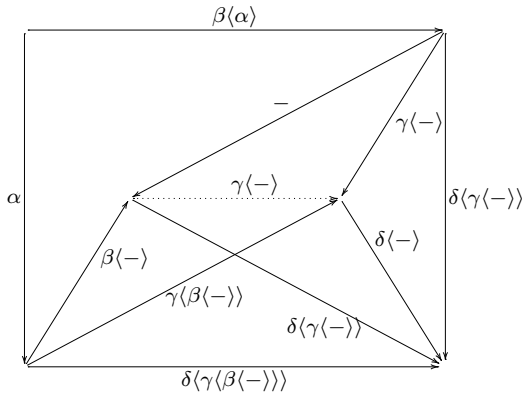
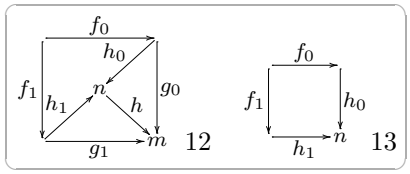


Figure 11: An example of an RPO and another candidate

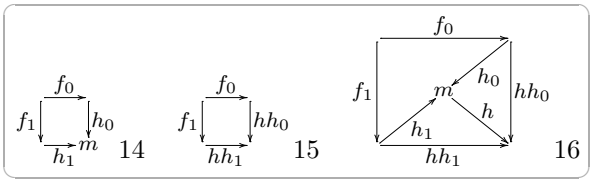
Proposition 2.6 (IPOs from RPOs) If Figure 12 is an RPO diagram then the square in Figure 13 is an IPO.



Proof Arrow chasing. See Proposition 2.7 in [Lei01]. ■

The next result provides a partial converse to the previous proposition. It serves as a key part of the proof of IPO pasting which comes afterwards:

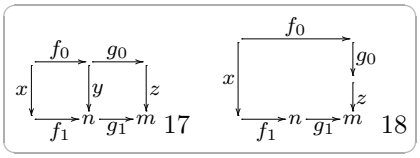
Proposition 2.7 (RPOs from IPOs) If Figure 14 is an IPO and Figure 15 has an RPO then Figure 16 is an RPO.



Proof Arrow chasing. See Proposition 2.8 in [Lei01]. ■

IPOs can be pasted together as shown by the following proposition, which is analogous to the standard pasting result for pushouts.

Proposition 2.8 (IPO pasting) Suppose that both squares in Figure 17 commute and that Figure 18 has an RPO. Then the following properties hold of Figure 17:



1. If the two squares are IPOs then so is the big rectangle.

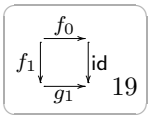
2. If the big rectangle and the left square are IPOs then so is the right square.

Proof Proposition 2.7 and arrow chasing. See Proposition 2.9 in [Lei01]. ■

Finally, I conclude this collection of categorical results with two concerning IPOs; they are not immediately relevant to this section, but play an important role, respectively, in Proposition 3.15 and in Proposition 3.6 of Part 2.

The first shows how IPOs can arise from epis.

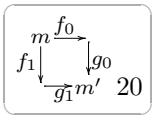
Proposition 2.9 (IPOs from epis) Suppose f_1 is an epi. Then the outer square in Figure 19 is an IPO.



Proof Arrow chasing. See Proposition 2.11 in [Lei01]. ■

The second asserts that only a subcategory of $\mathbf{C}$ plays any role in characterising that a particular square is an IPO.

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



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

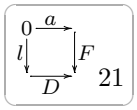
Suppose the square in Figure 20 commutes, where $f_i, g_i \in \mathbf{C}'$ for $i = 0, 1$. Then the square is an IPO in $\mathbf{C}$ iff it is an IPO in $\mathbf{C}'$.

Proof The only arrows relevant to the square being an IPO in $\mathbf{C}$ are contained in $\mathbf{C}'$. ■

2.3 Labelled transitions and congruence for strong bisimulation

The category theory developed in the previous subsection provides the machinery needed in this subsection to accomplish two aims. The first is to improve the unsatisfactory definition of labelled transition given earlier (Definition 2.3). The second is to prove that strong bisimulation over the new labelled transitions is a congruence. I return to the notations of Subsection 2.1, using $\mathbf{C}$ for a reactive system, with $a, b \in \mathbf{C}$ ranging over arrows with domain 0 (agents) and $C, D, F \in \mathbf{C}$ ranging over arbitrary arrows (contexts).

The new version of labelled transitions is a modification of the approximation given by Definition 2.3, where the condition $Fa = Dl$ is strengthened to require that the square in Figure 21 is an IPO:



Definition 2.11 (labelled transition) $a \xrightarrow{F}\!\triangleright\ a'$ iff there exists $(l, r) \in \mathsf{Reacts}$ and $D \in \mathbf{D}$ such that Figure 21 is an IPO and $a' = Dr$. ■

This definition assures that F, D provides a minimal upper bound on a and l, as required in Subsection 2.1. For suppose there is another upper bound F', D', i.e. $F'a = D'l$, and also $F = RF'$ and $D = RD'$ for some R. Then the IPO property for Figure 21 ensures that for some R' (with $RR' = \mathsf{id}$) we have $F' = R'F$ and $D' = R'D$ — so F, D provides a "lesser" upper bound than F', D' after all.

Proposition 2.12 For all contexts F we have that $a \xrightarrow{F}\!\triangleright\ a'$ implies $Fa \longrightarrow\!\triangleright\ a'$. ■

The converse fails in general (which is good, given the remarks made after Definition 2.3 about the first approximation for labelled transitions). I return to the converse property later in Subsection 3.4 in the special case that F is an iso. Strong bisimulation over $\dot{\longrightarrow\!\triangleright}$ follows its usual scheme [Par81]:

Definition 2.13 (strong bisimulation over $\dot{\longrightarrow\!\triangleright}$) Let $\mathcal{S} \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0,m)^2$ be a relation that contains pairs of agents with common codomain. $\mathcal{S}$ is a *simulation* over $\dot{\longrightarrow\!\triangleright}$ iff $\mathcal{S}$ satisfies the following property for all $(a,b) \in \mathcal{S}$: if $a \xrightarrow{F}\!\triangleright\, a'$ then there exists b' such that $b \xrightarrow{F}\!\triangleright\, b'$ and $(a',b') \in \mathcal{S}$. $\mathcal{S}$ is a *strong bisimulation* iff $\mathcal{S}$ and $\mathcal{S}^{-1}$ are strong simulations. Let $\sim$ be the *largest strong bisimulation* over $\dot{\longrightarrow\!\triangleright}$. ■

I now state and prove the congruence result for strong bisimulation, one of the central results of this paper: if $\mathbf{C}$ has a sufficiently rich collection of RPOs then $\sim$ is a congruence.

Definition 2.14 ($\mathbf{C}$ has all redex-RPOs) Say that $\mathbf{C}$ *has all redex-RPOs* if for all $(l,r) \in \mathsf{Reacts}$ and arrows a, F, D such that $D \in \mathbf{D}$ and $Fa = Dl$, the square in Figure 21 has an RPO. ■

Theorem 2.15 (congruence for $\sim$) Let $\mathbf{C}$ be a reactive system which has all redex-RPOs. Then $\sim$ is a congruence, i.e. $a \sim b$ implies $Ca \sim Cb$ for all $C \in \mathbf{C}$ with required domain.

Proof By symmetry, it is sufficient to show that the following relation is a strong simulation:

$$\mathcal{S} \hat{=} \{(Ca, Cb) \;/\; a \sim b \text{ and } C \in \mathbf{C}\} \,.$$

The proof falls into three parts, each of which is an implication as illustrated in Figure 22(1). Dashed lines connect pairs of points contained within the relation annotating the line. Each arrow " $\Downarrow$ " is tagged by the part of the proof below that justifies the implication. Suppose that $a \sim b$ and $C \in \mathbf{C}$, and thus $(Ca, Cb) \in \mathcal{S}$.

(i): If $Ca \xrightarrow{F}\!\triangleright\, a'$ then, by definition, there exists $(l,r) \in \mathsf{Reacts}$ and $D \in \mathbf{D}$ such that the big rectangle in Figure 22(2) is an IPO and $a' = Dr$. Because $\mathbf{C}$ has all redex-RPOs, there exists F', D', C' forming an RPO as in Figure 22(2); moreover, $D', C' \in \mathbf{D}$ since $C'D' = D \in \mathbf{D}$. By Proposition 2.6, Figure 22(3) is an IPO. Because $\mathbf{C}$ has all redex-RPOs, Proposition 2.8 implies that Figure 22(4) is an IPO too. By definition, $a \xrightarrow{F'}\!\triangleright\, D'r$ and $a' = C'D'r$.

(ii): Since $a \sim b$, there exists b'' such that $b \xrightarrow{F'}\!\triangleright\, b''$ and $D'r \sim b''$. By definition there exists $(l',r') \in \mathsf{Reacts}$ and $E' \in \mathbf{D}$ such that Figure 22(5) is an IPO and $b'' = E'r'$.

(iii): Because $\mathbf{C}$ has all redex-RPOs, Proposition 2.8 implies that we can paste Figure 22(5) with Figure 22(4) (both IPOs) along F' and conclude that Figure 22(6) is an IPO. Hence $Cb \xrightarrow{F}\!\triangleright\, C'E'r'$ and $(C'D'r, C'E'r') \in \mathcal{S}$ because $D'r \sim E'r'$, as desired. ■

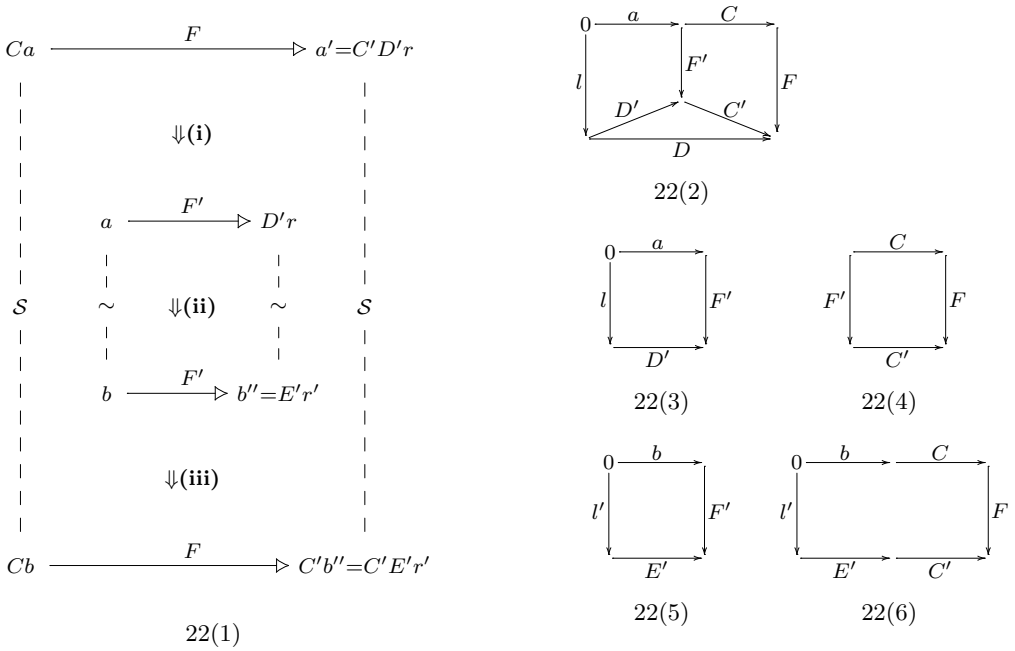


Figure 22: Congruence proof for strong bisimulation

The crux of the above proof is that Figure 22(4), which mediates between an F'-labelled transition of a and an F-labelled transition of Ca, is "portable", i.e. can be pasted onto a new diagram, serving the same function for b and Cb. This essential idea appears to be robust under variation both of the definition of labelled transition and of the congruence being established. Many examples are shown in Section 3.

