Technical Report

UCAM-CL-TR-598
ISSN 1476-2986

Number 598

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Transition systems, link graphs and Petri nets

James J. Leifer, Robin Milner

August 2004

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

© 2004 James J. Leifer, Robin Milner

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/TechReports/

ISSN 1476-2986

Transition systems, link graphs and Petri nets

James J. Leifer[1] Robin Milner[2]

Abstract: A framework is defined within which reactive systems can be studied formally. The framework is based upon *s-categories*, a new variety of categories, within which reactive systems can be set up in such a way that *labelled transition systems* can be uniformly extracted. These lead in turn to behavioural preorders and equivalences, such as the failures preorder (treated elsewhere) and bisimilarity, which are guaranteed to be congruential. The theory rests upon the notion of *relative pushout* previously introduced by the authors.

The framework is applied to a particular graphical model known as *link graphs*, which encompasses a variety of calculi for mobile distributed processes. The specific theory of link graphs is developed. It is then applied to an established calculus, namely *condition-event Petri nets*.

In particular, a labelled transition system is derived for condition-event nets, corresponding to a natural notion of observable actions in Petri net theory. The transition system yields a congruential bisimilarity coinciding with one derived directly from the observable actions. This yields a calibration of the general theory of reactive systems and link graphs against known specific theories.

[1] INRIA, Domaine de Voluceau, BP105, 78153 Le Chesnay, France
[2] University of Cambridge, Computer Laboratory, JJ Thomson Avenue, Cambridge CB3 0FD, UK

Contents:

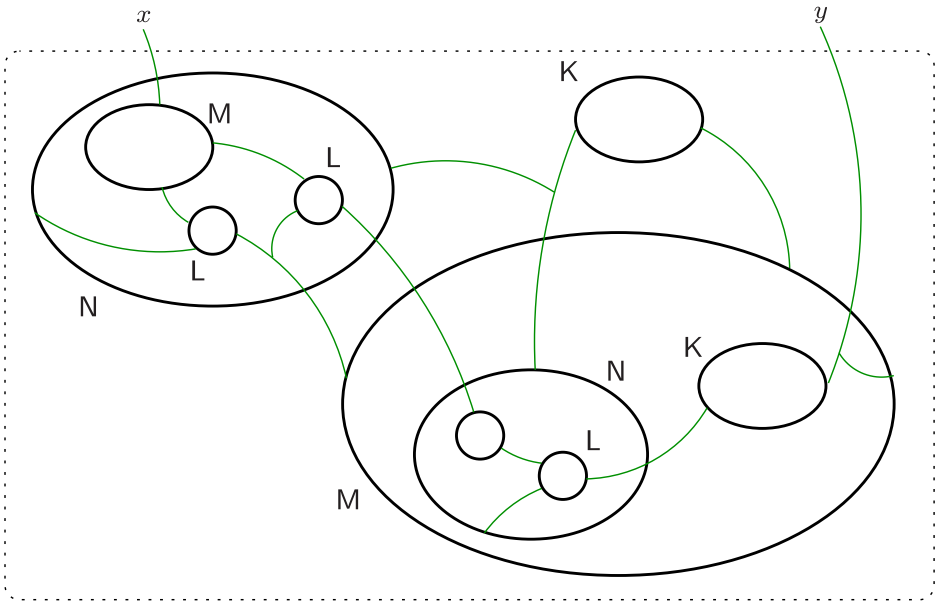


Figure 1: An example of a bigraph

1 Introduction

Process calculi have made progress in modelling interactive concurrent systems [6, 2, 22, 32], systems with mobile connectivity [39, 17] and systems with mobile locality [3, 10]. There is some agreement among all these approaches, both in their basic notions and in their theories; perhaps the strongest feature is a good understanding of behavioural specification and equivalence. At the same time the space of possible calculi is large, we lack a uniform development of their theories, and in particular there is no settled way to combine their various kinds of mobility. As shown by Castellani's [11] comprehensive survey, widely varying notions of locality have been explored, and this implies a similar variety in treating mobility.

There is therefore a dual challenge: first to find a larger common theoretical basis for process calculi, and second to find a common treatment of mobility. The two challenges may appear to be independent, and it would be simpler if that were so; but it appears that mobility is becoming essential to a huge range of applications, so that the search for a common theoretical basis should attend to mobility at the outset if it is not to risk irrelevancy.

The authors' response [30, 29] to the first aspect has been to propose a uniform treatment of transition systems for process calculi, and to erect upon it a uniform behavioural theory. In parallel, the response to the second aspect [37, 25] has been to propose and apply a (topo)graphical process model, known as *bigraphs*, which not only unifies a variety of treatments of mobility, but also underlies process calculi that are not obviously 'mobile'. In other words, it unifies mobility with other computational notions (such as scope and control) that appear separate at first sight. A typical bigraph is shown in Figure 1; it shows how the nesting of nodes (the *places*) is independent of the connectivity (the *links*) among them. Further details are deferred to Section 5.

These twin proposals have been combined in application to the π-calculus [25, 26], the ambient calculus [26, 24] and Petri nets [34], yielding behavioural theory agreeing well with those proposed independently. The theory developed to this point is rather rich; it is therefore appropriate to publish a paper presenting just those parts needed to support one particular case study. The study of Petri nets [34] is a good choice, since it requires just one of the two constituents of bigraphs: *link graphs*. The other constituent, *place graphs*, is not needed since Petri nets involve no nested localities and no notion of the scope of names.

Thus the present paper can serve both as an introduction to the theory and as a test of its value to applications. We present notions independently wherever possible, allowing the effect of different choices to be assessed. One choice we have made deserves special mention; we have adopted an approach based upon s-categories, which are a well-behaved class of precategories. Treating bigraphs as the arrows in an s-category is especially convenient for analysing the notion of *occurrence* of an entity in a bigraph. In Section 11 we compare this with two alternative approaches; one uses a category of graph embeddings and the other uses a 2-category.

Synopsis The rest of this paper is divided into three parts, followed by a concluding section on related and future work.

Part I begins with a view of the theoretical challenge, and then presents a category-theoretic framework for deriving transition systems. The main structural topics are the notion of s-category and the properties of relative pushouts (RPOs) and idem pushouts (IPOs). Reactive systems are introduced by adding reaction rules to the s-categories. Transition systems based upon IPOs are then derived uniformly from these rules, using RPOs. It is proved that, when enough RPOs exist, bisimilarity is a congruence. Part I ends with a study of how a reactive system may be equipped with different transition systems, and how these may be related to one another.

Part II begins with a view of the challenge from mobile applications, including a summary of the bigraphical model of which link graphs are a constituent. It continues with a mathematical formulation of link graphs, including a construction of RPOs and IPOs for them. A central feature is the characterisation of the family of IPOs for any consistent pair of link graphs. Link-graphical reactive systems (LRSs) are then defined, as reactive systems over link graphs. The theory of Part I is applied to derive transition systems for LRSs, for which a congruential bisimilarity is guaranteed. A particular class, the *simple* LRSs, is shown to admit especially simple transition systems.

Part III begins with the concept of *sorting disciplines* for LRSs. A certain class of sorting disciplines allows the transition theory of well-sorted LRSs to be transferred from the unsorted ones, by pulling RPOs back along a forgetful functor. In particular, *many-one* sorting is shown to enjoy this property; it also allows condition-event nets to be represented accurately as an LRS, for which the work of Part II yields a tractable transition system. It is then shown that the corresponding congruential bisimilarity coincides with one that arises from a natural experimental equivalence defined independently of link graphs.

The concluding section, Section 11, discusses related and future research.

Part I
Reactive systems and transition systems

2 The challenge from process theory

In process calculi it is common to present the *dynamics* of processes by means of *reactions* (typically known as rewriting rules) of the form $a \longrightarrow\!\!\triangleright\, a'$, where a and a' are agents. This treatment is often accompanied by *labelled transitions* of the form $a \xrightarrow{\ell}\!\!\triangleright\, a'$, where the label ℓ is drawn from some vocabulary expressing the possible interactions between an agent and its environment. Typically, there is a distinguished label τ such that the labelled transition relation $\xrightarrow{\tau}\!\!\triangleright$ concides with the reaction relation $\longrightarrow\!\!\triangleright$. The full family of labelled transitions has the great advantage that it supports the definition of behavioural preorders and equivalences, such as traces, failures and bisimilarity, and these often turn out to be congruences. But hitherto the labelled transitions have been tailored for each calculus.

We therefore ask whether these labels can be *derived* uniformly from any given set of reaction rules of the form $r \longrightarrow\!\!\triangleright\, r'$, where the *redex* r is an agent that may change its state to the *reactum* r'. A natural approach is to take the labels to be a certain class of (environmental) *contexts*; if L is such a context, the transition $a \xrightarrow{L}\!\!\triangleright\, a'$ implies that a reaction can occur in $L \circ a$ leading to a new state a'. In fact we shall represent agents and contexts as arrows in a category, or more generally a precategory, where the composition $L \circ a$ represents the insertion of agent a in context L. Moreover, we would like to be sure that the behavioural relations —such as bisimilarity— that are determined by the transitions are indeed congruential, i.e. preserved by insertion into any surrounding environment.

But we do not want *all* contexts as labels; as Sewell [49] points out, the behavioural equivalences that result from this choice are unsatisfactory. How to find a satisfactory —and suitably minimal— set of labels, and to do it uniformly, remained an open problem for many years. As a first step, Sewell [49] was able to derive context-labelled transitions uniformly for parametric term-rewriting systems with parallel composition and blocking, and showed bisimilarity to be a congruence. His approach did not handle reactive systems with "connectivity", the (potentially non-linear) sharing of names which arises in many process calculi.

Recently the authors [30] were able to define minimal labels in terms of the categorical notion of *relative pushout* (RPO), and moreover to ensure that behavioural equivalence is a congruence for a wide class of reactive systems. These results were extended and refined in Leifer's PhD Dissertation [29], and Cattani et al [12] applied this theory to action graphs with rich connectivity. Meanwhile, Milner developed the bigraph model [36, 37] from action graphs, with inspiration from the mobile ambients of Cardelli and Gordon. The development was driven by the simplicity that comes from treating locality and connectivity independently, and was also inspired by Gardner's development [19] of *symmetric* action graphs (i.e. with undirected edges).

These applications have motivated the effort to formulate the RPO theory more succinctly [26], in a way that eases both the theory itself and the characterisation of the transition systems to which it gives rise. This is the topic of Part I of our paper. It turns out that these two tasks can be addressed well using a variant of category which we call a *supported precategory*, or *s-category*.

A precategory is a category in which composition is not always defined. It is *supported* if both of the following conditions hold: (a) each arrow f has a *support* $|f|$, a finite set, and (b) the composition $g \circ f$ is defined if and only if $|g| \cap |f| = \emptyset$. This structure makes s-categories remarkably well-behaved. They inherit many notions from categories with no change, and most work is unaffected by the partiality of composition. They also admit direct treatment of the notion of *occurrence* (e.g. of a node in a graph), which in Part II we find essential to the characterisation of behaviour.

In Section 3 we introduce our categorical framework; we then define RPOs and IPOs and derive their properties. This leads in Section 4 to *reactive systems*, and thence to the derivation of *transition systems* based upon IPOs. The central theorem is proved that bisimilarity for these transition systems is a congruence provided enough RPOs exist. The remainder of the section deals with useful relationships among transition systems, in preparation for Part II where we need to refine them by varying their agents or transitions or both.

3 S-categories and relative pushouts

In this section and Section 4 we develop a mathematical framework for the static and dynamic properties of mobile interactive systems. Though abstract, it is developed with a view to underpinning the bigraphical model [37, 25] and its applications. More specifically, to keep the paper well-focussed, the abstract development is only taken far enough to underpin link graphs, which are constituents of bigraphs. These two sections are an adaptation and extension of work started by Leifer and Milner [30], then further developed by Leifer in his PhD Dissertation [29] and by Milner [37].

The reader can perfectly well study Parts II and III independently of Part I, provided he or she is willing to take the main results of Part I on trust and to refer back to important definitions from time to time.

The present section is concerned with the categorical framework and the important concepts, especially relative pushouts, that will underlie the treatment of dynamics in Section 4.

Notation We shall always accent the name of a precategory, as in $\acute{\mathbf{C}}$. We use '$\circ$', 'id' and '$\otimes$' for composition, identity and tensor product. We denote the domain I and codomain J of an arrow $f\colon I \to J$ by $\mathsf{dom}(f)$ and $\mathsf{cod}(f)$; the set of arrows from I to J, called a *homset*, will be denoted by $\acute{\mathbf{C}}(I \to J)$ or just $I \to J$.

Id_S will denote the identity function on a set S, and $\emptyset_S$ the empty function from $\emptyset$ to S. We shall use $S \uplus T$ for union of sets S and T known or assumed to be disjoint, and $f \uplus g$ for union of functions whose domains are known or assumed to be disjoint. This use of $\uplus$ on sets should not be confused with the disjoint sum '$+$', which disjoins sets *before* taking their union. We assume a fixed representation of disjoint sums; for example, $X+Y$ means $(\{0\}\times X)\cup(\{1\}\times Y)$, and $\sum_{v\in V} P_v$ means $\bigcup_{v\in V}(\{v\}\times P_v)$.

We write $f\restriction S$ for the restriction of a function f to the domain S. If R is a binary relation we write $R\restriction S$ for $R\cap S^2$; also if $\equiv$ is an equivalence then we define $R^{\equiv}$ to be the closure of R under $\equiv$ i.e. the relational composition $\equiv R\equiv$.

A natural number m is often interpreted as a finite ordinal $m=\{0,1,\ldots,m-1\}$. We often use i to range over m; when $m=2$ we use $\bar{\imath}$ for the complement $1-i$ of i. We use $\vec{x}$ to denote a sequence $\{x_i \mid i\in m\}$; when $m=2$ this is an ordered pair. ■

Definition 3.1 (precategory, functor) A *precategory* $\acute{\mathbf{C}}$ is defined exactly as a category, except that the composition of arrows is not always defined. Composition with the identities is always defined, and $\mathsf{id}\circ f = f = f\circ\mathsf{id}$. In the equation $h\circ(g\circ f) = (h\circ g)\circ f$, the two sides are either equal or both undefined.

A *subprecategory* $\acute{\mathbf{D}}$ of $\acute{\mathbf{C}}$ is defined like a subcategory, with $g\circ f$ defined in $\acute{\mathbf{D}}$ exactly when defined in $\acute{\mathbf{C}}$. A *functor* $\mathcal{F} : \acute{\mathbf{D}}\to\acute{\mathbf{C}}$ between precategories is a total function on objects and on arrows that preserves identities and composition, in the sense that if $g\circ f$ is defined in $\acute{\mathbf{D}}$, then $\mathcal{F}(g)\circ\mathcal{F}(f)=\mathcal{F}(g\circ f)$ in $\acute{\mathbf{C}}$. ■

In general we shall use $I, J, K, \ldots$ to stand for objects and $f, g, h, \ldots$ for arrows. We shall extend category-theoretic concepts to precategories without comment when they are unambiguous. One concept which we now extend explicitly is that of a monoidal category:

Definition 3.2 (tensor product, monoidal precategory, monoidal functor) A *(strict, symmetric) monoidal* precategory has a partial *tensor product* $\otimes$ both on objects and on arrows. It has a unit object ϵ, called the *origin*, such that $I\otimes\epsilon=\epsilon\otimes I=I$ for all I. Given $I\otimes J$ and $J\otimes I$ it also has a *symmetry* isomorphism $\gamma_{I,J} : I\otimes J\to J\otimes I$. The tensor and symmetries satisfy the following equations when both sides exist:

(1) $f\otimes(g\otimes h)=(f\otimes g)\otimes h$ and $\mathsf{id}_\epsilon\otimes f=f$

(2) $(f_1\otimes g_1)\circ(f_0\otimes g_0)=(f_1\circ f_0)\otimes(g_1\circ g_0)$

(3) $\gamma_{I,\epsilon}=\mathsf{id}_I$

(4) $\gamma_{J,I}\circ\gamma_{I,J}=\mathsf{id}_{I\otimes J}$

(5) $\gamma_{I,K}\circ(f\otimes g)=(g\otimes f)\circ\gamma_{H,J}$ (for $f : H\to I,\ g : J\to K$) .

A *monoidal* functor is one that preserves tensor product and origin. ■

Note that the symmetric identity law $f\otimes\mathsf{id}_\epsilon = f$ is provable from (1), (3) and (5). 'Strict' means that associativity holds exactly, as stated, not merely up to isomorphism; 'symmetric' refers to the symmetry isomorphisms satisfying equations (3)–(5). We shall omit 'strict' and 'symmetric' from now on, as we shall always assume these properties.

Why do we wish to work in precategories? In the introduction we pointed out that the dynamic theory of bigraphs will require the existence of relative pushouts (RPOs). This means that we need to develop the theory first for *concrete* bigraphs, those in which nodes have identity; then we can transfer the results to *abstract* graphs by the quotient functor that forgets this identity. Precategories allow a direct presentation of concrete bigraphs; for we can stipulate that two bigraphs F and G may be composed to form $H = G \circ F$ only if their node sets are disjoint. We can think of this composition as as *keeping track* of nodes[3]; we can see in H exactly which nodes come from F and which from G.

More generally, we are interested in monoidal precategories where the definedness of composition and of tensor product depends upon a *support* set associated with each arrow. In bigraphs the support of an arrow will be its node set. In general we assume support to be drawn from some unspecified infinite set. We now give a general definition of precategories $\acute{\mathbf{C}}$ with support; we continue to use this accented notation for them, dropping the accent only when we have a category.

Definition 3.3 ((monoidal) s-category) We say that a precategory $\acute{\mathbf{C}}$ is *supported*, or *an s-category*, if it has:

- for each arrow f, a finite set $|f|$ called its *support*, such that $|\mathsf{id}_I| = \emptyset$. The composition $g \circ f$ is defined iff $|g| \cap |f| = \emptyset$ and $\mathsf{dom}(g) = \mathsf{cod}(f)$; then $|g \circ f| = |g| \uplus |f|$.

- for any arrow $f : I \to J$ and any injective map ρ whose domain includes $|f|$, an arrow $\rho \bullet f : I \to J$ called a *support translation* of f such that

$$
\begin{array}{ll}
(1) & \rho \bullet \mathsf{id}_I = \mathsf{id}_I \\
(2) & \rho \bullet (g \circ f) = \rho \bullet g \circ \rho \bullet f \\
(3) & \mathsf{Id}_{|f|} \bullet f = f \\
(4) & (\rho_1 \circ \rho_0) \bullet f = \rho_1 \bullet (\rho_0 \bullet f) \\
(5) & \rho \bullet f = (\rho \restriction |f|) \bullet f \\
(6) & |\rho \bullet f| = \rho(|f|) \,.
\end{array}
$$

If $\acute{\mathbf{C}}$ is monoidal as a precategory, it is a *monoidal* s-category if, for $f \colon H \to I$ and $g \colon J \to K$, their tensor product $f \otimes g$ is defined exactly when $H \otimes J$ and $I \otimes K$ exist and $|f| \cap |g| = \emptyset$, and then the product satisfies $|f \otimes g| = |f| \uplus |g|$ and

$$(7) \;\; \rho \bullet (f \otimes g) = \rho \bullet f \otimes \rho \bullet g \,.$$

Each of these seven equations is required to hold only when both sides are defined. ■

Exercises Deduce condition (1) from conditions (5) and (3). Prove that every isomorphism has empty support. Show that in conditions (2) and (7) either both sides are defined or both are undefined.

We now consider functors between s-categories.

Definition 3.4 (support equivalence, supported functor) Let $\acute{\mathbf{A}}$ be an s-category. Two arrows $f, g : I \to J$ in $\acute{\mathbf{A}}$ are *support-equivalent*, written $f \simeq g$, if $\rho \bullet f = g$

[3] Leifer's development [29] (see Chapter 7) made use of a special category $\mathsf{Track}(\acute{\mathbf{C}})$ to keep track of nodes in a precategory $\acute{\mathbf{C}}$. This allowed the theory of RPOs to be developed for categories rather than for precategories. However, it can be developed more succinctly if we stay with the latter.

for some support translation ρ. By Definition 3.3 this is an equivalence relation. If $\acute{\mathbf{B}}$ is another supported precategory, then a functor $\mathcal{F}\colon \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ is called *supported* if it preserves support equivalence, i.e. $f \overset{\frown}{-} g$ implies $\mathcal{F}(f) \overset{\frown}{-} \mathcal{F}(g)$. ■

When we no longer need to keep track of support we may use a quotient *category* (not just s-category). To define such quotients, we need the following notion:[4]

Definition 3.5 (static congruence) Let $\equiv$ be an equivalence defined on every homset of a supported precategory $\acute{\mathbf{C}}$. We call $\equiv$ a *static* (*monoidal*) *congruence on* $\acute{\mathbf{C}}$ if it is preserved by composition (and by tensor product), namely: if $f \equiv f'$ and $g \equiv g'$ then $f \circ g \equiv f' \circ g'$ whenever the latter are defined (and likewise for tensor product). ■

As an example of a simple static congruence on link graphs, we may define $f \equiv f'$ to mean that f and f' are identical when all K-nodes are discarded, for some particular control K. (See Section 6 for the definitions of controls and link graphs.)

The most important example of a static congruence will be support equivalence ($\overset{\frown}{-}$). But the following definition shows that any static congruence at least as coarse as support equivalence will yield a well-defined quotient category:

Definition 3.6 (quotient categories) Let $\acute{\mathbf{C}}$ be an s-category, and let $\equiv$ be a static (monoidal) congruence on $\acute{\mathbf{C}}$ that includes support equivalence, i.e. $\overset{\frown}{-} \subseteq \equiv$. Then the *quotient* of $\acute{\mathbf{C}}$ by $\equiv$ is a category $\mathbf{C} \overset{\text{def}}{=} \acute{\mathbf{C}}/{\equiv}$, whose objects are the objects of $\acute{\mathbf{C}}$ and whose arrows are equivalence classes of arrows in $\acute{\mathbf{C}}$:

$$\mathbf{C}(I,J) \overset{\text{def}}{=} \{\, [f]_{\equiv} \mid f \in \acute{\mathbf{C}}(I,J) \,\} \,.$$

In $\mathbf{C}$, identities and composition (and tensor product when $\acute{\mathbf{C}}$ has it) are given by

$$\begin{aligned} \mathsf{id}_m &\overset{\text{def}}{=} [\mathsf{id}_m]_{\equiv} \\ [f]_{\equiv} \circ [g]_{\equiv} &\overset{\text{def}}{=} [f \circ g]_{\equiv} \\ [f]_{\equiv} \otimes [g]_{\equiv} &\overset{\text{def}}{=} [f \otimes g]_{\equiv} \,. \end{aligned}$$

By assigning empty support to every arrow we may also regard $\mathbf{C}$ as an s-category, so that $[\cdot]_{\equiv} : \acute{\mathbf{C}} \to \mathbf{C}$ is a special supported functor called the $\equiv$*-quotient functor* for $\acute{\mathbf{C}}$. ■

Note that the quotient does not affect objects; thus any tensor product on $\mathbf{C}$ may still be partial on objects. But $\mathbf{C}$ is indeed a category; composition is always well-defined because $f \overset{\frown}{-} g$ implies $f \equiv g$, and so also is tensor product provided it is defined on the domains and codomains.