We can isolate precisely in two lemmas how such portable IPO squares are cut and then pasted. These lemmas are just pieces of the congruence proof above, but their factorisation from the main proof greatly simplifies the latter and lays the ground for tractable presentations of more difficult congruences results in the next section.

The first lemma shows that portable IPO squares arise when a composite agent has a labelled transition:

Lemma 2.16 (portable IPO cutting) If $\mathbf{C}$ has all redex-RPOs then the following inference rule holds:

$$\frac{Ca \xrightarrow{F} a'}{\exists\ a'' \text{ and an IPO } \begin{array}{ccc} \cdot & \xrightarrow{C} & \cdot \\ {\scriptstyle F'}\downarrow & & \downarrow{\scriptstyle F} \\ \cdot & \xrightarrow[C']{} & \cdot \end{array}. \qquad a \xrightarrow{F'} a'' \qquad a' = C'a'' \qquad C' \in \mathbf{D}}.$$

■

The second shows how to "paste" a portable square to gain a labelled transition of a composite agent:

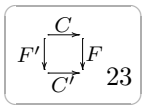
Lemma 2.17 (portable IPO pasting) If $\mathbf{C}$ has all redex-RPOs then the following inference rule holds:

$$\frac{\begin{array}{c}C\\ F'\square F\\ C'\end{array} \text{ is an IPO} \qquad a \xrightarrow{F'} a'' \qquad C' \in \mathbf{D}}{Ca \xrightarrow{F} C'a''} \ .$$

■

We can now replay the proof of Theorem 2.15 in a more concise form by employing these lemmas:

(i): If $Ca \xrightarrow{F} a'$ then by Lemma 2.16, there exists a'' and an IPO square shown in Figure 23 such that



23

$$a \xrightarrow{F'} a'' \qquad a' = C'a'' \qquad C' \in \mathbf{D} \ .$$

(ii): Since $a \sim b$, there exists b'' such that $b \xrightarrow{F'} b''$ and $a'' \sim b''$.

(iii): Since Figure 23 is an IPO and $C' \in \mathbf{D}$, Lemma 2.17 implies that $Cb \xrightarrow{F} C'b''$. Also, $a'' \sim b''$ implies $(C'a'', C'b'') \in \mathcal{S}$, as desired.

Let us return to Lemma 2.16 in order to expose an odd property. The Lemma contradicts the situation in many process calculi: normally, $Ca \xrightarrow{F}$ *does not necessarily imply* that a has any labelled transitions. In CCS, for example, $\mathbf{0} \mid x \xrightarrow{x}$ (using the traditional CCS non-contextual labels) but $\mathbf{0}$ has no transitions. As a result, in typical proofs of congruence for strong bisimulation, two cases are distinguished when considering the transition $Ca \xrightarrow{F}$ (using the notation of this section):

- C and F together conspire to create the transition without reference to a. In this case $Cb \xrightarrow{F}$ holds without using the assumption that $a \sim b$. (For example, see "Case 2" on p. 98 of [Mil88].)
- Or, a, C, and F together conspire to create the transition, as in part **(i)** above.

Recasting the CCS example in terms of contextual labels, we have that $\mathbf{0} \mid x \xrightarrow{-|\bar{x}}$. But then there *is* a contextual transition for $\mathbf{0}$, namely $\mathbf{0} \xrightarrow{-|x|\bar{x}}$, though not a satisfactory one: the label provides the entire redex, without any contribution from $\mathbf{0}$. This is attributable to a defect in the definition of contextual labelled transitions: if $a \xrightarrow{F}$, the IPO property requires that F contain parts of the relevant redex and no extra junk, but does not prevent F from containing *all* of the redex.

It is by enriching the categorical structure to express multi-hole contexts (see Subsection 3.8) that we eliminate this defect of transitions. When we do, exactly the same case analysis (shown above) is carried out when proving congruence.

3 Further congruence results

3.1 Introduction

This section generalises the definition of labelled transition given in the previous section and provides congruence proofs for additional equivalences and preorders. These include weak bisimulation, the traces preorder, and the failures preorder. I will describe them in turn as each congruence proof is presented.

The more important step, though, is the generalisation of reactive systems and labelled transitions. In the previous section, the central hypothesis required in the proof of congruence for strong bisimulation is that the reactive system $\mathbf{C}$ has all redex-RPOs. This section addresses the problem of what to do if $\mathbf{C}$ does not possess sufficient RPOs. Such a situation arises when considering, for example, a category of graph contexts (see Subsection 4.3 in Part 2). Roughly, the lack of RPOs is attributable to the absence of enough intensional information about the occurrence of nodes: it is ambiguous which node in a context corresponds to a node in the composition of the context with another. Thus if $C_0B_0 = C_1B_1$, it is ambiguous which nodes are common to both C_0 and C_1 and thus impossible to choose the context to be factored off when constructing an RPO.

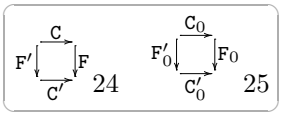
What can be done when there are not enough RPOs in a reactive system? In general, it is not a good solution simply to enrich the reactive system to force it to have enough RPOs. The enrichment could yield a category with too much intensional information. For example, the enrichment considered for graph contexts (Subsection 4.6 in Part 2) forces arrows with the same domain and codomain to have the same number of nodes. Since the definition of strong bisimulation requires that $a \sim b$ implies that $a, b : 0 \rightarrow m$ for some object m, the strong bisimulation relation could only compare arrows of the same number of nodes. Such a restriction is unacceptable because strong bisimulation should include pairs of agents with widely differing static structure.

The solution presented in this section is to accommodate two categories $\hat{\mathbf{C}}$ and $\mathbf{C}$ related by a functor:

$$\begin{array}{c} \hat{\mathbf{C}} \\ \downarrow \mathcal{F} \\ \mathbf{C} \end{array} .$$

The idea is that $\mathbf{C}$ is a reactive system, whose arrows correspond to the agents and agent contexts; $\mathbf{C}$ does not necessarily have enough RPOs. Sitting "above" $\mathbf{C}$ is $\hat{\mathbf{C}}$, a category with sufficient RPOs. The definition of the labelled transition *relates* arrows in $\mathbf{C}$, just as in the previous section: i.e. $a \xrightarrow{F} a'$ is defined for a, a' agents in $\mathbf{C}$ and F an agent context of $\mathbf{C}$. But, by contrast with the previous section, the definition is given in terms of the existence of an IPO "upstairs" in $\hat{\mathbf{C}}$. (If $\hat{\mathbf{C}} = \mathbf{C}$ and $\mathcal{F}$ is the identity functor, then the new definition of this section collapses into the old one given in the previous.)

Thus we get the best of both worlds: the agents whose labelled transition behaviour we consider need not contain any superfluous intensional data; as long as we can construct a more intensional category above containing sufficient RPOs and a suitable functor, then *we can get congruence results downstairs.*



These congruence results require the functor $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ to satisfy certain properties. Most are trivial but one is interesting, namely that $\mathcal{F}$ *allows IPO sliding.* Recall from the previous section that the crux of the congruence proof for strong bisimulation was the portability of the IPO square that related F'-transitions of an agent to F-transitions of C applied to the agent. This square was cut off when passing from $Ca \xrightarrow{F}\triangleright$ to $a \xrightarrow{F'}\triangleright$ and then pasted back on when passing from $b \xrightarrow{F'}\triangleright$ to $Cb \xrightarrow{F}\triangleright$. In the new definition of labelled transition considered in this section, the pasting operation is more complex. The portable square, e.g. Figure 24, now lives in $\hat{\mathbf{C}}$ (the upstairs category) and its left leg is $\mathtt{F}'$, some arrow for which $\mathcal{F}(\mathtt{F}') = F'$. (Teletype font is used for arrows in $\hat{\mathbf{C}}$.) However, the transition $b \xrightarrow{F'}\triangleright$ is justified by an IPO square upstairs whose right-leg is $\mathtt{F}'_0$, an arrow in the preimage of F' not necessarily equal to $\mathtt{F}'$. Thus Figure 24 cannot be pasted without first sliding it to a new IPO square, e.g. Figure 25, whose left-leg is $\mathtt{F}'_0$ and whose $\mathcal{F}$-image is kept invariant. The present section assumes that $\mathcal{F}$ allows IPO sliding; Section 3 in Part 2 proves that this is the case when $\mathcal{F}$ is of a certain general form.

The outline of this section is as follows. In the next subsection, I define the notion of a functorial reactive system, giving precise requirements for $\mathcal{F}$. Then I define the reaction and labelled transition relations. In the following subsection, I prove some results about portable IPO squares that are direct analogies to those (Lemma 2.16 and Lemma 2.17) at the end of the previous section. The main subsections are concerned with a series of congruence proofs for strong bisimulation, weak bisimulation, the traces preorder, and the failures preorder. The final subsection treats a richer notion of functorial reactive system with arrows corresponding to multi-hole contexts and shows that strong bisimulation is indeed a congruence here as well.

3.2 Functorial reactive systems

The first part of the setup is to define precisely the notion of functorial reactive system, which was introduced informally above. Its definition rests on that of a reactive system, given in the previous section, which we recall here first for ease of reference:

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

- a distinguished object 0 (not necessarily initial);

- a set of *reaction rules* called $\mathsf{Reacts} \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0,m)^2$, a relation containing pairs of agents with common codomain;
- a subcategory $\mathbf{D}$ of $\mathbf{C}$, whose arrows are the *reactive contexts*, with the property that $D_1 D_0 \in \mathbf{D}$ implies $D_1, D_0 \in \mathbf{D}$. ■

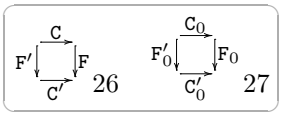
Definition 3.1 (functorial reactive system) Let $\mathbf{C}$ be a reactive system. A *functorial reactive system over* $\mathbf{C}$ consists of a functor $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ which maps a distinguished object $\boldsymbol{\varepsilon} \in \mathsf{obj}\,\hat{\mathbf{C}}$ to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

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

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

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

$\mathcal{F}$ **allows IPO sliding:** for any IPO square as in Figure 26 and any arrow $\mathtt{F}'_0$ with $\mathcal{F}(\mathtt{F}'_0) = \mathcal{F}(\mathtt{F}')$ there exist $\mathtt{C}_0, \mathtt{C}'_0, \mathtt{F}_0$ forming an IPO square as in Figure 27 with



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

Throughout this section, I use uppercase teletype characters to denote arrows in $\hat{\mathbf{C}}$ and lowercase teletype characters ($\mathtt{a}, \mathtt{l}, \ldots$) to denote arrows with domain $\boldsymbol{\varepsilon}$ in $\hat{\mathbf{C}}$.