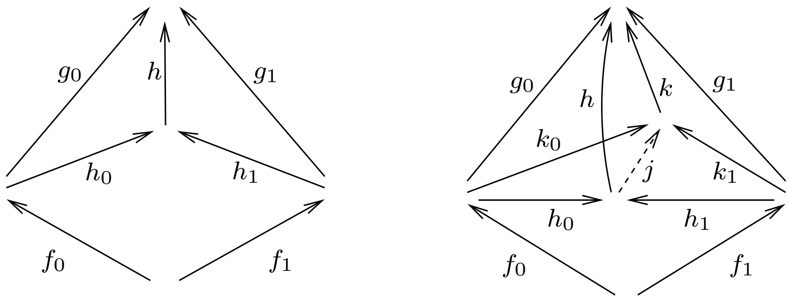
We often abbreviate $[\cdot]_{\overset{\frown}{-}}$ to $[\cdot]$; we call it the *support quotient functor*. From the definition, clearly a coarser quotient $[\cdot]_{\equiv}$ exists whenever $\equiv$ is a congruence that includes support equivalence. In Part II we shall define a coarser quotient functor by this means.

We now turn to the notion of relative pushout (RPO), which is of crucial importance in defining labelled transitions in the following section.

[4]We use the term *static* congruence to emphasize that these congruences depend only on static structure, in contrast with *dynamic* congruences such as bisimilarity, which depend upon transitions.

Notation In what follows we shall frequently use $\vec{f}$ to denote a pair f_0, f_1 of arrows in a precategory. If, for example, the two arrows share a domain H and have codomains I_0, I_1 we write $\vec{f} : H \to \vec{I}$.

Definition 3.7 (bound, consistent) If two arrows $\vec{f} : H \to \vec{I}$ share domain H, the pair $\vec{g} : \vec{I} \to K$ share codomain K and $g_0 \circ f_0 = g_1 \circ f_1$, then we say that $\vec{g}$ is a *bound* for $\vec{f}$. If $\vec{f}$ has any bound, then it is said to be *consistent*. ■



Definition 3.8 (relative pushout) In a precategory, let $\vec{g} : \vec{I} \to K$ be a bound for $\vec{f} : H \to \vec{I}$. A *bound for* $\vec{f}$ *relative to* $\vec{g}$ is a triple $(\vec{h}, h)$ of arrows such that $\vec{h}$ is a bound for $\vec{f}$ and $h \circ h_i = g_i$ $(i = 0, 1)$. We may call the triple a *relative bound* when $\vec{g}$ is understood.

A *relative pushout* (*RPO*) *for* $\vec{f}$ *relative to* $\vec{g}$ is a relative bound $(\vec{h}, h)$ such that for any other relative bound $(\vec{k}, k)$ there is a unique arrow j for which $j \circ h_i = k_i$ $(i = 0, 1)$ and $k \circ j = h$.

We say that a precategory *has RPOs* if, whenever $\vec{f}$ has a bound, it also has an RPO relative to that bound. ■

We shall often omit the word 'relative'; for example we may call $(\vec{h}, h)$ a bound (or RPO) for $\vec{f}$ to $\vec{g}$.

The more familiar notion, a pushout, is a bound $\vec{h}$ for $\vec{f}$ such that *for any* bound $\vec{g}$ there exists an h which makes the left-hand diagram commute. Thus a pushout is a *least* bound, while an RPO provides a *minimal* bound relative to a given bound $\vec{g}$. In Section 6 we find that RPOs exist for link graphs in cases where there is no pushout.

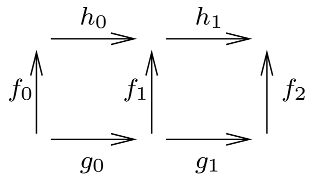
Now suppose that we can create an RPO $(\vec{h}, h)$ for $\vec{f}$ to $\vec{g}$; what happens if we try to iterate the construction? More precisely, is there an RPO for $\vec{f}$ to $\vec{h}$? The answer lies in the following important concept:

Definition 3.9 (idem pushout) In a precategory, if $\vec{f} : H \to \vec{I}$ is a pair of arrows with common domain, then a pair $\vec{h} : \vec{I} \to J$ is an *idem pushout* (*IPO*) for $\vec{f}$ if $(\vec{h}, \mathsf{id}_J)$ is an RPO for $\vec{f}$ to $\vec{h}$. ■

Then it turns out that the attempt to iterate the RPO construction will yield the *same* bound (up to isomorphism); intuitively, the minimal bound for $\vec{f}$ to any bound $\vec{g}$ is reached in just one step. This is a consequence of the first two parts of the following proposition, which summarises the essential properties of RPOs and IPOs on which our work relies. They are proved for categories in Leifer's Dissertation [29] (see also Leifer and Milner [30]), and the proofs can be routinely adapted for precategories.[5]

Proposition 3.10 (properties of RPOs) *In any precategory* $\acute{\mathbf{A}}$*:*

1. *If an RPO for* $\vec{f}$ *to* $\vec{g}$ *exists, then it is unique up to isomorphism.*
2. *If* $(\vec{h}, h)$ *is an RPO for* $\vec{f}$ *to* $\vec{g}$*, then* $\vec{h}$ *is an IPO for* $\vec{f}$*.*
3. *If* $\vec{h}$ *is an IPO for* $\vec{f}$*, and an RPO exists for* $\vec{f}$ *to* $h \circ h_0, h \circ h_1$*, then the triple* $(\vec{h}, h)$ *is such an RPO.*
4. *(IPO pasting) Suppose that the diagram below commutes, and that* f_0, g_0 *has an RPO to the pair* $h_1 \circ h_0, f_2 \circ g_1$*. Then*
   - *if the two squares are IPOs, so is the big rectangle;*
   - *if the big rectangle and the left square are IPOs, so is the right square.*



5. *(IPO sliding) If* $\acute{\mathbf{A}}$ *is an s-category then IPOs are preserved by support translation; that is, if* $\vec{g}$ *is an IPO for* $\vec{f}$ *and* ρ *is any injective map whose domain includes the supports of* $\vec{f}$ *and* $\vec{g}$*, then* $\rho \cdot \vec{g}$ *is an IPO for* $\rho \cdot \vec{f}$*.*

We now consider a property of RPOs which may not be present in all precategories, but will be enjoyed by link graphs. We know that the RPO status of a triple is preserved by isomorphism at its mediating interface, i.e. if $(\vec{h}, h)$ is an RPO then so is $(i \circ \vec{h}, h \circ j)$ where i, j is an iso. But can RPO status be retained by keeping $\vec{h}$ fixed and varying h? If not we say that the RPO is rigid. Formally:

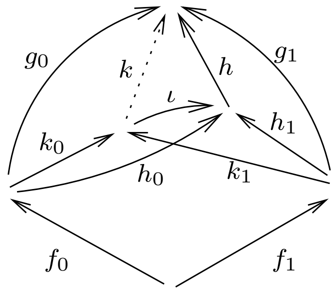
Definition 3.11 (rigid RPO) An RPO $(\vec{h}, h)$ for $\vec{f}$ to $\vec{g}$ is *rigid* if, whenever $(\vec{h}, k)$ is another RPO for $\vec{f}$ to $\vec{g}$, then $k = h$. ■

Exercises Prove that if $\vec{f}$ has a rigid RPO relative to $\vec{g}$, then all its RPOs relative to $\vec{g}$ are rigid. More difficult: find a category in which there is a non-rigid RPO. (These exercises are not needed for what follows.)

In Section 6 we shall show that every link graph RPO is rigid. This in useful, since we can then deduce from the following proposition that, in link graphs, a unique IPO is a pushout.

Proposition 3.12 (unique IPOs are pushouts) *Let* $\vec{f}$ *have a rigid RPO relative to some bound. Then an IPO for* $\vec{f}$ *that is unique up to isomorphism is a pushout.*

[5]This adaptation works for the definition of precategory in Definition 3.1, which is satisfied by our supported precategories.



Proof Let $\vec{k}$ be an IPO for $\vec{f}$, and let $\vec{g}$ be any bound. Under the assumptions we must find a unique mediator k such that $k \circ k_i = g_i$ $(i = 0, 1)$.

Take a rigid RPO $(\vec{h}, h)$ for $\vec{f}$ to $\vec{g}$. Then $\vec{h}$ is an IPO by Proposition 3.10(2); hence by assumption there is an isomorphism ι as shown such that $\iota \circ k_i = h_i$ $(i = 0, 1)$. Then $h \circ \iota$ satisfies the required property of the mediator k.

Now let k be any such mediator, and let ι' be the inverse of ι. Then $(k \circ \iota') \circ h_i = k \circ \iota' \circ \iota \circ k_i = k \circ k_i = g_i$ $(i = 0, 1)$. It follows from Proposition 3.10(3) that $(\vec{h}, k \circ \iota')$ is an RPO for $\vec{f}$ to $\vec{g}$. But $(\vec{h}, h)$ is rigid by assumption, hence $k \circ \iota' = h$. So finally $k = h \circ \iota$, showing that the mediator $h \circ \iota$ is unique as required. ∎

4 Reactive and transition systems

We now introduce a kind of dynamical system, of which link graphs will be an instance. In previous work [30, 29] a notion of reactive system was defined. In our present terms, this consists first of a monoidal s-category whose arrows are called *contexts*. The objects $I, J, \ldots$ will be called *interfaces*. We adopt a change of notation from the preceding section: we shall now use upper case $A, B, C, \ldots$ for arbitrary arrows. A composition $C \circ A$ represents placing A in the context C.

Contexts $C : \epsilon \to I$ with the origin as domain are in a sense trivial, since in this case we have $C \circ A = C \otimes A$. We shall call a context *ground* if its domain is the origin, and use lower case $a, b, c, \ldots$ for ground arrows. We write $a : I$ for $a : \epsilon \to I$, and $\mathsf{Gr}(I)$ for the homset $\epsilon \to I$.

The second ingredient of a reactive system in [30, 29] was a set of ground pairs (r, r') called *reaction rules*, and a subprecategory of so-called *active* contexts. The reaction relation $\longrightarrow\!\triangleright$ between agents was taken to be the smallest such that $D \circ r \longrightarrow\!\triangleright D \circ r'$ for every active context D and reaction rule (r, r').

For such systems we uniformly derived labelled transitions based upon IPOs. Several behavioural preorders and equivalences based upon these transitions, including bisimilarity, were shown to be congruences, subject to two conditions: first, that sufficient RPOs exist in the s-category; second, that decomposition preserves activity — i.e. $D \circ C$ active implies both C and D active. In subsequent work, sufficient RPOs were found in interesting cases (Leifer [29], Cattani et al [12]).

The present section is essentially a reformulation of the work in [30, 29]. However, we omit the notion of 'active' context since it does not apply to link graphs (where *every* context is active); we also simplify the treatment of functors between reactive systems.

We are now ready to define reactive systems:

Definition 4.2 (reactive system) A *reactive system (RS)* is a supported monoidal precategory $\acute{\mathbf{A}}$ equipped with a set $\acute{\mathcal{R}}$ of *reaction rules* of the form $(r : I, r' : I)$, in which r is the *redex* and r' the *reactum*. We require $\acute{\mathcal{R}}$ to be closed under support equivalence, i.e. if (r, r') is a rule then so is (s, s') whenever $r \simeq s$ and $r' \simeq s'$.

The *reaction relation* $\longrightarrow\!\!\triangleright$ over ground arrows is the smallest closed both sides under support equivalence, and such that $D \circ r \longrightarrow\!\!\triangleright D \circ r'$ whenever (r, r') is a reaction rule and D a context. ■

We denote this RS by $\acute{\mathbf{A}}(\acute{\mathcal{R}})$, or just $\acute{\mathbf{A}}$ when $\acute{\mathcal{R}}$ is understood. Closing the reaction rules under support equivalence allows us in Definition 4.5 to divide $\acute{\mathbf{A}}$ by $\simeq$, forming a quotient RS.

To close $\acute{\mathcal{R}}$ under support equivalence is a significant decision. Recall that we have adopted the notion of support in concrete link graphs, or bigraphs, so that nodes have identity; this enables us to construct RPOs (which would otherwise not exist) and thence to derive transitions, as we shall see shortly. For this derivation it was not necessary that node-identity should persist through a reaction. Our closure condition prevents this persistent identity; we adopt it order to capture in bigraphs the standard behavioural equivalences in process calculi, where there is no notion of tracking the identity of components through reaction.

An alternative decision merits close attention. It would replace the closure condition by a more modest one: that if (r, r') is a reaction rule then so is $(\rho \cdot r, \rho \cdot r')$. It therefore respects the transmission of the identity of nodes from r to r'. One important use of this is to admit logical analysis in the style of Caires and Cardelli [8], using spatio-temporal assertions like "here there will always be a K-node". We leave this promising avenue of research to the future.

We extend the notion of functor $\mathcal{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ to RSs, requiring it to preserve reaction. Recall from Definition 3.4 that a supported functor is one that preserves support equivalence.

Definition 4.3 (RS functor, sub-RS) A supported monoidal functor $\mathcal{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ of monoidal s-categories is an *RS functor* if it preserves reaction rules, i.e. if (r, r') is a rule of $\acute{\mathbf{A}}$ then $(\mathcal{F}(r), \mathcal{F}(r'))$ is a rule of $\acute{\mathbf{B}}$. If $\mathcal{F}$ is injective on objects and arrows then we call $\acute{\mathbf{A}}$ a *sub-RS* of $\acute{\mathbf{B}}$. ■

Proposition 4.4 (RS functors preserve reaction) *An RS functor $\mathcal{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ preserves reaction, i.e. if $g \longrightarrow\!\!\triangleright g'$ in $\acute{\mathbf{A}}$ then $\mathcal{F}(g) \longrightarrow\!\!\triangleright \mathcal{F}(g')$ in $\acute{\mathbf{B}}$.*

Clearly RSs and their functors form a category. An important example of a functor is the support quotient functor, extended to RSs as follows:

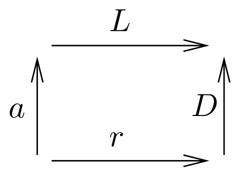
Definition 4.5 (quotient RS) Let $\acute{\mathbf{A}}$ be a reactive system equipped with $\acute{\mathcal{R}}$. Then its *support quotient* reactive system is based upon the support quotient $\mathbf{A} = \acute{\mathbf{A}}/\simeq$. Its reaction rules are $\{([r], [r']) \mid (r, r') \in \acute{\mathcal{R}}\}$. ■

Proposition 4.6 (quotient reflects reaction) *The support quotient functor $[\cdot] : \acute{\mathbf{A}} \to \mathbf{A}$ both preserves and reflects reaction, i.e. $[g] \longrightarrow\!\!\triangleright [g']$ in $\mathbf{A}$ iff $g \longrightarrow\!\!\triangleright g'$ in $\acute{\mathbf{A}}$.*

The quotient functor takes a *concrete* RS, based on an s-category, to an *abstract* RS based upon a category. Later we show how to obtain a behavioural congruence for an arbitrary concrete RS $\acute{\mathbf{A}}$ with sufficient RPOs. The support quotient $\mathbf{A}$ of $\acute{\mathbf{A}}$ may not possess RPOs, but nevertheless the quotient functor allows us to derive a behavioural congruence for $\mathbf{A}$ also. This use of a concrete RS with RPOs to supply a behavioural congruence for the corresponding abstract RS was first represented by the *functorial reactive systems* of Leifer's thesis [29].

We now consider how to equip an RS with labelled transitions. Conventionally, a labelled transition takes the form $a \xrightarrow{\ell} a'$, where a, a' are agents and the label ℓ comes from some explicitly defined set. Here we shall study *contextual* transitions, in which the labels are contexts into which agents may be inserted; these are in contrast with *raw* transitions where the label set is defined by other means.

Traditionally (for example in CCS) transitions were raw, and defined independently of, or even in preference to, reaction rules. But the latter are conceptually simpler, so it is natural to take them —rather than transitions— as primitive. Given a reactive system, we have previously [30] defined a labelled transition to be a triple written $a \xrightarrow{L} a'$ for which there is a reaction rule (r, r') and an 'active' context D such that (L, D) is an idem pushout (IPO) for (a, r) and $a' = D \circ r'$. We shall adopt this, except that we do not always require an IPO, nor do we impose an activeness condition:



Definition 4.7 (transition) A *(contextual) transition* is a triple written $a \xrightarrow{L} a'$, where a and a' are ground, L is a context, and there exist a reaction rule (r, r') and a context D such that the diagram commutes and $a' \simeq D \circ r'$. We say that the reaction rule and the diagram *underlie* the transition. A transition is *minimal* if the underlying diagram is an IPO. ■

For a fixed reactive system many different sets of transitions may be considered, according to the agents that we wish to observe, and the experiments —represented by labels— that we wish to perform upon them. This leads to the following:

Definition 4.8 (transition system) Given an RS $\acute{\mathbf{A}}$, a *(labelled) transition system* $\mathcal{L}$ for $\acute{\mathbf{A}}$ is a pair $(\mathsf{Int}_{\mathcal{L}}, \mathsf{Trans}_{\mathcal{L}})$, where

- $\mathsf{Int}_{\mathcal{L}}$ is a set of interfaces called the *agent interfaces*; the *agents* of $\mathcal{L}$ are defined as $\mathsf{Ag}_{\mathcal{L}} \stackrel{\text{def}}{=} \{a : I \mid I \in \mathsf{Int}_{\mathcal{L}}\}$.
- $\mathsf{Trans}_{\mathcal{L}}$ is a set of transitions whose sources and targets are agents of $\mathcal{L}$; the *labels* of $\mathcal{L}$ are those that appear in some transition of $\mathsf{Trans}_{\mathcal{L}}$.

The *full* (resp. *standard*) transition system for an RS consists of all interfaces, together with all (resp. all minimal) transitions. When the RS is understood we shall denote these two transition systems respectively by FT and ST.

We abbreviate '(labelled) transition system' to TS. Another transition system $\mathcal{M}$ is a *sub-TS* of $\mathcal{L}$, written $\mathcal{M} \prec \mathcal{L}$, if $\mathsf{Int}_{\mathcal{M}} \subseteq \mathsf{Int}_{\mathcal{L}}$ and $\mathsf{Trans}_{\mathcal{M}} \subseteq \mathsf{Trans}_{\mathcal{L}}$. ■

Whether transitions are derived from reactions or defined in some other way, we may use them to define behavioural equivalences and preorders. We are also interested conditions under which these behavioural relations are congruential, i.e. preserved by context. Here we shall limit attention to strong bisimilarity. (Throughout this paper we shall omit 'strong' since we do not define or use weak bisimilarity.)

Definition 4.9 (bisimilarity, congruence) Let $\acute{\mathbf{A}}$ be a reactive system equipped with a TS $\mathcal{L}$. A *simulation on* $\mathcal{L}$ is a binary relation $\mathcal{S}$ between agents with equal interface such that if $a\mathcal{S}b$ and $a \xrightarrow{L} a'$ in $\mathcal{L}$, then whenever $L \circ b$ is defined there exists b' such that $b \xrightarrow{L} b'$ in $\mathcal{L}$ and $a'\mathcal{S}b'$. A *bisimulation* is a symmetric simulation. Then *bisimilarity on* $\mathcal{L}$, denoted by $\sim_{\mathcal{L}}$, is the largest bisimulation on $\mathcal{L}$.

We say that bisimilarity on $\mathcal{L}$ is a *congruence* if

$$a \sim_{\mathcal{L}} b \Rightarrow C \circ a \sim_{\mathcal{L}} C \circ b$$

for all $a, b : I$ and $C : I \to J$, where $I, J \in \mathsf{Int}_{\mathcal{L}}$. ■

We shall often omit 'on $\mathcal{L}$', and write $\sim$ for $\sim_{\mathcal{L}}$, when $\mathcal{L}$ is understood from the context. This will usually be when $\mathcal{L}$ is ST.

Note the slight departure from the usual definition of bisimulation of Park [41]; since we are in an s-category we must require $L \circ b$ to be defined. This is merely a technical detail, provided that the TS respects support translation; for then, whenever $L \circ a$ is defined there will always exist $L' \doteq L$ for which *both* $L' \circ a$ and $L' \circ b$ are defined. If we are working in a category, in particular if it is a support quotient category, then the side-condition holds automatically and the definition of bisimilarity reduces to the standard one.

We define bisimilarity only for ground link graphs. As a consequence, if bisimilarity is a congruence then it is also preserved by tensor product; that is, if $a \sim_{\mathcal{L}} b$ then $a \otimes c \sim_{\mathcal{L}} b \otimes c$. To see this, note that $a \otimes c = (\mathsf{id} \otimes c) \circ a$.

Definition 4.10 (respect) Let $\equiv$ be a static congruence (Definition 3.5) in an RS equipped with $\mathcal{L}$. Suppose that for every transition $a \xrightarrow{L} a'$ in $\mathcal{L}$, if $a \equiv b$ and $L \equiv M$ for another label M of $\mathcal{L}$ with $M \circ b$ defined, then there exist an agent b' and a transition $b \xrightarrow{M} b'$ in $\mathcal{L}$ such that $a' \equiv b'$. Then $\equiv$ and $\mathcal{L}$ are said to *respect* one another. ■

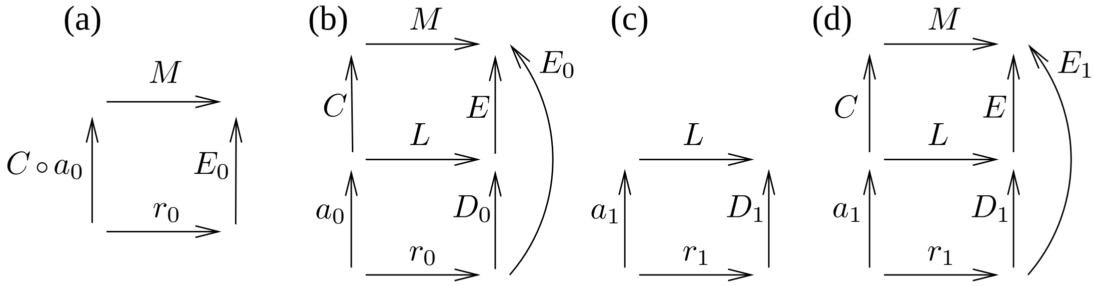
Note that 'respect' is mutual between an equivalence and a TS, so that '$\mathcal{L}$ respects $\equiv$ ' means the same as '$\equiv$ respects $\mathcal{L}$'; we shall use them interchangeably.