The $\mathcal{F}$ images of these are agents in $\mathbf{C}$. The special domain requirement of $\mathtt{a}, \mathtt{l}, \ldots$ is left tacit throughout this section: thus $(\exists \mathtt{l} \in \hat{\mathbf{C}}.\ \ldots)$ means $(\exists \mathtt{l} \in \hat{\mathbf{C}}.\ \mathsf{Dom}\,\mathtt{l} = \boldsymbol{\varepsilon}\ \&\ \ldots)$.

The definition of the reaction relation is identical to the one given earlier:

Definition (reaction relation ($\longrightarrow\!\!\triangleright$) recalled; cf. Definition 2.2) Given a functorial reactive system $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$, the *reaction relation* $\longrightarrow\!\!\triangleright \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0,m)^2$ contains pairs of agents with common codomain and is defined by lifting the reaction rules through all reactive contexts: $a \longrightarrow\!\!\triangleright a'$ iff there exists $D \in \mathbf{D}$ and $(l, r) \in \mathsf{Reacts}$ such that $a = Dl$ and $a' = Dr$. ■

This definition has an alternative characterisation given by the following result:

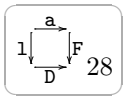
Proposition 3.2 (characterisation of $\longrightarrow\!\!\triangleright$) $a \longrightarrow\!\!\triangleright a'$ iff there exist $\mathtt{a}, \mathtt{l}, \mathtt{D} \in \hat{\mathbf{C}}$ and $r \in \mathbf{C}$ such that $\mathtt{a} = \mathtt{Dl}$ and

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

Proof Follows immediately because $\mathcal{F}$ lifts agents and creates compositions. ■

We now turn to the definition of labelled transition. As stated earlier, $\dot{\longrightarrow}$ is a ternary relation whose arguments are all arrows in $\mathbf{C}$. The original requirement that a particular square be an IPO in $\mathbf{C}$ (see Definition 2.11) is replaced here by requiring that there exist a preimage of this square that is an IPO in $\hat{\mathbf{C}}$:

Definition 3.3 (labelled transition ($\dot{\longrightarrow}$); cf. Definition 2.11) $a \xrightarrow{F} a'$ iff there exist $\mathtt{a}, \mathtt{l}, \mathtt{F}, \mathtt{D} \in \hat{\mathbf{C}}$ and $r \in \mathbf{C}$ such that Figure 28 is an IPO in $\hat{\mathbf{C}}$ and



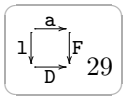
28

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

Notice that a', the RHS of the transition, is required to be $\mathcal{F}(\mathtt{D})r$, and *not* $\mathcal{F}(\mathtt{Dr})$ for some $\mathtt{r}$ with $\mathcal{F}(\mathtt{r}) = r$. This is important since it allows the reaction rules Reacts to contain pairs (l, r) for which $\mathcal{F}$-preimages of l and r might not have common codomains.

By analogy with Definition 2.14, we can define when a functorial reactive system $\mathcal{F}$ has all redex-RPOs; the primary difference is that here the RPOs exist upstairs in $\hat{\mathbf{C}}$, not downstairs in $\mathbf{C}$:

Definition 3.4 ($\mathcal{F}$ has all redex-RPOs; cf. Definition 2.14) A functorial reactive system $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ *has all redex-RPOs* if any square, such as in Figure 29, has an RPO, provided that $\mathcal{F}(\mathtt{D}) \in \mathbf{D}$ and that there exists $r \in \mathbf{C}$ such that $(\mathcal{F}(\mathtt{l}), r) \in \mathsf{Reacts}$.



29

■

Note that we do not demand that *all* RPOs exist, just ones for which the left-leg of the enclosing square is a preimage of a redex and the bottom leg is a preimage of an arrow in $\mathbf{D}$. (This narrowing of the definition is exactly analogous to what happens in Definition 2.14.)

3.3 Cutting and pasting portable IPO squares

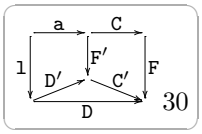
This subsection replays the results at the end of the previous section which show how to cut and paste portable IPO squares. The main difference lies in the proof of pasting: here we make explicit use of the assumption that $\mathcal{F}$ allows IPO sliding.

The first result shows how the transitions of composite agents yield IPO squares:

Lemma 3.5 (portable IPO cutting; cf. Lemma 2.16) Suppose $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ is a functorial reactive system and has all redex-RPOs. The following inference rule holds:

$$\frac{Ca \xrightarrow{F} a'}{\exists\ a'' \in \mathbf{C} \text{ and an IPO square } \begin{smallmatrix}\mathtt{C}\to\\ \mathtt{F}'\downarrow\ \ \downarrow\mathtt{F}\\ \to\mathtt{C}'\end{smallmatrix}.\quad \left(\begin{array}{c} a \xrightarrow{\mathcal{F}(\mathtt{F}')} a'' \qquad a' = \mathcal{F}(\mathtt{C}')a'' \qquad \mathcal{F}(\mathtt{C}') \in \mathbf{D} \\ \mathcal{F}(\mathtt{C}) = C \qquad \mathcal{F}(\mathtt{F}) = F \end{array}\right)}\ .$$

Proof By the definition of $\xrightarrow{F}\triangleright$ and the hypothesis that $\mathcal{F}$ creates compositions, there exists $\mathtt{a}, \mathtt{C}, \mathtt{l}, \mathtt{F}, \mathtt{D} \in \hat{\mathbf{C}}$ and $r \in \mathbf{C}$ such that the big rectangle in Figure 30 is an IPO and



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

Because $\mathcal{F}$ has all redex-RPOs, there exist $\mathtt{F}', \mathtt{D}', \mathtt{C}'$ forming an RPO in $\hat{\mathbf{C}}$, as in Figure 30. Then $\mathcal{F}(\mathtt{C}') \in \mathbf{D}$ since $\mathcal{F}(\mathtt{C}')\mathcal{F}(\mathtt{D}') = \mathcal{F}(\mathtt{D}) \in \mathbf{D}$. By Proposition 2.6, the small left-hand square of Figure 30 is an IPO. Because $\mathcal{F}$ has all redex-RPOs, Proposition 2.8 implies that the small right-hand square is an IPO too. By definition, $a \xrightarrow{\mathcal{F}(\mathtt{F}')}\triangleright a''$ and $a' = \mathcal{F}(\mathtt{C}')a''$ where $a'' \mathrel{\hat{=}} \mathcal{F}(\mathtt{D}')r$. ■

The next result shows how the reactions of composite agents can be decomposed. There is no analogue of this result the previous section since it is not needed in the congruence proof for strong bisimulation.

Lemma 3.6 (portable IPO cutting for reactions) Suppose $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ is a functorial reactive system and has all redex-RPOs. The following inference rule holds:

$$\frac{Ca \longrightarrow\triangleright a'}{\exists\ a'' \in \mathbf{C} \text{ and } \mathtt{C}', \mathtt{F}' \in \hat{\mathbf{C}}.\quad \left(a \xrightarrow{\mathcal{F}(\mathtt{F}')}\triangleright a'' \qquad a' = \mathcal{F}(\mathtt{C}')a'' \qquad \mathcal{F}(\mathtt{C}') \in \mathbf{D} \qquad \mathcal{F}(\mathtt{C}'\mathtt{F}') = C\right)}\ .$$

Moreover, if $\mathtt{F}'$ is an iso in the conclusion then it is equal to id.

Proof See Lemma 3.6 in [Lei01]. ■

The final result shows how to paste a portable IPO square in order to gain a transition for a composite agent. As stated above, this is where IPO sliding is used:

Lemma 3.7 (portable IPO pasting; cf. Lemma 2.17) Suppose $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ is a functorial reactive system and has all redex-RPOs. The following inference rule holds:

$$\frac{\begin{array}{c}\mathtt{C}\\ \mathtt{F}'\ \square\ \mathtt{F}\\ \mathtt{C}'\end{array} \text{ is an IPO} \qquad a \xrightarrow{\mathcal{F}(\mathtt{F}')}\triangleright a' \qquad \mathcal{F}(\mathtt{C}') \in \mathbf{D}}{\mathcal{F}(\mathtt{C})a \xrightarrow{\mathcal{F}(\mathtt{F})}\triangleright \mathcal{F}(\mathtt{C}')a'}\ .$$

Proof See Lemma 3.7 in [Lei01]. ■

That concludes the setup for functorial reactive systems. The rest of the section is devoted to a sequence of congruence proofs.

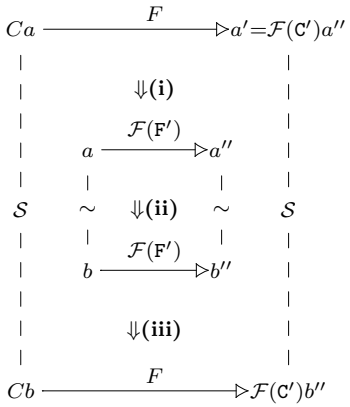


Figure 31: Schema of the congruence proof for $\sim$

3.4 Strong bisimulation

This subsection proves that strong bisimulation is a congruence for a functorial reactive system. The definition is straightforward:

Definition 3.8 (strong bisimulation over $\dot{\longrightarrow}$; cf. Definition 2.13) Let $\sim$ be the largest strong bisimulation over $\dot{\longrightarrow}$. ■

The proof of congruence is almost identical to the one presented at the end of the previous section: the only difference is that the updated IPO cutting and pasting results for functorial reactive systems are substituted for the old ones.

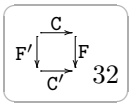
Theorem 3.9 (congruence for $\sim$) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\sim$ is a congruence, i.e. $a \sim b$ implies $Ca \sim Cb$ for all $C \in \mathbf{C}$ of the required domain.

Proof By symmetry, it is sufficient to show that the following relation is a strong simulation:

$$\mathcal{S} \hat{=} \{(Ca, Cb) \ / \ a \sim b \text{ and } C \in \mathbf{C}\} \ .$$

The proof falls into three parts, each of which is an implication as illustrated in Figure 31. Dashed lines connect pairs of points contained within the relation annotating the line. Each arrow "⇓" is tagged by the part of the proof below that justifies the implication. Suppose that $a \sim b$ and $C \in \mathbf{C}$, and thus $(Ca, Cb) \in \mathcal{S}$.

(i): If $Ca \xrightarrow{F} a'$ then by Lemma 3.5, there exist $a'' \in \mathbf{C}$ and an IPO square shown in Figure 32 such that $a \xrightarrow{\mathcal{F}(\mathtt{F}')} a''$ and



32

$$a' = \mathcal{F}(\mathtt{C}')a'' \qquad \mathcal{F}(\mathtt{C}') \in \mathbf{D}$$
$$\mathcal{F}(\mathtt{C}) = C \qquad \mathcal{F}(\mathtt{F}) = F \ .$$

(ii): Since $a \sim b$, there exists b'' such that $b \xrightarrow{\mathcal{F}(\mathtt{F}')} b''$ and $a'' \sim b''$.