It is well known [32] that if $\equiv$ is included in (strong) bisimilarity, then to establish bisimilarity it is enough exhibit a *bisimulation up to* $\equiv$; that is, a symmetric relation $\mathcal{S}$ such that whenever $a\mathcal{S}b$ then each transition of a is matched by b in $\mathcal{S}^{\equiv}$. We now deduce from Proposition 3.10(5) that support equivalence can be used in this way:

Proposition 4.11 (support translation of transitions) *In a reactive system* $\acute{\mathbf{A}}$ *the full and standard transition systems respect support equivalence. Hence in each case* $\doteq$ *is a bisimulation, and a bisimulation up to* $\doteq$ *suffices to establish bisimilarity.*

We may now prove our main congruence theorem for RSs, asserting that ST ensures bisimulation congruence. The reader can deduce the (more obvious!) result that FT ensures the same; simply replace the word 'IPO' by'commuting square' in the proof.

Theorem 4.12 (congruence of bisimilarity) *In a reactive system with RPOs, equipped with the standard transition system, bisimilarity of agents is a congruence; that is, if* $a_0 \sim a_1$ *then* $C \circ a_0 \sim C \circ a_1$.



Proof The proof is along the lines of Theorem 3.9 in Leifer [29]. We establish the following as a bisimulation up to $\simeq$:

$$\mathcal{S} \stackrel{\text{def}}{=} \{(C \circ a_0, C \circ a_1) \mid a_0 \sim a_1\} .$$

Suppose that $a_0 \sim a_1$, and that $C \circ a_0 \xrightarrow{M} b'_0$, for some label M such that $M \circ C \circ a_1$ is defined. It is enough to find b'_1 such that $C \circ a_1 \xrightarrow{M} b'_1$ and $(b'_0, b'_1) \in \mathcal{S}^{\simeq}$.

There exist a reaction rule (r_0, r'_0) and a context E_0 such that diagram (a) is an IPO; moreover $b'_0 \simeq E_0 \circ r'_0$. Then because consistent pairs have RPOs, there exists an RPO for (a_0, r_0) relative to the given bound, and using Proposition 3.10(4) we can complete diagram (b) so that each square is an IPO.

So the lower square of (b) underlies a transition $a_0 \xrightarrow{L} a'_0$, where $a'_0 = D_0 \circ r'_0$. Now $L \circ a_1$ is defined (since $M \circ C \circ a_1$ is defined and $M \circ C = E \circ L$) and $a_0 \sim a_1$, so there is a transition $a_1 \xrightarrow{L} a'_1$ with $a'_0 \sim a'_1$. But support translation of a'_1 preserves both of these properties; so we may assume a rule (r_1, r'_1) and context D_1 such that diagram (c) is an IPO, $a'_1 = D_1 \circ r'_1$ and $|E| \cap |a'_1| = \emptyset$.

Now replace the lower square of (b) by diagram (c), obtaining diagram (d) in which, by Proposition 3.10(4), the large square is an IPO. Hence, setting $E_1 \stackrel{\text{def}}{=} E \circ D_1$, we have $C \circ a_1 \xrightarrow{M} b'_1$ where $b'_1 \stackrel{\text{def}}{=} E_1 \circ r'_1$. Finally $(b'_0, b'_1) \in \mathcal{S}^{\simeq}$ as required, because $b'_0 \simeq E \circ a'_0$ and $b'_1 \simeq E \circ a'_1$ with $a'_0 \sim a'_1$. ■

We should remark that we are taking (strong) bisimilarity as a representative of many preorders and equivalences; Leifer [29] has proved congruence theorems for several others, and we expect that those results can be transferred to the present setting.

Now, if an RS is equipped with a TS we wish to define transitions for various quotient RSs. For this purpose, it is useful to extend a functor in the obvious way to sets and tuples of objects and arrows. Thus, for example, on transitions we have $\mathcal{F}(a \xrightarrow{L} a') = \mathcal{F}(a) \xrightarrow{\mathcal{F}(L)} \mathcal{F}(a')$.

Definition 4.13 (functors respecting, inducing transitions) Let $\mathcal{F}\colon \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ be an RS functor, and let $\acute{\mathbf{A}}$ be equipped with a TS $\mathcal{L}$. We say that $\mathcal{F}$ *respects* $\mathcal{L}$ if the static congruence it induces on $\acute{\mathbf{A}}$ respects $\mathcal{L}$. We call $\mathcal{F}(\mathcal{L})$ the TS *induced* on $\acute{\mathbf{B}}$ by $\mathcal{F}$. ■

This definition always makes sense, but it will not always make bisimilarity a congruence in $\acute{\mathbf{B}}$, even if it is so in $\acute{\mathbf{A}}$. However, the next theorem shows that congruence of bisimilarity is preserved when we quotient by any static congruence that includes support equivalence. Recall that a *full* functor is surjective for each homset.

Theorem 4.14 (functors on bisimilarity) *Let $\acute{\mathbf{A}}$ be equipped with a TS $\mathcal{L}$. Let $\mathcal{F}$ be a full RS functor from $\acute{\mathbf{A}}$ to $\acute{\mathbf{B}}$ that is the identity on objects and respects $\mathcal{L}$, and such that $a \doteq b$ implies $\mathcal{F}(a) = \mathcal{F}(b)$. Then the following hold for $\mathcal{F}(\mathcal{L})$:*

1. *$a \sim_{\mathcal{L}} b$ in $\acute{\mathbf{A}}$ iff $\mathcal{F}(a) \sim_{\mathcal{F}(\mathcal{L})} \mathcal{F}(b)$ in $\acute{\mathbf{B}}$.*
2. *If $\sim_{\mathcal{L}}$ is a congruence in $\acute{\mathbf{A}}$ then $\sim_{\mathcal{F}(\mathcal{L})}$ is a congruence in $\acute{\mathbf{B}}$.*

Proof (1) ($\Rightarrow$) We establish in $\acute{\mathbf{B}}$ the bisimulation

$$\mathcal{R} = \{(\mathcal{F}(a), \mathcal{F}(b)) \mid a \sim_{\mathcal{L}} b\} .$$

Let $a \sim_{\mathcal{L}} b$ in $\acute{\mathbf{A}}$, and let $p = \mathcal{F}(a)$, $q = \mathcal{F}(b)$ and $p \xrightarrow{M} p'$ in $\acute{\mathbf{B}}$, with $M \circ q$ defined. Then by definition of the induced TS we can find L and a' such that $M = \mathcal{F}(L)$ and $p' = \mathcal{F}(a')$, and $a \xrightarrow{L} a'$ in $\acute{\mathbf{A}}$ with $L \circ b$ defined. So for some b' we have $b \xrightarrow{L} b'$ with $a' \sim_{\mathcal{L}} b'$. It follows that $q \xrightarrow{M} q'$ in $\acute{\mathbf{B}}$, where $q' = \mathcal{F}(b')$ and $(p', q') \in \mathcal{R}$, so we are done.

(1) ($\Leftarrow$) We establish in $\acute{\mathbf{A}}$ the bisimulation

$$\mathcal{S} = \{(a, b) \mid \mathcal{F}(a) \sim_{\mathcal{F}(\mathcal{L})} \mathcal{F}(b)\} .$$

Let $\mathcal{F}(a) \sim_{\mathcal{F}(\mathcal{L})} \mathcal{F}(b)$ in $\acute{\mathbf{B}}$, and let $p = \mathcal{F}(a)$, $q = \mathcal{F}(b)$ where $a \xrightarrow{L} a'$ in $\acute{\mathbf{A}}$ with $L \circ b$ defined. Then $p \xrightarrow{M} p'$ in $\acute{\mathbf{B}}$, where $M = \mathcal{F}(L)$ and $p' = \mathcal{F}(a')$. So for some q' we have $q \xrightarrow{M} q'$ with $p' \sim_{\mathcal{F}(\mathcal{L})} q'$. This transition must arise from a transition $b_1 \xrightarrow{L_1} b_1'$ in $\acute{\mathbf{A}}$, where $q = \mathcal{F}(b_1)$, $M = \mathcal{F}(L_1)$ and $q' = \mathcal{F}(b_1')$. But then $b_1 \equiv b$ and $L_1 \equiv L$, where $\equiv$ is the equivalence induced by $\mathcal{F}$; we also have $L \circ b$ defined, and $\mathcal{L}$ respects $\equiv$, so we can find b' for which $b \xrightarrow{L} b'$ and $b_1' \equiv b'$. But also $(a', b') \in \mathcal{S}$ so we are done.

(2) Assume that $\sim_{\mathcal{L}}$ is a congruence. In $\acute{\mathbf{B}}$, let $p \sim_{\mathcal{F}(\mathcal{L})} q$ and let G be a context with $G \circ p$ and $G \circ q$ defined. Then there exist a, b, C in $\acute{\mathbf{A}}$ with $p = \mathcal{F}(a)$, $q = \mathcal{F}(b)$ and $G = \mathcal{F}(C)$, and with $C \circ a$ and $C \circ b$ defined. From (1)($\Leftarrow$) we have $a \sim_{\mathcal{L}} b$, hence by assumption $C \circ a \sim_{\mathcal{L}} C \circ b$. Applying the functor $\mathcal{F}$ we have from (1)($\Rightarrow$) that $G \circ p \sim_{\mathcal{F}(\mathcal{L})} G \circ q$ in $\acute{\mathbf{B}}$, as required. ■

In a later section we shall set up link-graphical reactive systems as RSs. Then using the theorems we have just proved, or close analogues of them, we shall derive TS and deduce behavioural congruences for them.

We now turn to a question that arises strongly in applications. Our standard TS, containing only the minimal transitions, is of course much smaller than the full TS. But it turns out that in particular cases we can reduce the standard TS still further, without affecting bisimilarity. We introduce here the basic concepts to make this idea precise, since they do not depend on the domain of application of our theory.

Definition 4.15 (relative bisimulation, adequacy) Assume given a TS $\mathcal{L}$, with a sub-TS $\mathcal{M}$. A *relative bisimulation for* $\mathcal{M}$ *on* $\mathcal{L}$ is a symmetric relation $\mathcal{S}$ such that whenever $a\mathcal{S}b$, then for every transition $a \xrightarrow{L} a'$ in $\mathcal{M}$, with $L \circ b$ defined, there exists b' such that $b \xrightarrow{L} b'$ in $\mathcal{L}$ and $a'\mathcal{S}b'$. Define *relative bisimilarity for* $\mathcal{M}$ *on* $\mathcal{L}$, denoted by $\sim_{\mathcal{L}}^{\mathcal{M}}$, to be the largest relative bisimulation for $\mathcal{M}$ on $\mathcal{L}$.

We call $\mathcal{M}$ *adequate (for* $\mathcal{L}$*)* if $\sim_{\mathcal{L}}^{\mathcal{M}}$ coincides with $\sim_{\mathcal{L}}$ on the agents of $\mathcal{M}$; we write this as $\sim_{\mathcal{L}}^{\mathcal{M}} = \sim_{\mathcal{L}} \restriction \mathsf{Int}_{\mathcal{M}}$. ■

When $\mathcal{L}$ is understood we may omit 'on $\mathcal{L}$'; equally we may write $\sim^{\mathcal{M}}$ for $\sim_{\mathcal{L}}^{\mathcal{M}}$. Note that, for $a \sim_{\mathcal{L}}^{\mathcal{M}} b$, we require b only to match the transitions of a that lie in $\mathcal{M}$, and b's matching transition need not lie in $\mathcal{M}$. This means that relative bisimilarity is in general not transitive, so it is not in itself a behavioural equivalence.