(iii): Since Figure 32 is an IPO and $\mathcal{F}(\mathtt{C}') \in \mathbf{D}$, Lemma 3.7 implies that $Cb \xrightarrow{F} \mathcal{F}(\mathtt{C}')b''$. Also, $a'' \sim b''$ implies $(\mathcal{F}(\mathtt{C}')a'', \mathcal{F}(\mathtt{C}')b'') \in \mathcal{S}$, as desired. ■

In most process calculi the reaction relation and the τ-labelled transition relation coincide. See, for example Proposition 5.2 in [Mil92] and Theorem 2 in [Sew00]. A τ transition is a "silent move": a transition that takes place without interacting with the external environment. Intuitively, a τ-labelled transition corresponds to an id-labelled transition when using contexts as labels, i.e. $a \xrightarrow{\mathsf{id}} a'$ iff the environment need only supply a vacuous identity context in order to enable a to react. However, if we look carefully at the definition of labelled transition given in Definition 3.3 and the characterisation of reaction in Proposition 3.2, we see that $\xrightarrow{\mathsf{id}}$ and $\longrightarrow$ are not necessarily identical. There is an implication in one direction, namely $\xrightarrow{\mathsf{id}} \subseteq \longrightarrow$, since every IPO square is also a commuting square, but not necessarily the converse. Indeed, Figure 3 contains a non-IPO commuting square whose outermost legs on the right are both identity arrows.

In the special situation when all preimages of redexes are epis, $\xrightarrow{\mathsf{id}}$ and $\longrightarrow$ *do* coincide, thanks to Proposition 2.9. This situation is explored at the end of this subsection in Proposition 3.15.

Before taking the epi hypothesis on board let us a consider an alternate definition of labelled transition for which we do recover the reaction relation:

Definition 3.10 (labelled transition by cases ($\xrightarrow[\mathsf{c}]{\cdot}$))

$$a \xrightarrow[\mathsf{c}]{F} a' \quad \text{iff} \quad \begin{cases} Fa \longrightarrow a' & \text{if } F \text{ is an iso} \\ a \xrightarrow{F} a' & \text{if } F \text{ is not an iso} \ . \end{cases}$$

■

It follows immediately from the definition that $a \xrightarrow[\mathsf{c}]{\mathsf{id}} a'$ iff $a \longrightarrow a'$. Furthermore, the induced strong bisimulation (defined next) is a congruence, as shown below. It is worth considering whether there are other definitions that recover the reaction relation but do *not* involve case analysis. This point is taken up in Appendix B in [Lei01] where I present a definition of labelled transition involving retractions that satisfies this requirement; I show furthermore that this definition induces a congruence which includes $\sim_{\mathsf{c}}$. Let us return to the main flow of the argument now and consider $\sim_{\mathsf{c}}$ in detail.

Definition 3.11 (strong bisimulation over $\dot{\longrightarrow}_c$) Let $\sim_c$ be the largest strong bisimulation over $\dot{\longrightarrow}_c$. ■

Theorem 3.12 (congruence for $\sim_c$) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\sim_c$ is a congruence, i.e. $a \sim_c b$ implies $Ca \sim_c Cb$ for all $C \in \mathbf{C}$ of the required domain.

Proof See Theorem 3.12 in [Lei01]. ■

Because $\dot{\longrightarrow}$ and $\dot{\longrightarrow}_c$ are so closely related, it is not surprising that the induced congruences are also related. Indeed the following result shows that $\sim_c$ is a coarser equivalence than $\sim$. It is an open question whether there exists a functorial reactive system for which the inequality is strict.

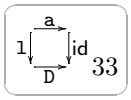
Proposition 3.13 ($\sim \subseteq \sim_c$) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\sim \subseteq \sim_c$.

Proof See Proposition 3.13 in [Lei01]. ■

I now return to the epi hypothesis discussed earlier and show that if it is satisfied then $\dot{\longrightarrow} = \longrightarrow$. A corollary is that $\dot{\longrightarrow} = \dot{\longrightarrow}_c$ and thus $\sim = \sim_c$. (In Chapter 7 of [Lei01], I show exact conditions on redexes of a functorial reactive system of graph contexts which hold iff the epi hypothesis is satisfied.)

Definition 3.14 (redexes have epi preimages) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system. Say that *redexes have epi preimages* iff for all $\mathtt{l} \in \hat{\mathbf{C}}$, if there exists $r \in \mathbf{C}$ such that $(\mathcal{F}(\mathtt{l}), r) \in \mathsf{Reacts}$, then $\mathtt{l}$ is an epi. (Recall that lowercase teletype letters stand for arrows in $\hat{\mathbf{C}}$ with domain ε.) ■

Proposition 3.15 Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system. Suppose that redexes have epi preimages. Then $\xrightarrow{\mathsf{id}} = \longrightarrow$.



Proof Follows from Proposition 2.9. See Proposition 3.15 in [Lei01]. ■

Corollary 3.16 Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Suppose the epi hypothesis of Proposition 3.15 is satisfied, namely, the preimage of every redex is an epi. Then $\dot{\longrightarrow} = \dot{\longrightarrow}_c$ and thus $\sim = \sim_c$.

Proof See Corollary 3.16 in [Lei01]. ■

3.5 Weak bisimulation

Weak bisimulation [Mil88] is a coarser equivalence than strong bisimulation and is less sensitive to the number of silent steps made by the agents it compares. A single labelled transition by one agent may be matched by a *weak labelled transition* of another, namely a sequence of reactions, followed by a like transition, followed by further reactions.

Definition 3.17 (weak labelled transition ($\stackrel{\cdot}{\Longrightarrow}$)) The *weak labelled transition relation* is defined as follows:

$$a \stackrel{F}{\Longrightarrow} a' \quad \text{iff} \quad a \longrightarrow^* \stackrel{F}{\longrightarrow} \longrightarrow^* a' \,. \qquad \blacksquare$$

The definition of weak bisimulation follows exactly the form set out by Milner:

Definition 3.18 (weak bisimulation over $\stackrel{\cdot}{\longrightarrow}$; cf. Definition 5 on p. 108 in [Mil88]) Let $\mathcal{S} \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0,m)^2$ be a relation that contains pairs of agents with common codomain. $\mathcal{S}$ is a *weak simulation* over $\stackrel{\cdot}{\longrightarrow}$ iff it satisfies the following properties for all $(a,b) \in \mathcal{S}$:

1. If $a \longrightarrow a'$, then there exists b' such that $b \longrightarrow^* b'$ and $(a',b') \in \mathcal{S}$.
2. If $a \stackrel{F}{\longrightarrow} a'$ where F is not an iso, then there exists b' such that $b \stackrel{F}{\Longrightarrow} b'$ and $(a',b') \in \mathcal{S}$.

$\mathcal{S}$ is a *weak bisimulation* iff $\mathcal{S}$ and $\mathcal{S}^{-1}$ are weak simulations. Let $\approx$ be the *largest weak bisimulation* over $\stackrel{\cdot}{\longrightarrow}$. $\blacksquare$

In the second clause of the definition of weak simulation, the label F is required to be not an iso. Without this requirement, the definition would force $a \stackrel{F}{\longrightarrow}$ to be matched by $b \stackrel{F}{\Longrightarrow}$, for F an iso. By the definition of IPOs, iso-labelled transitions are essentially like id labelled transitions. As argued in the previous subsection $\stackrel{\mathsf{id}}{\longrightarrow} \subseteq \longrightarrow$, and moreover the converse holds in certain cases. So allowing F to be an iso in clause 2 would override clause 1. But clause 1 embodies a basic principle, namely that a silent transitions is matched by *zero or more* silent transitions and not by *one or more*.

The congruence property of weak bisimulation is more limited than that of strong bisimulation: $\approx$ is a congruence with respect to arrows only in $\mathbf{D}$, a subcategory of $\mathbf{C}$. Recall that $\mathbf{D}$ consists of the "reactive contexts", i.e. the contexts that allow reaction under them: $a \longrightarrow a'$ implies $Da \longrightarrow Da'$ for $D \in \mathbf{D}$. (See Definition 2.) This limitation is not surprising. In CCS, for example, weak bisimulation is not a congruence with respect to summation contexts, which are not reactive, i.e. we do not have that $a \longrightarrow a'$ implies $a + b \longrightarrow a' + b$. (I am using a, b for agents *and not for names* in order to maintain consistency with the notation of the rest of this paper.) Use of the hypothesis $C \in \mathbf{D}$ occurs twice in the proof.

In CCS, weak bisimulation *is* a congruence with respect to some non-reactive contexts, namely the prefixing contexts $x.-$ and $\bar{x}.-$. We would require richer structure than is contained in Part 2 in order to have a category of CCS contexts, namely the nesting of graphs to represent prefixing, some added data (not yet well understood) to represent summation, and the inclusion of free names to represent the naming structure of CCS. (See Section 5 in Part 2 for a discussion of these open problems.) If we could construct

such a category then it is likely that proving explicitly that weak bisimulation is preserved by prefixing would be easy since the only initial labelled transition of a prefixed agent is based on the prefix itself. Nonetheless, it is worth considering whether we could get a better general congruence result for weak bisimulation by dividing the set of reactive contexts in two, with one set containing prefixing-like contexts and another containing sum-like contexts. I am not sure how this would work, but something similar is done in work on rule formats for structural operational semantics by Bloom in [Blo93] who distinguishes "tame" from "wild" contexts.

Theorem 3.19 (congruence for $\approx$ w.r.t. D) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\approx$ is a congruence with respect to all contexts in $\mathbf{D}$, i.e. $a \approx b$ implies $Ca \approx Cb$ for all $C \in \mathbf{D}$ of the required domain.

Proof See Theorem 3.19 in [Lei01]. ■

How can we get a congruence with respect to $\mathbf{C}$, not just $\mathbf{D}$? One possibility is to replace the $\overset{F}{\Longrightarrow\!\triangleright}$ of clause 2 in Definition 3.18 with $\overset{F}{\longrightarrow\!\triangleright}\!\!\longrightarrow^{*}\!\!\triangleright$. The largest symmetric relation satisfying this new definition is a congruence with respect to all of $\mathbf{C}$. A second possibility is to make this change for the *first step only* of the weak bisimulation relation and then revert to the normal definition in Definition 3.18:

Definition 3.20 (greedy weak bisimulation ($\approx_{\mathsf{gr}}$)) Let $\approx_{\mathsf{gr}} \subseteq \bigcup_{m \in \mathsf{obj}\,\mathbf{C}} \mathbf{C}(0,m)^2$ be the largest symmetric relation that contains pairs of agents with common codomain and which satisfies the following properties:

1. If $a \longrightarrow\!\triangleright\, a'$, then there exists b' such that $b \longrightarrow\!\!\triangleright^{*} b'$ and $a' \approx b'$.
2. If $a \overset{F}{\longrightarrow\!\triangleright}\, a'$ where F is not an iso, then there exists b' such that $b \overset{F}{\longrightarrow\!\triangleright}\!\!\longrightarrow^{*}\!\!\triangleright\, b'$ and $a' \approx b'$.