Relative bisimilarity is valuable when $\mathcal{M}$ is adequate for $\mathcal{L}$, for then the proof technique of relative bisimulation can lighten the task of checking a large class of transitions. Indeed fewer labels may occur in $\mathcal{M}$-transitions than in $\mathcal{L}$-transitions; then we only need consider transitions involving this smaller set of labels.

An important example of adequacy arises from the intuition that the transitions that really matter are those where the agent 'contributes' to the underlying reaction, i.e. a supplies a 'part' of the redex r, leaving the label L to supply the rest. We can make this precise in terms of support: we are interested in transitions a whose underlying redex r is such that $|a| \cap |r| \neq \emptyset$. We call such transitions *engaged*.

Intuitively, we may conjecture that the engaged transitions are adequate, for the standard TS. We shall later prove this for a particular class of link-graphical reactive systems, and indeed in [25] the result is shown to extend to a class of *bigraphical* reactive systems (BRSs) broad enough to include the π-calculus [39] and the ambient calculus [10]. It is pleasant when the conjecture holds, for it means that the only significant labels L are such that $|L| \subsetneq |r|$ for some redex r.

We now look at a well-behaved kind of sub-TS whose transitions are determined by a set of labels.

Definition 4.16 (definite, full sub-TS) Let $\mathcal{M} \prec \mathcal{L}$. Then we call $\mathcal{M}$ *definite* for $\mathcal{L}$ if, for some subset Ls of the labels of $\mathcal{L}$,

$$\mathsf{Trans}_{\mathcal{M}} = \{a \xrightarrow{L} a' \ \in \mathsf{Trans}_{\mathcal{L}} \mid L \in Ls\} \, .$$

We call $\mathcal{M}$ *full* for $\mathcal{L}$ if Ls contains all labels $L : I \to J$ of $\mathcal{L}$ such that $I \in \mathsf{Int}_{\mathcal{M}}$. ■

To clarify these ideas, suppose that $a \xrightarrow{L} a'$ is a transition of $\mathcal{L}$. If $\mathcal{M}$ is definite for $\mathcal{L}$, then the transition's presence in $\mathcal{M}$ is determined entirely by $L : I \to J$, i.e. whether $L \in Ls$. For this, it is clearly *necessary* that $I \in \mathsf{Int}_{\mathcal{M}}$. If furthermore $\mathcal{M}$ is full for $\mathcal{L}$, then the latter condition is also *sufficient* for the transition's presence in $\mathcal{M}$.

Thus a definite sub-TS of $\mathcal{L}$ is obtained by cutting down the *transitions*, possibly leaving the interfaces unchanged; on the other hand a full sub-TS is obtained by reducing to a smaller set of *interfaces* but keeping all transitions at those interfaces. We now show that both definiteness and fullness yield congruence properties that will be useful in Section 9. For a definite sub-TS (hence also for a full sub-TS) we immediately find that a relative bisimilarity is an absolute one:

Proposition 4.17 (definite sub-TS) *If $\mathcal{M}$ is definite for $\mathcal{L}$ then $\sim_{\mathcal{M}} = \sim_{\mathcal{L}}^{\mathcal{M}}$.*

Corollary 4.18 (adequate sub-congruence) *Let $\mathcal{M}$ be definite and adequate for $\mathcal{L}$. Then*

1. *The bisimilarities on $\mathcal{M}$ and $\mathcal{L}$ coincide at $\mathsf{Int}_{\mathcal{M}}$, i.e. $\sim_{\mathcal{M}} = \sim_{\mathcal{L}} \restriction \mathsf{Int}_{\mathcal{M}}$.*
2. *If $\sim_{\mathcal{L}}$ is a congruence, then $\sim_{\mathcal{M}}$ is a congruence; that is, for any $C\colon I \to J$ where $I, J \in \mathsf{Int}_{\mathcal{M}}$, if $a \sim_{\mathcal{M}} b$ then $C \circ a \sim_{\mathcal{M}} C \circ b$.*

Finally, we discover that fullness implies not only definiteness, but also adequacy:

Proposition 4.19 (full sub-congruence) *If $\mathcal{M}$ is full for $\mathcal{L}$ then it is also adequate for $\mathcal{L}$, and hence the results of Corollary 4.18 hold.*

Proof It is enough to prove that $\sim_{\mathcal{M}} = \sim_{\mathcal{L}} \restriction \mathsf{Int}_{\mathcal{M}}$; for this, we show that $\sim_{\mathcal{M}}$ is an $\mathcal{L}$-bisimulation and that $\sim_{\mathcal{L}} \restriction \mathsf{Int}_{\mathcal{M}}$ is an $\mathcal{M}$-bisimulation. ■

---

oOo

Part II
Link graphs and their dynamics

---

5 Introduction to link graphs

Bigraphical reactive systems [36, 37, 38, 25, 26] are a graphical model of computation in which both *locality* and *connectivity* are prominent. Recognising the increasingly topographical quality of global computing, they take up the challenge to base all distributed computation on graphical structure. A typical bigraph was shown in Figure 1. Such a graph is reconfigurable, and its nodes (the ovals and circles) may represent a great variety of computational objects: a physical location, an administrative region, a data constructor, a π-calculus input guard, an ambient, a cryptographic key, a message, a replicator, and so on. We discussed several applications of bigraphs in Section 1.

Bigraphs are a development of action calculi [33]. They use ideas from many sources: the Chemical Abstract machine (Cham) of Berry and Boudol [3], the π-calculus of Milner, Parrow and Walker [39], the interaction nets of Lafont [27], the mobile ambients of Cardelli and Gordon [10], the explicit fusions of Gardner and Wischik [20] developed from the fusion calculus of Parrow and Victor [44], Nomadic Pict by Wojciechowski and Sewell [52]. They also use the theoretical basis set out in Part I.

The nesting of nodes in Figure 1 has many uses. A node may represent a location; it may limit or even prevent activity within its boundary; it may represent the scope of a link, i.e. forbid certain links to cross its boundary; it may define what should be replicated or discarded by certain reactions. When none of these are needed, then the theory is simpler. But it has been set up [26] so that the *placing* —i.e. the nesting structure of nodes— is orthogonal to the *linking* of nodes; this means that the theory of bigraphs consists of two almost independent theories, so it is easy to factor out the theory of placing.

If the nesting structure of Figure 1 is forgotten, then what remains is a *link graph*; a simple one is shown in Figure 2. These graphs are almost exactly those of standard graph theory, except that we enrich them with inner and outer interfaces to allow categorical composition.

In Sections 6 and 7 we set out respectively the structure and the dynamic theory of link graphs, in preparation for their application in Part III.

6 Link graphs

In this section we define the notion of *link graph* formally. In Section 7 we define a *link-graphical reactive system* (LRS) and study its dynamic behaviour; then we apply the results on RSs to derive labelled transitions and congruences for LRSs.

The family of link graphs in any LRS is determined by the kinds of nodes it has, and these are specified as follows:

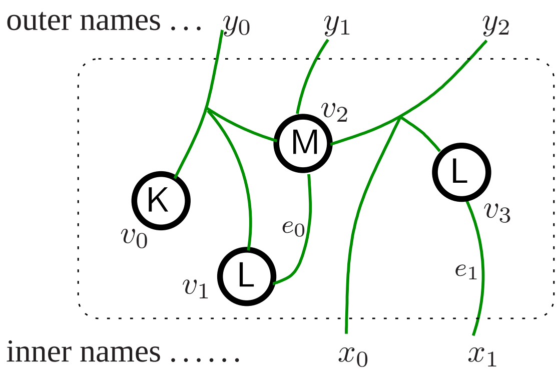


Figure 2: A link graph $G : \{x_0, x_1\} \to \{y_0, y_1, y_2\}$

Definition 6.1 (pure signature) A *pure signature* $\mathcal{K}$ provides a set whose elements are called *controls*. For each control K the signature also provides a finite ordinal $ar(K)$, its *arity*. We write $K : n$ for a control K with arity n. ■

In refinements of the theory a signature may carry further information, such as a *sort* for each arity member. These *sorted* signatures will be defined in Section 8,

In developing link graphs and LRSs we shall use two running examples. Here first are their signatures:

arithmetic nets $\mathcal{K}_{\mathsf{arith}} = \{\mathbf{0} : 1,\ \mathsf{S} : 2,\ + : 3,\ \to : 2\}$

These controls represent *zero*, *successor*, *plus* and *forwarding*. The associated LRS will evaluate arithmetic expressions. It resembles Lafont's interaction nets, but allows sharing of subevaluations.

condition-event nets $\mathcal{K}_{\mathsf{petri}} = \{\mathsf{M} : 1,\ \mathsf{U} : 1,\ \mathsf{E}_{hk} : h{+}k\}$

These controls represent a *marked condition*, an *unmarked condition*, and an *event* with h preconditions and k postconditions. The associated LRS will represent the behaviour of condition-event Petri nets. We shall derive for it a labelled transition system and an observational congruence relation, and compare them with those in the literature.

We now proceed to define link graphs over a signature $\mathcal{K}$. Informally, every node in a link graph has an associated control $K : n$, and has n ports; the graph consists essentially of an arbitrary linking of these ports, together with an inner and outer interface which provides access to some of these links. These interfaces will be the objects of an s-category whose arrows are link graphs. To express the interface we presume an infinite set $\mathcal{X}$ of *names*. Formally:

Definition 6.2 (interface) An *interface* $X, Y, \ldots$ is a finite set of names drawn from $\mathcal{X}$. We refer to the empty interface as the *origin*. ■

Definition 6.3 (link graph) A *concrete link graph*

$$A = (V, E, ctrl, link) : X \to Y$$

has interfaces X and Y, called its *inner* and *outer names*, and disjoint finite sets V of *nodes* and E of *edges*. It also has a *control map* and a *link map*, respectively $ctrl : V \to \mathcal{K}$ and $link : X \uplus P \to E \uplus Y$, where $P \stackrel{\text{def}}{=} \sum_{v \in V} ar(ctrl(v))$ is the set of *ports* of A.

We shall call the inner names X and ports P the *points* of A, and the edges E and outer names Y its *links*. ■

The term 'concrete' means that nodes and edges have identity. The support of a concrete link graph consists of its nodes and edges; in terms of the definition, $|A| = V \uplus E$. If ρ is an injective map on $|A|$, the support translation $\rho \bullet A$ is obtained by replacing each $v \in V$ by $\rho(v)$ and each $e \in E$ by $\rho(e)$ in every component of A.

Figure 2 shows a link graph $G : X \to Y$ with $X = \{x_0, x_1\}$ and $Y = \{y_0, y_1, y_2\}$, over the signature $(\mathsf{K} : 1, \mathsf{L} : 2, \mathsf{M} : 4)$. The figure shows both the nodes $V = \{v_0, \ldots, v_3\}$ and the edges $E = \{e_0, e_1\}$; in future diagrams we omit these details unless we need them. Note that the links corresponding to y_0, y_1 and y_2 have three, one and three points respectively; one of these points is the inner name x_0.

By working in an s-category of link graphs, with explicit node and edge identities, we enable the construction of RPOs. Later we shall take the quotient by support equivalence to obtain *abstract* link graphs, where RPOs do not exist in general. As is usual in graph theory, we shall omit the adjectives 'concrete' and 'abstract' when they are unimportant or implied by the context.

Note that the names in an interface are identified alphabetically, not positionally. Alphabetical names are convenient for link graphs just as they are convenient in the λ-calculus, and they also lead naturally to forms of parallel product that are familiar from process calculi, as we shall see below.

Let us now look at the elementary link graphs. The first kind, the elementary *wirings*, are shown in Figure 3; they have no nodes. The *linker* $y/\vec{x} : \{\vec{x}\} \to \{y\}$ has no edges, and its link map sends the names $\vec{x}$ (all distinct) to y. The case when $\vec{x}$ is empty, written $y : \emptyset \to y$, is just a link graph with a single idle name (see Definition 6.8). The *closure* $/x : \{x\} \to \emptyset$ has just one edge, to which it maps the inner name x. When we draw a link graph we put all its nodes in a dotted rectangle, with the outer names above and the inner names below, and links (usually curved) joining them.

The second kind of elementary link graph is the *atom* $K\vec{x} : \emptyset \to \{\vec{x}\}$, where $K : n$ is a control and $\vec{x}$ a vector of n distinct names. It consists of a single node with a link x_i for each port $i \in n$. Figures 4 and 5 show the node graphs $\mathbf{0}x$ etc for arithmetic nets, and $\mathsf{E}_{21}xyz$ etc for condition-event nets. We draw nodes with a variety of shapes; the shape has no formal purpose except to determine the ordering of ports.

All link graphs can be expressed in terms of atoms and elementary wirings, with the help of composition and tensor product, which we now define.

Definition 6.4 (s-category of link graphs) The s-category $\acute{\text{L}}\text{IG}(\mathcal{K})$ over a signature $\mathcal{K}$ has name sets as objects and link graphs as arrows. The composition $A_1 \circ A_0 : X_0 \to X_2$ of two link graphs $A_i = (V_i, E_i, ctrl_i, link_i) : X_i \to X_{i+1}$ $(i = 0, 1)$ is defined when their supports are disjoint; then their composite is

$$A_1 \circ A_0 \stackrel{\text{def}}{=} (V_0 \uplus V_1, E_0 \uplus E_1, ctrl, link) : X_0 \to X_2$$

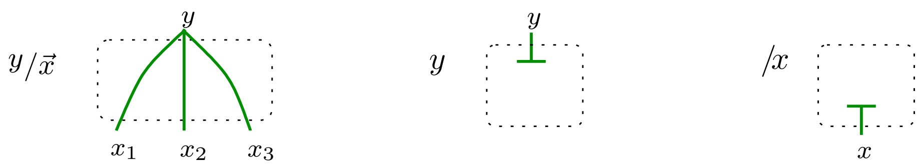


Figure 3: elementary wirings

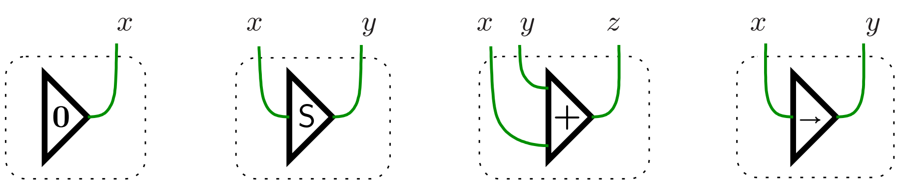


Figure 4: atoms for arithmetic nets

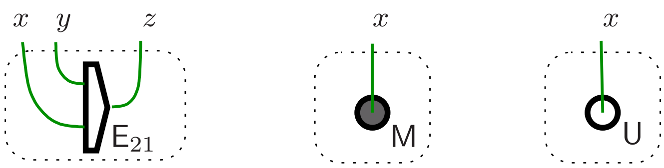


Figure 5: atoms for condition-event nets

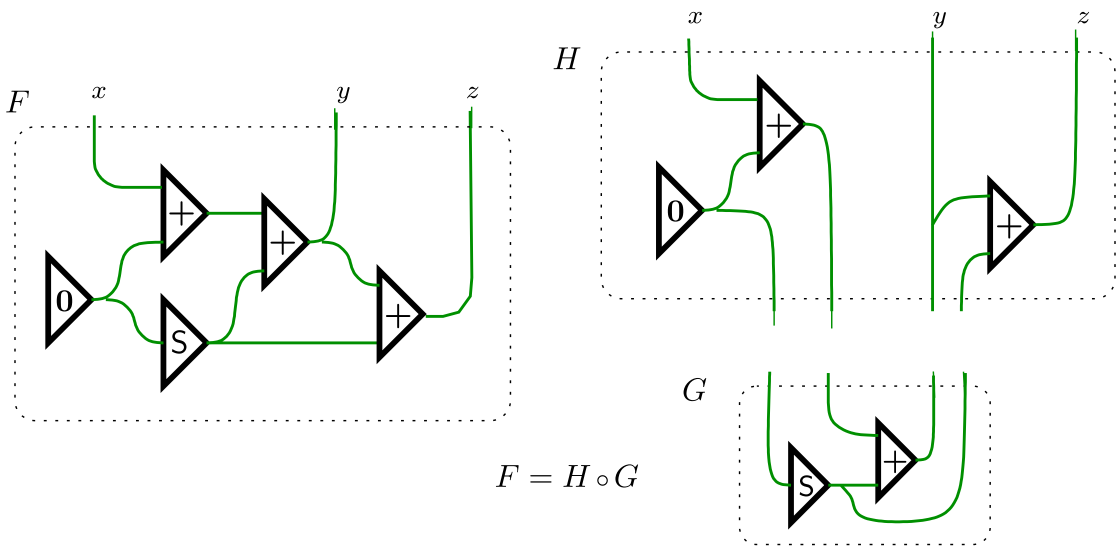


Figure 6: A ground link graph and its decomposition

where $ctrl = ctrl_0 \uplus ctrl_1$ and $link = (\mathsf{Id}_{E_0} \uplus link_1) \circ (link_0 \uplus \mathsf{Id}_{P_1})$.

The identity link graph at X is $\mathsf{id}_X \stackrel{\text{def}}{=} (\emptyset, \emptyset, \emptyset_{\mathcal{K}}, \mathsf{Id}_X) : X \to X$. A *ground* link graph $G : \emptyset \to X$ is one whose inner interface is the origin. ■

To clarify composition, here is another way to define the link map of $A_1 \circ A_0$, considering all possible arguments $p \in X_0 \uplus P_0 \uplus P_1$:

$$link(p) = \begin{cases} link_0(p) & \text{if } p \in X_0 \uplus P_0 \text{ and } link_0(p) \in E_0 \\ link_1(x) & \text{if } p \in X_0 \uplus P_0 \text{ and } link_0(p) = x \in X_1 \\ link_1(p) & \text{if } p \in P_1 \ . \end{cases}$$

We often denote the link map of A simply by A.

Note that the link map treats inner and outer names differently. Two inner names may be linked —indeed, this is the purpose of the elementary linker— but each outer name constitutes (the target of) a distinct link. The effect is that we do not allow ‘aliases’, i.e. synonymous outer names. A previous version of bigraphs [37] allowed these; the effect was a much harder proof of the existence of RPOs, and then only under certain conditions. The present version has wide application.

Figure 6 shows a ground link graph F in ´LIG($\mathcal{K}_{\mathsf{arith}}$). In such diagrams we often omit the identities of nodes and edges. Also note that a link with several points is represented by forking lines. The way the lines fork has no formal significance, but may be suggestive of the intended application; for example, here it suggests that the ‘output’ of the successor node is ‘input’ by two plus nodes.

The figure also shows how F may be composed from a smaller ground link graph G and a context H. Later we shall see that G is the *redex* of a reaction rule for arithmetic; it is in fact part of the primitive-recursive definition of summation in terms of zero and successor. The sharing of the successor node is achieved by composition because its ‘output’ port belongs to a link of G that is open (see Definition 6.8).

Definition 6.5 (tensor product) The *tensor product* $\otimes$ in $\acute{\text{L}}\text{IG}(\mathcal{K})$ is defined as follows: On objects, $X \otimes Y$ is simply the union $X \uplus Y$ of sets required to be disjoint. For two link graphs $A_i : X_i \to Y_i$ $(i = 0, 1)$ we take $A_0 \otimes A_1 : X_0 \otimes X_1 \to Y_0 \otimes Y_1$ to be defined when they have disjoint support and the interface products are defined; its link map is the union of those of A_0 and A_1. ■

The identity $\mathsf{id}_\emptyset$ is clearly a unit for tensor product, which also obeys the axioms for a monoidal s-category. We therefore obtain the following:

Proposition 6.6 (link graphs are monoidal) *The s-category $\acute{\text{L}}\text{IG}(\mathcal{K})$ is monoidal, with origin $\epsilon = \emptyset$.*

We shall call a tensor product of linkers a *substitution*, and use σ, τ to range over substitutions. A tensor product of linkers and closures is call a *wiring*; we use ω to range over wirings.

We can conveniently blur the distinction between substitutions as functions and as link graphs; their composition and tensor product means the same in either case. Substitutions can be used to derive an important variant of tensor product of link graphs that merges outer names, i.e. does not require them to be disjoint:

Definition 6.7 (parallel product) The *parallel product* $\mid$ in $\acute{\text{L}}\text{IG}(\mathcal{K})$ is defined as follows: On objects, $X \mid Y \stackrel{\text{def}}{=} X \cup Y$. On link graphs $A_i : X_i \to Y_i$ $(i = 0, 1)$ with disjoint support we define $A_0 \mid A_1 : X_0 \otimes X_1 \to Y_0 \mid Y_1$ whenever X_0 and X_1 are disjoint, by taking the union of link maps. ■

In fact let $\sigma_i : Y_i \to Z_i$ $(i = 0, 1)$ be bijective substitutions with disjoint codomains, and let $\tau : Z_0 \otimes Z_1 \to Y_0 \cup Y_1$ be the union of their inverses; then

$$A_0 \mid A_1 = \tau \circ ((\sigma_0 \circ A_0) \otimes (\sigma_1 \circ A_1)) \ .$$

Parallel product has fewer algebraic properties than the tensor (categorically, it is not a bifunctor), but will be important in modelling process calculi such as the π-calculus and the ambient calculus.

We now define some basic properties:

Definition 6.8 (idle, open, closed, peer, lean) A link with no preimage under the link map is *idle*. An outer name is an *open* link, an edge is a *closed* link. A point (i.e. an inner name or port) is *open* if its link is open, otherwise *closed*. Two distinct points are *peers* if they are in the same link. A link graph with no idle edges is *lean*. ■

An idle *name* is sometimes needed; for example we may want to consider two link graphs as members of the same homset, even if one of them uses a name x and the other does not. On the other hand an idle *edge* serves no useful purpose, but may be created by composition. Sometimes we shall need to ensure that the property of leanness (no idle edges) is preserved by certain constructions.