We call $\approx_{\mathsf{gr}}$ *greedy weak bisimulation.* ■

Notice that this definition requires that $a' \approx b'$, not the stronger condition that $a' \approx_{\mathsf{gr}} b'$. The relation $\approx_{\mathsf{gr}}$ is also a congruence with respect to $\mathbf{C}$. The proof is almost identical to that of Theorem 3.19:

Theorem 3.21 (congruence for $\approx_{\mathsf{gr}}$ w.r.t. C) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\approx_{\mathsf{gr}}$ is a congruence with respect to all contexts in $\mathbf{C}$, i.e. $a \approx b$ implies $Ca \approx Cb$ for all $C \in \mathbf{C}$ of the required domain. ■

The idea of having a special requirement for the first step of a weak bisimulation followed by the use of standard weak bisimulation to compare the continuations is well-established. See, for example, the definition of *observational congruence* (Definition 2 on p. 153 in [Mil88]), also known as *rooted weak bisimulation* in the literature. Experience with CCS suggests that a congruence in ensured by changing clause 1 (to require $b \longrightarrow\!\triangleright$

$\longrightarrow^{*} b'$) and not clause 2 in Definition 3.18 — exactly the opposite of what is done in Definition 3.20. This anomaly requires further research, both to find categorical theory to model different kinds of non-reactive contexts and to show that RPOs exist for categories of graph-like contexts that *contain summation*.

3.6 Traces preorder

This subsection addresses the *traces preorder*, a simple preorder that compares agents based on their finite traces. A *trace* (see p. 41 in [Hoa85]) is a sequence of labelled transitions.

The traces preorder is insensitive to non-determinism and deadlock so is of limited use. Nonetheless, traces are good for specifying security properties since it is easy to formulate that an agent does not possess certain "bad" traces (see, for example, [Pau98, SV00]).

The main motivation for this subsection is to provide a warmup for the next one, which looks at the *failures preorder*. As a result, the traces considered here are all strong (not interspersed with reaction steps) since the way to handle the weak case is subsumed by the results presented in the next subsection.

Definition 3.22 (traces preorder ($\gtrsim_{\mathsf{tr}}$); cf. p. 45 in [Ros98]) A pair of agents b and a with common codomain are related by the *traces preorder*, written $a \gtrsim_{\mathsf{tr}} b$, iff all the traces of a are traces of b: for every trace $\langle F_1, \ldots, F_n \rangle$,

$$a \xrightarrow{F_1} \cdots \xrightarrow{F_n} \quad \text{implies} \quad b \xrightarrow{F_1} \cdots \xrightarrow{F_n} .$$ ■

(In this subsection and the next, n, m are natural numbers and not objects of a category.)

The proof of congruence is more complicated that that of strong bisimulation (Theorem 3.9) because we need to consider traces, not just individual labelled transitions. The heart of the argument is an inductive construction of a trace of a from a trace of Ca. Each inductive step cuts off a portable IPO square (see Lemma 3.5) which is subsequently pasted back on (Lemma 3.7) when constructing a trace of Cb from a trace of b.

Theorem 3.23 (congruence for $\gtrsim_{\mathsf{tr}}$) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\gtrsim_{\mathsf{tr}}$ is a congruence, i.e. $a \gtrsim_{\mathsf{tr}} b$ implies $Ca \gtrsim_{\mathsf{tr}} Cb$ for all $C \in \mathbf{C}$ of the required domain.

Proof Suppose that $a \gtrsim_{\mathsf{tr}} b$. Let C be any context of appropriate domain. We wish to prove that $Ca \gtrsim_{\mathsf{tr}} Cb$. The proof is divided into three parts, which are shown schematically in Figure 34.

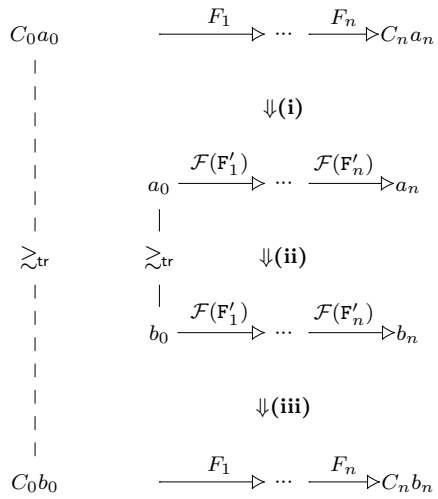
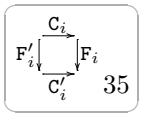


Figure 34: Schema of the congruence proof for $\gtrsim_{\mathsf{tr}}$

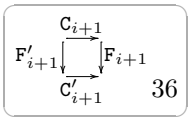
(i): Let $\bar{a}_0 \mathrel{\hat{=}} Ca$ and consider any trace $\bar{a}_0 \xrightarrow{F_1} \cdots \xrightarrow{F_n} \bar{a}_n$, where $n \geq 0$. We construct $(a_0, C_0) \ldots (a_n, C_n)$ and the square shown in Figure 35 for $1 < i \leq n$ such that the following conditions hold for $0 \leq i \leq n$:



$\bar{a}_i = C_i a_i$		Tr-a
$a_{i-1} \xrightarrow{\mathcal{F}(\mathtt{F}'_i)} a_i$	for $i \neq 0$	Tr-lab
$C_i \in \mathbf{D}$	for $i \neq 0$	Tr-D
$\mathcal{F}(\mathtt{C}_i) = C_{i-1}$	for $i \neq 0$	Tr-C
$\mathcal{F}(\mathtt{C}'_i) = C_i$	for $i \neq 0$	Tr-Cprime
$\mathcal{F}(\mathtt{F}_i) = F_i$	for $i \neq 0$	Tr-F
Figure 35 is an IPO	for $i \neq 0$.	Tr-IPO

base: Let $a_0 \mathrel{\hat{=}} a$ and $C_0 \mathrel{\hat{=}} C$. Then Tr-a holds and the other conditions are vacuous.

step: We construct a_{i+1}, C_{i+1} and the square in Figure 36 from a_i, C_i as follows, assuming $0 \leq i < n$. By the inductive hypothesis Tr-a holds for i, thus $\bar{a}_i = C_i a_i$. Since $\bar{a}_i \xrightarrow{F_i} \bar{a}_{i+1}$, Lemma 3.5 implies that there exist $a_{i+1} \in \mathbf{C}$ and an IPO square shown in Figure 36 such that $a_i \xrightarrow{\mathcal{F}(\mathtt{F}'_{i+1})} a_{i+1}$ and



$$\bar{a}_{i+1} = C_{i+1} a_{i+1} \qquad C_{i+1} \in \mathbf{D} \qquad \mathcal{F}(\mathtt{C}_{i+1}) = C_i \qquad \mathcal{F}(\mathtt{F}_{i+1}) = F_{i+1} \,.$$

where we let $C_{i+1} \mathrel{\hat{=}} \mathcal{F}(\mathtt{C}'_{i+1})$. Then all of the inductive properties are satisfied for $i + 1$.

Thus by Tr-lab $a = a_0 \xrightarrow{\mathcal{F}(\mathtt{F}'_1)\triangleright} \cdots \xrightarrow{\mathcal{F}(\mathtt{F}'_n)\triangleright} a_n$.

(ii): Since $a \gtrsim_{\mathrm{tr}} b$ there exist $b_0 \ldots b_n$ such that $b = b_0 \xrightarrow{\mathcal{F}(\mathtt{F}'_1)\triangleright} \cdots \xrightarrow{\mathcal{F}(\mathtt{F}'_n)\triangleright} b_n$.

(iii): We now claim that $C_i b_i \xrightarrow{F_{i+1}\triangleright} C_{i+1} b_{i+1}$ for $0 \leq i < n$.

Since $b_i \xrightarrow{\mathcal{F}(\mathtt{F}'_{i+1})\triangleright} b_{i+1}$ and Figure 36 is an IPO (by Tr-IPO), with $\mathcal{F}(\mathtt{C}'_{i+1}) =^{\text{Tr-Cprime}} C_{i+1} \in \mathbf{D}$ by Tr-D, then Lemma 3.7 implies

$$C_i b_i \;=^{\text{Tr-C}}\; \mathcal{F}(\mathtt{C}_{i+1}) b_i \xrightarrow{\mathcal{F}(\mathtt{F}_{i+1})\triangleright} \mathcal{F}(\mathtt{C}'_{i+1}) b_{i+1} \;=^{\text{Tr-Cprime}}\; C_{i+1} b_{i+1}\ ,$$

and thus by Tr-F, $C_i b_i \xrightarrow{F_{i+1}\triangleright} C_{i+1} b_{i+1}$. So

$$Cb = C_0 b_0 \xrightarrow{F_1\triangleright} \cdots \xrightarrow{F_n\triangleright}$$

as desired. ■

3.7 Failures preorder

This subsection looks at the failures preorder, which is a fundamental part of the failures and divergences model of CSP [Hoa85]. I do not consider divergences here, so the definition I use only employs failures. The failures preorder is sensitive to non-determinism and deadlock (see Subsection 3.3 in [Ros98]). The failures of an agent provide a domain-theoretic interpretation, assigning a meaning to each agent independently of the others (unlike for bisimulation). This makes failures properties well-suited to model checking [Ros94, Low96].

In order to define a failure of an agent, I first extend the notion of a weak labelled transition to allow for sequences of labels (not just single labels):

Definition 3.24 (weak labelled transition extended; cf. Definition 3.17)

$$\begin{array}{rcl} a \xRightarrow{\langle\,\rangle\triangleright} a' & \text{iff} & a \longrightarrow\!\!\!\!\triangleright^{*}\, a' \\ a \xRightarrow{\langle F\rangle \hat{}\, t\triangleright} a' & \text{iff} & a \xRightarrow{F\triangleright} \xRightarrow{t\triangleright} a'\ , \end{array}$$

where t is a sequence of arrows of appropriate domain and $\hat{}$ is the concatenation operator. ■

A failure of a consists of a sequence of weak labelled transitions t and a set of labels X such that a evolves by $\xRightarrow{t\triangleright}$ to a *stable state* (one for which no reactions are possible) which *refuses* X, i.e. cannot engage in a transition labelled by any of the arrows in X. To prevent reactions from masquerading as labelled transitions, every arrow in t and X is not an iso (cf. the discussion immediately following Definition 3.18).

Definition 3.25 (failure; cf. p. 171 in [Ros98]) A *failure* of a is a pair (t, X) where t is a finite sequence of arrows (each not an iso) and X is a set of arrows (each not an iso) for which there exists a' such that the following conditions hold:

$a \stackrel{t}{\Longrightarrow\!\!\triangleright} a'$	a has a weak trace t;
$a' \not\longrightarrow\!\!\triangleright$	a' is stable;
$\forall F \in X.\ a' \stackrel{F}{\not\longrightarrow\!\!\triangleright}$	a' refuses X. ■

Definition 3.26 (failures preorder ($\gtrapprox_{\mathsf{f}}$); cf. p. 193 in [Ros98]) A pair of agents b and a with common codomain are related by the *failures preorder*, written $a \gtrapprox_{\mathsf{f}} b$, iff all the failures of a are possessed by b. ■

The relation $\gtrapprox_{\mathsf{f}}$ is only a congruence with respect $\mathbf{D}$, the subcategory of $\mathbf{C}$ consisting of reactive contexts (cf. Theorem 3.19). The only use of the hypothesis $C \in \mathbf{D}$ occurs in the base case of the induction.

The proof, which we omit, is similar to that of the traces preorder however there are two aspects that require care: the cutting and pasting of portable IPO squares for weak labelled transitions and the propagation of refusal sets.

Theorem 3.27 (congruence for $\gtrapprox_{\mathsf{f}}$ w.r.t. D) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial reactive system which has all redex-RPOs. Then $\gtrapprox_{\mathsf{f}}$ is a congruence with respect to all contexts in $\mathbf{D}$, i.e. $a \gtrapprox_{\mathsf{f}} b$ implies $Ca \gtrapprox_{\mathsf{f}} Cb$ for all $C \in \mathbf{D}$ of the required domain.

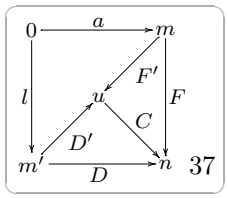
Proof See Theorem 3.26 in [Lei01]. ■

3.8 Multi-hole contexts

As anticipated at the end of the previous section, this subsection enriches the definition of functorial reactive systems to model explicitly multi-hole contexts. At first glance, there is no obstacle in accommodating multi-hole contexts directly within the existing theory of reactive systems. (For simplicity, let us ignore functorial reactive systems until later.) However this does not work well as I illustrate in the next few paragraphs.

If we consider multi-hole term algebra contexts (as used in term rewriting), say, then we can choose the objects of $\mathbf{C}$ to be natural numbers and the arrows to be tuples of contexts (that use each hole exactly once) constructed from function symbols taken from some signature Σ. Concretely, if $\Sigma = \{\alpha, \alpha', \beta, \gamma\}$, where α and α' are constants, β is a 1-place function symbol, and γ is a 2-place function symbol, then, we have the following examples of arrows:

$$\begin{aligned} C_0 &\;\hat{=}\; \langle -_1, \beta(\alpha')\rangle &&: 1 \to 2 \\ C_1 &\;\hat{=}\; \langle \gamma(-_2, \alpha'), \alpha, \beta(-_1)\rangle &&: 2 \to 3 \\ C_1C_0 &= \langle \gamma(\beta(\alpha'), \alpha'), \alpha, \beta(-_1)\rangle &&: 1 \to 3 \end{aligned}$$

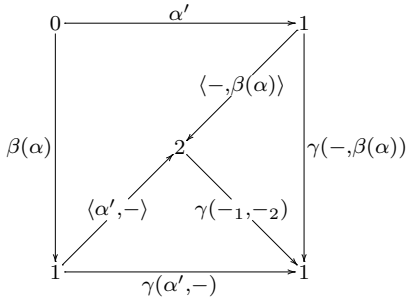


But what is an *agent*? A natural choice is to take agents to be pure terms, i.e. arrows $0 \to 1$ (a 1-tuple containing a term context with 0 holes). But this is not supported by the definition of reactive system, where agents consist of all arrows $0 \to m$ for m an arbitrary object of $\mathbf{C}$. This discrepancy is non trivial: if we try to confine the definition of labelled transition and bisimulation to use only a limited set of agents, say those arrows $0 \to m$ for $m \in A$, where A is some subset of the objects of $\mathbf{C}$, then the proof of congruence goes wrong. The problem is that even if $m, m', n \in A$ in Figure 37, it is not necessarily the case that an RPO F', D', C yields an object $u \in A$. Thus even though F', D' forms an IPO with respect to a, l, it is not the case that $a \xrightarrow{F'}\triangleright$ since the codomain of F' is not in A.

However, it is exactly the fact that RPOs sometimes do *not* produce an object in A that gives multi-hole contexts their power and that makes them worth considering. To see why, suppose that we take $\mathbf{C}$ to be a category of exclusively 0- and 1-hole contexts. Then RPOs exist, as a corollary of Sewell's dissection result for terms (Lemma 1 in [Sew]). Consequently, bisimulation is a congruence for term rewriting systems. The resulting labels are unnecessarily heavy, though. For consider the reaction rule $(\beta(\alpha), \alpha')$; we have $\alpha \xrightarrow{\beta(-)}\triangleright \alpha'$ which corresponds to our intuition that α needs $\beta(-)$ to react. (When there is no confusion I use $-$ for $-_1$.) Unfortunately, we also have a labelled transition where the label contains a *complete copy of the redex*:

$$\alpha' \xrightarrow{\gamma(-,\beta(\alpha))}\triangleright \gamma(\alpha', \alpha') \ .$$

This was discussed at the end of the previous section. This heavy labelled transition is absent when we look at multi-hole contexts, as illustrated with the help of the diagram below. (Tuple brackets $\langle\ \rangle$ are omitted from singletons.)



If we work in the category of 0- and 1-hole contexts then the outer square is an IPO, which gives rise to the transition $\xrightarrow{\gamma(-,\beta(a))}\triangleright$ mentioned earlier. By admitting multi-hole contexts we have given the outer-square a simpler RPO.

It is now possible to make precise the motivation for the results developed in this subsection. The goal is to reconcile two things: (i) we want to restrict of the collection of agents of $\mathbf{C}$ to arrows $0 \to m$ for $m \in A$, where A can be a proper subset of the objects

of $\mathbf{C}$ (for example $A = \{1\}$ in the case of multi-hole term contexts); (ii) we want to admit RPOs that yield objects which are not contained in A.

The key idea is to consider the notion of a *strict monoidal category* $(\mathbf{C}, \otimes, 0)$, a category $\mathbf{C}$ equipped with a functor $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ and an object 0 such that $\otimes$ is strictly associative and has unit 0. The role of the tensor $\otimes$ is to "tuple" arrows and objects, e.g. recalling the term contexts C_0, C_1 from above, we have that:

$$\begin{aligned} C_0 &\hat{=} \langle -_1, \beta(\alpha') \rangle && : 1 \to 2 \\ C_1 &\hat{=} \langle \gamma(-_2, \alpha'), \alpha, \beta(-_1) \rangle && : 2 \to 3 \\ C_0 \otimes C_1 &= \langle -_1, \beta(\alpha'), \gamma(-_3, \alpha'), \alpha, \beta(-_2) \rangle && : 3 \to 5 \\ C_1 \otimes C_0 &= \langle \gamma(-_2, \alpha'), \alpha, \beta(-_1), -_3, \beta(\alpha') \rangle && : 3 \to 5 \end{aligned}$$

The following definition extends that of a functorial reactive system by postulating that both the upstairs and downstairs categories are monoidal and requiring that the functor between them respects the monoidal structure. The same enrichment could be performed on a reactive system. There is, however, no reason not to take more general approach shown here.

Definition 3.28 (functorial monoidal reactive system; cf. Definition 3.1) A *functorial monoidal reactive system* consists of a functorial reactive system $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ with the following added structure:

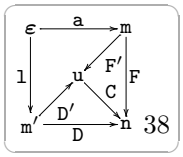
- Both $\hat{\mathbf{C}}$ and $\mathbf{C}$ are strict monoidal categories with unit objects $\boldsymbol{\varepsilon}$ and 0, respectively, and tensor $\otimes$ (the same symbol being used for both categories).
- $\mathcal{F}$ preserves tensors, i.e. $\mathcal{F}(\mathtt{C}_1 \otimes \mathtt{C}_0) = \mathcal{F}(\mathtt{C}_1) \otimes \mathcal{F}(\mathtt{C}_0)$.
- $\mathcal{F}$ preserves and creates units, i.e. for all $\mathtt{u} \in \mathsf{obj}\,\hat{\mathbf{C}}$, $\mathcal{F}(\mathtt{u}) = 0$ iff $\mathtt{u} = \boldsymbol{\varepsilon}$.
- There is a subset A of $\mathbf{C}$-objects and a subset $\mathtt{A}$ of $\hat{\mathbf{C}}$-objects, where $\mathtt{A}$ is the preimage under $\mathcal{F}$ of A. We use $m, m', \ldots$ to range over A and $\mathtt{m}, \mathtt{m}', \ldots$ to range over $\mathtt{A}$.
- $\mathsf{Reacts} \subseteq \bigcup_{m \in A} \mathbf{C}(0, m)^2$.
- Pairing with an agent yields a reactive context: $a \otimes \mathsf{id}_{m'} \in \mathbf{D}$ for $a : 0 \to m$. ■

The *agents* are arrows $0 \to m$ where $m \in A$ and the *agent contexts* are arrows $m \to m'$, for $m, m' \in A$. Thus the reaction rules of Reacts are only between agents. We overload the terminology for $\hat{\mathbf{C}}$ in a straightforward way: arrows $\boldsymbol{\varepsilon} \to \mathtt{m}$ of $\hat{\mathbf{C}}$ are also agents for $\mathtt{m} \in \mathtt{A}$; arrows $\mathtt{m} \to \mathtt{m}'$ of $\hat{\mathbf{C}}$ are also agent contexts for $\mathtt{m}, \mathtt{m}' \in \mathtt{A}$.

The definition of labelled transition confines the arguments to be agents:

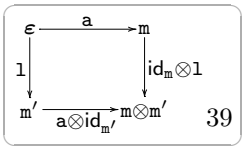
Definition 3.29 (labelled transition for functorial monoidal reactive systems ($\xrightarrow[\mathtt{m}]{\cdot}\!\triangleright$); cf. Definition 3.3) $a \xrightarrow[\mathtt{m}]{F}\!\triangleright\, a'$ iff $a \xrightarrow{F}\!\triangleright\, a'$ and a, a' are agents (thus forcing F to be an agent context). ■

Two properties now replace the hypothesis that $\mathcal{F}$ has all redex-RPOs. The first asserts that the RPO consists either of agent contexts or of pairing operations that place disjoint instances of $\mathtt{a}$ and $\mathtt{l}$ into the composite $\mathtt{Fa} = \mathtt{Dl}$:



Definition 3.30 ($\mathcal{F}$ has all monoidal-redex-RPOs; cf. Definition 3.4) Suppose $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ is a functorial monoidal reactive system and that $\mathtt{a}, \mathtt{l}$ are agents, $\mathtt{F}, \mathtt{D}$ are agent contexts, $\mathcal{F}(\mathtt{D}) \in \mathbf{D}$, and there exists $r \in \mathbf{C}$ such that $(\mathcal{F}(\mathtt{l}), r) \in \mathsf{Reacts}$. Then $\mathcal{F}$ *has all monoidal-redex-RPOs* if any square, such as in Figure 38, has an RPO (as shown) such that if $\mathtt{u} \notin \mathtt{A}$ then $\mathtt{u} = \mathtt{m} \otimes \mathtt{m}'$, $\mathtt{F}' = \mathsf{id}_{\mathtt{m}} \otimes \mathtt{l}$, and $\mathtt{D}' = \mathtt{a} \otimes \mathsf{id}_{\mathtt{m}'}$. ■

The second property asserts that pairing agent contexts yields an IPO:



Definition 3.31 ($\mathcal{F}$ has all monoidal-redex-IPOs) A functorial monoidal reactive system $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ *has all monoidal-redex-IPOs* if any square, such as in Figure 39, is an IPO, provided $\mathtt{a}, \mathtt{l}$ are agents and there exists $r \in \mathbf{C}$ such that $(\mathcal{F}(\mathtt{l}), r) \in \mathsf{Reacts}$. ■

Just before giving a proof of congruence for strong bisimulation we need to consider a corollary of Lemma 3.7 for functorial monoidal reactive systems:

Corollary 3.32 (portable IPO pasting for functorial monoidal reactive systems; cf. Lemma 3.7) Suppose $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ is a functorial monoidal reactive system and has all monoidal-redex-RPOs. The following inference rule holds:

$$\frac{\begin{array}{c}\mathtt{C} \\ \mathtt{F}' \square \mathtt{F} \\ \mathtt{C}'\end{array} \text{ is an IPO consisting of agent contexts} \qquad a \xrightarrow[\mathtt{m}]{\mathcal{F}(\mathtt{F}')} a' \qquad \mathcal{F}(\mathtt{C}') \in \mathbf{D}}{\mathcal{F}(\mathtt{C})a \xrightarrow[\mathtt{m}]{\mathcal{F}(\mathtt{F})} \mathcal{F}(\mathtt{C}')a'} .$$

■

Strong bisimulation compares pairs of agents:

Definition 3.33 (strong bisimulation over $\xrightarrow[\mathtt{m}]{\cdot}\triangleright$; cf. Definition 3.8) Let $\sim_{\mathtt{m}} \subseteq \bigcup_{m \in A} \mathbf{C}(0, m)^2$ be the largest strong bisimulation over $\xrightarrow[\mathtt{m}]{\cdot}\triangleright$ such that $\sim_{\mathtt{m}}$ contains only pairs of agents with common codomain. ■

Finally we prove congruence. As promised at the end of the previous section, the argument mirrors closely congruence proofs in typical process calculi. In particular, two cases are distinguished when analysing the transition $Ca \xrightarrow[\mathtt{m}]{F}\triangleright$: (i) a, C, and F all contribute, in which case a itself has a labelled transition; (ii) only C and F contribute, in which case $Cb \xrightarrow[\mathtt{m}]{F}\triangleright$ without needing to consider the behaviour of a and b.

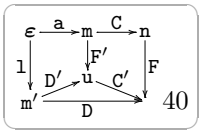
Theorem 3.34 (congruence for $\sim_{\mathtt{m}}$ w.r.t. agent contexts) Let $\mathcal{F} : \hat{\mathbf{C}} \to \mathbf{C}$ be a functorial monoidal reactive system which has all monoidal-redex-RPOs and has all

monoidal-redex-IPOs. Then $\sim_{\mathtt{m}}$ is a congruence with respect to agent contexts, i.e. $a \sim_{\mathtt{m}} b$ implies $Ca \sim_{\mathtt{m}} Cb$ for any agent context $C \in \mathbf{C}$ of the required domain.

Proof By symmetry, it is sufficient to show that the following relation is a strong simulation:

$$\mathcal{S} \hat{=} \{(Ca, Cb) \;\; / \;\; a \sim_{\mathtt{m}} b \text{ and } C \in \mathbf{C} \text{ is an agent context}\} .$$

Suppose that $a \sim_{\mathtt{m}} b$ and $C \in \mathbf{C}$, and thus $(Ca, Cb) \in \mathcal{S}$, where $a, b : 0 \to m$ and $C : m \to n$. Suppose $Ca \xrightarrow[\mathtt{m}]{F} a'$. By the definition of $\xrightarrow[\mathtt{m}]{F}$ and the hypothesis that $\mathcal{F}$ creates compositions, there exist $\hat{\mathbf{C}}$-arrows $\mathtt{a} : \varepsilon \to \mathtt{m}, \mathtt{C} : \mathtt{m} \to \mathtt{n}, \mathtt{l} : \varepsilon \to \mathtt{m}', \mathtt{F}, \mathtt{D}$ and a $\mathbf{C}$-arrow $r : 0 \to \mathcal{F}(\mathtt{m}')$ such that the big rectangle in Figure 40 is an IPO and



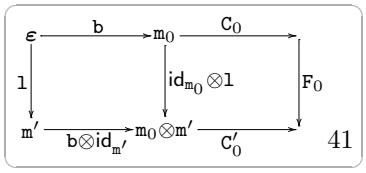
40

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

Because $\mathcal{F}$ has all monoidal-redex-RPOs, there exist $\mathtt{F}', \mathtt{D}', \mathtt{C}'$ forming an RPO in $\hat{\mathbf{C}}$, as in Figure 40. Note that $\mathcal{F}(\mathtt{C}') \in \mathbf{D}$ since $\mathcal{F}(\mathtt{C}')\mathcal{F}(\mathtt{D}') = \mathcal{F}(\mathtt{D}) \in \mathbf{D}$. By Proposition 2.6, the small left-hand square of Figure 40 is an IPO; Proposition 2.8 implies that the small right-hand square is an IPO too. Since $\mathcal{F}$ has all monoidal-redex-RPOs, we have additional information depending on whether $\mathtt{u} \in \mathtt{A}$. We consider both cases.

Case $\mathtt{u} \in \mathtt{A}$**:** By definition, $a \xrightarrow[\mathtt{m}]{\mathcal{F}(\mathtt{F}')} a''$ and $a' = \mathcal{F}(\mathtt{C}')a''$, where $a'' \hat{=} \mathcal{F}(\mathtt{D}')r$. Since $a \sim_{\mathtt{m}} b$, there exists b'' such that $b \xrightarrow[\mathtt{m}]{\mathcal{F}(\mathtt{F}')} b''$ and $a'' \sim_{\mathtt{m}} b''$. Since the small right-hand square in Figure 40 is an IPO and $\mathcal{F}(\mathtt{C}') \in \mathbf{D}$, Corollary 3.32 implies that $Cb \xrightarrow[\mathtt{m}]{F} \mathcal{F}(\mathtt{C}')b''$. Also, $a'' \sim_{\mathtt{m}} b''$ implies $(\mathcal{F}(\mathtt{C}')a'', \mathcal{F}(\mathtt{C}')b'') \in \mathcal{S}$, as desired.

Case $\mathtt{u} \notin \mathtt{A}$**:** We have that $\mathtt{F}' = \mathsf{id}_{\mathtt{m}} \otimes \mathtt{l}$ and $\mathtt{D}' = \mathtt{a} \otimes \mathsf{id}_{\mathtt{m}'}$. Since $\mathcal{F}$ lifts agents there exists $\mathtt{b} : \varepsilon \to \mathtt{m}_0$ such that $\mathcal{F}(\mathtt{b}) = b$, and thus $\mathcal{F}(\mathtt{m}_0) = m = \mathcal{F}(\mathtt{m})$. Since $\mathcal{F}$ preserves tensors, $\mathcal{F}(\mathtt{F}') = \mathsf{id}_{\mathcal{F}(\mathtt{m})} \otimes \mathcal{F}(\mathtt{l}) = \mathsf{id}_{\mathcal{F}(\mathtt{m}_0)} \otimes \mathcal{F}(\mathtt{l}) = \mathcal{F}(\mathsf{id}_{\mathtt{m}_0} \otimes \mathtt{l})$. Since $\mathcal{F}$ allows IPO sliding, the small right-hand IPO square of Figure 40 can be slid to form the small right-hand IPO square in Figure 41, where



41

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

Since $\mathcal{F}$ has all monoidal-redex-IPOs, the small left-hand square of Figure 41 is an IPO. Since $\mathcal{F}$ has all monoidal-redex-RPOs, Proposition 2.8 implies that the big rectangle in Figure 41 is an IPO. Since $\mathcal{F}$ preserves tensors and since pairing with an agent yields a reactive context, $\mathcal{F}(\mathtt{C}'_0(\mathtt{b} \otimes \mathsf{id}_{\mathtt{m}'})) = \mathcal{F}(\mathtt{C}')(b \otimes \mathsf{id}_{\mathcal{F}(\mathtt{m}')}) \in \mathbf{D}$ so:

$$Cb \xrightarrow[\mathtt{m}]{F} \mathcal{F}(\mathtt{C}'_0(\mathtt{b} \otimes \mathsf{id}_{\mathtt{m}'}))r = \mathcal{F}(\mathtt{C}')(b \otimes r) = \mathcal{F}(\mathtt{C}')(\mathsf{id}_m \otimes r)b .$$

The last equality is a standard property of strict monoidal categories. Furthermore,

$$a' = \mathcal{F}(\mathtt{D})r = \mathcal{F}(\mathtt{C}')\mathcal{F}(\mathtt{D}')r = \mathcal{F}(\mathtt{C}')\mathcal{F}(\mathtt{a} \otimes \mathsf{id}_{\mathtt{m}'})r = \mathcal{F}(\mathtt{C}')(a \otimes r) = \mathcal{F}(\mathtt{C}')(\mathsf{id}_m \otimes r)a \ .$$

Thus $a \sim_{\mathtt{m}} b$ implies $(\mathcal{F}(\mathtt{C}')(\mathsf{id}_m \otimes r)a, \mathcal{F}(\mathtt{C}')(\mathsf{id}_m \otimes r)b) \in \mathcal{S}$ as desired. ■

To round out this subsection, let us look again at the example of multi-hole term contexts over a signature Σ.

Definition 3.35 (multi-hole term contexts ($\mathbf{T}^*(\Sigma)$)) Given a *signature* Σ of function symbols then the category of *multi-hole term contexts* $\mathbf{T}^*(\Sigma)$ over Σ is constructed as follows: the objects are the natural numbers; an arrow $j \to k$ is a k-tuple of terms over the signature $\Sigma \cup \{-_1, \ldots, -_j\}$ containing exactly one use of each hole $-_i$ $(1 \leq i \leq j)$. (When $j = 1$, I often write $-_1$ as $-$.) The *identities* are: $\mathsf{id}_j \mathrel{\hat{=}} \langle -_1, \ldots, -_j \rangle : j \to j$. For $C = \langle t_1, \ldots, t_k \rangle : j \to k$ and $D : k \to m$, their *composition* is the substitution

$$DC \mathrel{\hat{=}} \{t_1/-_1, \cdots, t_k/-_k\}D \ .$$ ■

In order to apply Theorem 3.34 to $\mathbf{T}^*(\Sigma)$, we let $\mathbf{C} = \hat{\mathbf{C}} = \mathbf{T}^*(\Sigma)$ and $\mathcal{F}$ be the identity functor. If we let $A = \{1\}$ then

- an agent of $\mathbf{T}^*(\Sigma)$ is a term $a : 0 \to 1$;
- agent context of $\mathbf{T}^*(\Sigma)$ is a term context $C : 1 \to 1$, i.e. a term containing a single hole.

We may choose any subcategory of $\mathbf{T}^*(\Sigma)$ to be the reactive contexts, subject to the conditions in Definition 3.28. The labelled transitions of $\mathbf{T}^*(\Sigma)$ depend, of course, on the reaction rules. Once these are specified, the labelled transition relation $\xrightarrow{\cdot}_{\mathtt{m}}$ is determined by Definition 3.29 and the induced strong bisimulation $\sim_{\mathtt{m}}$ by Definition 3.33.

As a corollary of Sewell's dissection result for terms (Lemma 1 in [Sew]), $\mathcal{F}$ has all monoidal-redex-RPOs and has all monoidal-redex-IPOs. Hence from Theorem 3.34 the induced strong bisimulation $\sim_{\mathtt{m}}$ is preserved by all term contexts. Let us now revisit the reactive system whose only reaction rule is $(\beta(\alpha), \alpha')$. It contains exactly the following labelled transitions:

$$D(\beta(\alpha)) \xrightarrow{-}_{\mathtt{m}} D(\alpha') \qquad \text{for all reactive term contexts } D \in \mathbf{D}$$
$$\alpha \xrightarrow{\beta(-)}_{\mathtt{m}} \alpha'$$

These agree with the transitions found by Sewell in the case of ground-term rewriting. I believe for any reaction rules specified by Reacts the synthesised labelled transitions for $\mathbf{T}^*(\Sigma)$ coincide exactly with Sewell's.

4 Conclusions

This final section summarises some of the accomplishments of the paper. I delay the discussion of future work to the conclusion of Part 2, since the relevant open problems involve the combination of both parts.

My main motivation in this work is to provide mathematical tools that ease the construction of new process calculi. Every computational phenomenon (distributed hosts, failures, cryptography, etc.) that gives rise to a new calculus carries with it rules for describing how it works, how a computation evolves. These are often reaction rules, which capture the allowable state changes. The redexes are typically *composites* formed from atoms, so the following questions naturally arise. When an agent has parts of a redex, which atoms does the agent require to form a complete redex? Can we get a definition of labelled transition if we choose the labels to be those small *contexts* that complement an agent to complete a redex? These questions are not original with me: they are often used by the process calculus community as motivation and intuition for designing specific labelled transition relations. I believe, though, that the original contribution of my work is to give a general way of transforming reaction rules into tractable labelled transitions.

By tractable, I mean two things: (i) the labels are small, i.e. contain *only* those parts of a redex necessary to complete a whole redex in an agent; (ii) the labelled transitions yield operational equivalences and preorders that are congruences. The key ideas in trying to achieve both desiderata are that of a *relative pushout* (RPO) and an *idem pushout* (IPO). With respect to (ii) the results are good: I prove congruence for strong bisimulation, weak bisimulation, the traces preorder, and the failures preorder. For (i), the results are encouraging: IPOs ensure that the labels of transitions contain no junk, i.e. that no lesser label would do. There is, however, more work required in making the labels even simpler by exploiting the uniformity present in reaction rules that contain metavariables. The conclusion of Part 2 discusses this problem in the paragraphs concerned with multi-hole redexes.

One of the most attractive qualities of the theory of RPOs is that it is *abstract*: it is applicable to all categories of contexts and therefore provides a unifying discipline. Before I understood RPOs, I spent several years thinking about *dissection results* for specific graph-like contexts: "If $C_0 a_0 = C_1 a_1$, then what part of a_i is present in C_{1-i} for $i = 0, 1$?" I could only get results about nodes (for example, which nodes of a_1 are supplied by C_0) but not about arcs. Trying to work in an *ad hoc* way with graphs was unsatisfying until the RPO theory arrived: it was only the discipline of a universal property in the category theory sense that sustained me in carrying out the constructions contained in Chapter 6, "RPOs for action graph contexts", in [Lei01]. Sewell's *ad hoc* reasoning about dissection for term algebras with parallel composition [Sew] was successful without employing RPOs, but his statement of dissection was complicated. To be fair, his dissections were for multi-

hole redexes that RPOs do not neatly handle. He is now recasting some of his results in terms of RPOs and other universal constructions.

RPOs thus support the goal of developing a shared theory that can produce labelled transitions and operational congruences for many process calculi. The task that we usually face for each new process calculus of defining labelled transitions and proving that equivalences and preorders are congruences is replaced by the task of choosing reactions and proving that RPOs exist. This is attractive leverage from RPOs. This leverage gains us a greater advantage if we can find families of functorial reactive systems with rich graph-like structure for which the existence of RPOs is proved once, and for which practical examples of process calculi are instances. Part 2 works towards this goal: it provides a series of theorems that verify the functorial reactive system axioms (such as IPO sliding) for a general class of functors, thus showing that that subclass of Milner's action calculi contexts satisfy these axioms. It then states the RPO result (omitting the proof for the sake of brevity). Finally, it concludes by detailing the future work required to enrich the structure of these graphs to the point that they embrace practical process calculi.

References

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

[AG97] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proc. 4th ACM Conf. on Computer and Communications Security, Zürich*, pages 36–47. ACM Press, 1997. {7}

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North Holland, revised edition, 1984. {8}

[BB90] G. Berry and G. Boudol. The chemical abstract machine. In *Proc. 17th Annual Symposium of Principles of Programming Languages*, pages 81–94. ACM Press, 1990. {6}

[BB92] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992. {6}

[Blo93] B. Bloom. Structural operational semantics for weak bisimulations. Technical Report TR-93-1373, Department of Computer Science, Cornell University, August 1993. {14, 38}

[BW01] M. Barr and C. F. Wells. Toposes, triples and theories. Version 1.1. Available from: http://www.cwru.edu/artsci/math/wells/pub/ttt.html, 2001. {18}

[CG98] L. Cardelli and A. D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structure, First International Conference, FoSSaCS '98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS '98, Lisbon, Portugal, March 28 – April 4, 1998, Proceedings*, volume 1378 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998. {7}

[CLM00] G. L. Cattani, J. J. Leifer, and R. Milner. Contexts and embeddings for closed shallow action graphs. Technical Report 496, Computer Laboratory, University of Cambridge, July 2000. Available from http://pauillac.inria.fr/∼leifer/. {16}

[CM91] A. Corradini and U. Montanari. An algebra of graphs and graph rewriting. In *4th Biennial Conference on Category Theory and Computer Science, Proceedings*, volume 530 of *Lecture Notes in Computer Science*, pages 236–260. Springer-Verlag, 1991. {16}

[DH84] R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984. {15}

[Ehr79] H. Ehrig. Introduction to the algebraic theory of graph grammar. In *Proc. first international Workshop on Graph Grammars and their application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69. Springer-Verlag, 1979. {16}

[FF86] M. Felleisen and D. P. Friedman. Control operators, the SECD-machine and the λ-calculus. In M. Wirsing, editor, *Formal Description of Programming Concepts III*, pages 193–217. North Holland, 1986. {8, 20}

[FG98] C. Fournet and G. Gonthier. A hierarchy of equivalences for asynchronous calculi. In *Automata, Languages and Programming, 25th International Colloquium, ICALP '98, Aalborg, Denmark, July 13–17, 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*, pages 844–855. Springer-Verlag, 1998. {15}

[FGL+96] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26–29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 406–421. Springer-Verlag, 1996. {7}

[Fou98] C. Fournet. *The Join-Calculus: a Calculus for Distributed Mobile Programming.* PhD thesis, École Polytechnique, November 1998. {15}

[GV92] J. F. Groote and F. W. Vaandrager. Structural operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992. {14}

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, August 1978. {5}

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985. {5, 39, 41}

[HY95] K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, November 1995. {15}

[JR99] A. Jeffrey and J. Rathke. Towards a theory of bisimulation for local names. In *14th Annual Symposium on Logic in Computer Science, 2–5 July, 1999, Trento, Italy*, pages 56–66. IEEE Press, 1999. {15}

[Kön99] B. König. Generating type systems for process graphs. In *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24–27, 1999, Proceedings*, volume 1664, pages 352–367. Springer-Verlag, 1999. {16}

[Lei01] J. J. Leifer. *Operational congruences for reactive systems*. PhD thesis, Computer Laboratory, University of Cambridge, 2001. Available in revised form as Technical Report 521,Computer Laboratory, University of Cambridge, 2001, from http://pauillac.inria.fr/∼leifer/. {14, 16, 19, 24, 25, 33, 35, 36, 38, 42, 48}

[Lei02] J. J. Leifer. Synthesising labelled transitions and operational congruences in reactive systems, part 2. 2002. Submitted for publication. Available from http://pauillac.inria.fr/∼leifer/. {3}

[Lév78] J.-J. Lévy. *Réductions correctes et optimales dans le lambda calcul*. PhD thesis, Université Paris VII, 1978. {8}

[LM00] J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In C. Palamidessi, editor, *CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22–25, 2000, Proceedings*, volume 1877 of *Lecture Notes in Computer Science*, pages 243–258. Springer-Verlag, 2000. Available from http://pauillac.inria.fr/∼leifer/. {16}

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27–29, 1996, Proceedings*, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer-Verlag, 1996. {5, 41}

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971. {18}

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980. {5}

[Mil88] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1988. {5, 28, 36, 37, 38}

[Mil92] R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2(2):119–141, 1992. {6, 20, 35}

[Mil96] R. Milner. Calculi for interaction. *Acta Informatica*, 33(8):707–737, 1996. {11}

[Mil01] R. Milner. Bigraphical reactive systems: basic theory. Technical Report 523, Computer Laboratory, University of Cambridge, 2001. Available from http://www.cl.cam.ac.uk/∼rm135/. {16}

[Moo56] E. F. Moore. Gedanken-experiments on sequential machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 129–153. Princeton University Press, 1956. {3}

[MPW89] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. Technical Report ECS-LFCS-89-85 and ECS-LFCS-89-86, Laboratory for the Foundations of Computer Science, University of Edinburgh, 1989. {6}

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, parts I and II. *Information and Computation*, 100(1):1–77, September 1992. {6}

[MS92] R. Milner and D. Sangiorgi. Barbed bisimulation. In *Automata, Languages and Programming, 19th International Colloquium, ICALP '92, Vienna, Austria, July 13–17, 1992, Proceedings*, volume 623 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992. {15}

[Par81] D. Park. Concurrency and automata on infinite sequences. In P. Duessen, editor, *Proc. 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981. {6, 26}

[Pau98] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *J. Computer Security*, 6:85–128, 1998. {39}

[Plo75] G. D. Plotkin. Call-by-name, call-by-value, and the λ-calculus. *Theoretical Computer Science*, 1(2):125–159, December 1975. {20}

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Department of Computer Science, University of Aarhus, 1981. {5}

[Rei85] W. Reisig. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1985. {4}

[Ros94] A. W. Roscoe. Model-checking CSP. In A. W. Roscoe, editor, *A Classical Mind: Essays in Honour of C. A. R. Hoare*, pages 353–378. Prentice-Hall, 1994. {5, 41}

[Ros98] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1998. {4, 5, 39, 41, 42}

[Sew] P. Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*. To appear. {11, 15, 20, 43, 47, 48}

[Sew98] P. Sewell. Global/local subtyping and capability inference for a distributed pi-calculus. In *Automata, Languages and Programming, 25th International Colloquium, ICALP '98, Aalborg, Denmark, July 13–17, 1998, Proceedings*, volume 1443 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998. {7}

[Sew00] P. Sewell. Applied π — A brief tutorial. Technical Report 498, Computer Laboratory, University of Cambridge, August 2000. {35}

[SV99] P. Sewell and J. Vitek. Secure compositions of insecure components. In *Proc. 12th Computer Security Foundations Workshop*. IEEE Press, June 1999. {7}

[SV00] P. Sewell and J. Vitek. Secure composition of untrusted code: wrappers and causality types. In *Proc. 13th Computer Security Foundations Workshop*. IEEE Press, July 2000. {39}

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001. {7}

[TP97] D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 – July 2, 1997*, pages 280–291. IEEE Press, 1997. {14}

[WN95] G. Winskel and M. Nielsen. Models for concurrency. In D. M. Gabbay, S. Abramsky, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 1–148. Oxford University Press, 1995. {4}

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scienti£que
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

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