Isomorphisms, epimorphisms and monomorphisms are easy to characterise, and will play an important part:

Proposition 6.9 (isos, epis and monos in link graphs) *A link graph is an iso iff it is a bijective substitution; it is epi iff no outer name is idle; it is mono iff no two inner names are peers.*

For what follows we need some more notation:

Notation When considering a pair $\vec{A} : W \to \vec{X}$ of link graphs with common domain W, we shall adopt a convention for naming their nodes, ports and edges. We denote the node set of A_i $(i = 0, 1)$ by V_i, and denote $V_0 \cap V_1$ by V_2. We shall use $v_i, v_i', \ldots$ to range over V_i $(i = 0, 1, 2)$. Similarly we use $p_i \in P_i$ and $e_i \in E_i$ for ports and edges $(i = 0, 1, 2)$. However, we shall sometimes use p_i also for points, i.e. $p_i \in W \uplus P_i$; the context will resolve any ambiguity. ■

We now turn to constructing RPOs for concrete link graphs. An informal intuition will help in understanding the construction. Suppose $\vec{D}$ is a bound for $\vec{A}$, and we wish to construct the RPO $(\vec{B}, B)$. To form $\vec{B}$, we first truncate $\vec{D}$ by removing its outer names, and all nodes and edges not present in $\vec{A}$. (Of course, for this the identity of nodes and edges is essential.) Then for the outer names of $\vec{B}$, we create a name for each link severed by the truncation, equating these new names only when required to ensure that $B_0 \circ A_0 = B_1 \circ A_1$. Formally:

Construction 6.10 (RPOs in link graphs) An RPO $(\vec{B} \colon \vec{X} \to \hat{X}, B \colon \hat{X} \to Z)$, for a pair $\vec{A} \colon W \to \vec{X}$ of link graphs relative to a bound $\vec{D} \colon \vec{X} \to Z$, will be built in three stages. We use the notational conventions introduced above.

nodes and edges: If V_i are the nodes of A_i $(i = 0, 1)$ then the nodes of D_i are $(V_{\bar{i}} - V_2) \uplus V_3$ for some V_3. Define the nodes of B_i and B to be $V_{\bar{i}} - V_2$ $(i = 0, 1)$ and V_3 respectively. Edges are treated exactly analogously, and ports inherit the analogous treatment from nodes.

interface: Construct the outer names $\hat{X}$ of $\vec{B}$ as follows. First, define the names in each X_i that must be mapped into $\hat{X}$:

$$X_i' \stackrel{\text{def}}{=} \{x \in X_i \mid D_i(x) \in E_3 \uplus Z\} .$$

Next, on the disjoint sum $X_0' + X_1'$, define $\cong$ to be the smallest equivalence for which $(0, x_0) \cong (1, x_1)$ whenever $A_0(p) = x_0$ and $A_1(p) = x_1$ for some point $p \in W \uplus P_2$. Then define $\hat{X}$ up to isomorphism as follows:

$$\hat{X} \stackrel{\text{def}}{=} (X_0' + X_1')/\cong .$$

For each $x \in X_i'$ we denote by $\widehat{i, x}$ the name in $\hat{X}$ corresponding to the $\cong$-equivalence class of (i, x).

links: Define B_0 to simulate D_0 as far as possible (B_1 is similar):

$$\text{For } x \in X_0 : \quad B_0(x) \stackrel{\text{def}}{=} \begin{cases} \widehat{0, x} & \text{if } x \in X_0' \\ D_0(x) & \text{if } x \notin X_0' \end{cases}$$

$$\text{For } p \in P_1 - P_2 : \quad B_0(p) \stackrel{\text{def}}{=} \begin{cases} \widehat{1, x} & \text{if } A_1(p) = x \in X_1 \\ D_0(p) & \text{if } A_1(p) \notin X_1 . \end{cases}$$

Finally define B, to simulate both D_0 and D_1:

$$\begin{array}{lll} \text{For } \hat{x} \in \hat{X}: & B(\hat{x}) \stackrel{\text{def}}{=} D_i(x) & \text{where } x \in X_i \text{ and } \widehat{i,x} = \hat{x} \\ \text{For } p \in P_3: & B(p) \stackrel{\text{def}}{=} D_i(p)\ . & \end{array}$$
■

To prove this definition sound we have to show that the right-hand sides in the clauses defining link maps B_i and B are well-defined links in B_i and B respectively:

Lemma 6.11 *The definition in Construction 6.10 is sound.*

Proof The second clause defining $B_0(x)$ is sound, since if $x \notin X'_0$ then by definition $D_0(x) \in E_1 - E_2$, which is indeed the port set of B_0. Similar reasoning applies to the second clause defining $B_0(p)$.

The first clause defining $B_0(p)$ is sound, since if $A_1(p) = x$ with $p \in P_1 - P_2$ then we have $x \in X'_1$; for if not, then $D_1(x) \in E_0 - E_2$, which is impossible since $D_1 \circ A_1 = D_0 \circ A_0$.

Finally, the clauses defining B are sound because the right-hand sides are independent of the choice of i and of x; this is seen by appeal to the definition of $\cong$ and the equation $D_1 \circ A_1 = D_0 \circ A_0$. ■

The full justification of our construction lies in the following lemma and theorem, both of which are proved in the Appendix:

Lemma 6.12 $(\vec{B}, B)$ *is a candidate RPO for* $\vec{A}$ *relative to* $\vec{D}$*.*

Theorem 6.13 (RPOs in link graphs) *´*LIG*($\mathcal{K}$) has RPOs; that is, whenever a pair* $\vec{A}$ *of link graphs has a bound* $\vec{D}$*, there exists an RPO* $(\vec{B}, B)$ *for* $\vec{B}$ *to* $\vec{D}$*. Moreover Construction 6.10 yields such an RPO.*

It is clear that the identity of nodes and edges plays an important role in our RPO construction. Indeed, the category LIG of abstract link graphs does not possess RPOs in general. A counter-example appears as Example 10 (Figure 12) in [26]; it is presented in terms of bigraphs, but involves only their link graph components.

Now, to prepare for the derivation of labelled transition systems, we proceed to characterise all the IPOs for a given pair $\vec{A}\colon W \to \vec{X}$ of link graphs. Recall that $\vec{B}$ is an IPO for $\vec{A}$ iff $(\vec{B}, B)$ is an RPO for some B.

How does a link graph RPO $(\vec{B}, B)$ vary, when we keep $\vec{A}$ fixed but vary the given bound $\vec{D}$? The answer is that if $\vec{A}$ are both epi, then $\vec{B}$ remains fixed and only B varies, so that in this case $\vec{B}$ is a pushout. But we need to treat the general case. The first step is to establish consistency conditions.

Definition 6.14 (consistency conditions for link graphs) We define three *consistency* conditions on a pair $\vec{A} : W \to \vec{X}$ of place graphs. We use p to range over arbitrary points and $p_2, p'_2, \ldots$ to range over $W \uplus P_2$, the shared points.

CL0 If $v \in V_0 \cap V_1$ then $ctrl_0(v) = ctrl_1(v)$.
CL1 If $A_i(p) \in E_2$ then $p \in W \uplus P_2$ and $A_{\bar{\imath}}(p) = A_i(p)$.
CL2 If $A_i(p_2) \in E_i - E_2$ then $A_{\bar{\imath}}(p_2) \in X_{\bar{\imath}}$, and if also $A_{\bar{\imath}}(p) = A_{\bar{\imath}}(p_2)$ then $p \in W \uplus P_2$ and $A_i(p) = A_i(p_2)$. ■

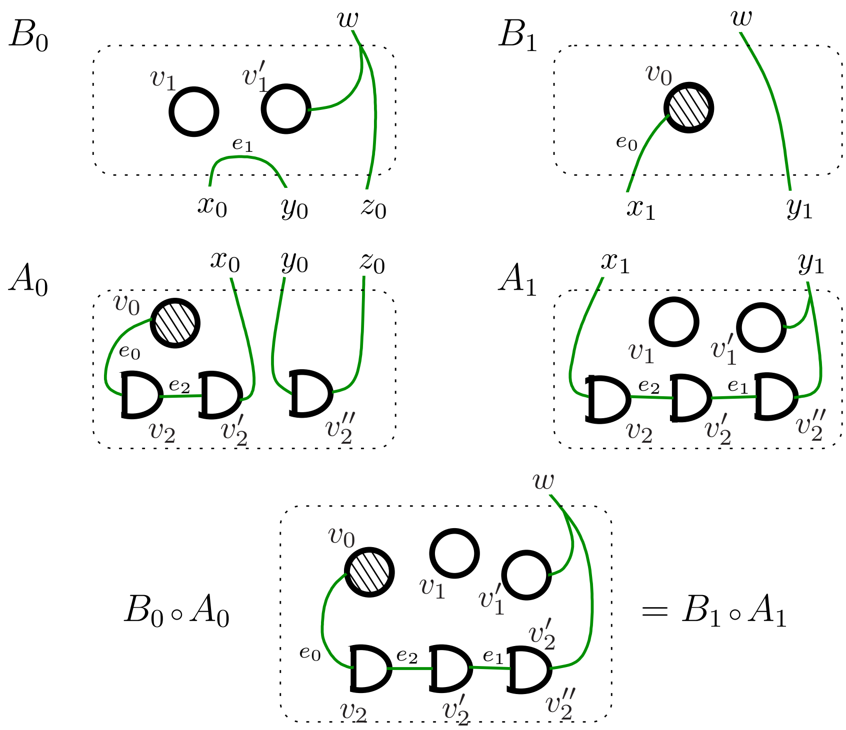


Figure 7: A consistent pair $\vec{A}$ of link graphs, with bound $\vec{B}$

Let us express CL1 and CL2 in words. If $i = 0$, CL1 says that if the link of any point p in A_0 is closed and shared with A_1, then p is also shared and has the same link in A_1. CL2 says, on the other hand, that if the link of a shared point p_2 in A_0 is closed and *unshared*, then its link in A_1 must be open, and further that any peer of p_2 in A_1 must also be its peer in A_0.

We shall find that the consistency conditions are necessary and sufficient for at least one IPO to exist. Necessity is straightforward:

Proposition 6.15 (consistency in link graphs) *If the pair $\vec{A}$ has a bound, then the consistency conditions hold.*

Before going further, it may be helpful to see a simple example.

Example 1 (consistent link graphs) Consider the pair $\vec{A}: \emptyset \to \vec{X}$ of link graphs in Figure 7, where $X_0 = \{x_0, y_0, z_0\}$ and $X_1 = \{x_1, y_1\}$. Nodes and edges with subscript 2 are shared; circular nodes are unshared. (Controls are not shown). The pair is consistent, with bound $\vec{B}$ as shown. It is worth checking the consistency conditions. ■

Now, assuming the consistency conditions of Definition 6.14, we shall construct a non-empty family of IPOs for arbitrary $\vec{A}$. Informally, the construction works as follows: We choose an arbitrary subset of the idle outer names of $\vec{A}$ which will be given special treatment. If there are no idle outer names then there will be a unique

IPO which is also a pushout. We have a degree of freedom for each such outer name x in A_i ($i = 0, 1$). In an IPO $\vec{C}$ we may choose $C_i(x)$ either to be a new open link, or to be any closed link in C_i. We call the latter case an *elision* of the idle name x; in the following construction the set L_i represents the set of idle names to be elided.

Construction 6.16 (IPOs in link graphs) Assume the consistency conditions for the pair of link graphs $\vec{A} : W \to \vec{X}$. We define a family of IPOs $\vec{C}: \vec{X} \to Y$ for $\vec{A}$ as follows.

nodes and edges: Take the nodes and edges of C_i to be $V_{\overline{\imath}} - V_2$ and $E_{\overline{\imath}} - E_2$.

interface: For $i = 0, 1$ choose any subset L_i of the names X_i such that all members of L_i are idle. Set $K_i = X_i - L_i$. Define $K'_i \subseteq K_i$, the names to be mapped to the codomain Y, by

$$K'_i \stackrel{\text{def}}{=} \{x_i \in K_i \mid \forall p \in P_2.\ A_i(p) = x_i \Rightarrow A_{\overline{\imath}}(p) \in X_{\overline{\imath}}\} \ .$$

Next, on the disjoint sum $K'_0 + K'_1$, define $\simeq$ to be the smallest equivalence such that $(0, x_0) \simeq (1, x_1)$ whenever $A_0(p) = x_0$ and $A_1(p) = x_1$ for some $p \in W \uplus P_2$. Then define the codomain up to isomorphism:

$$Y \stackrel{\text{def}}{=} (K'_0 + K'_1)/\simeq \ .$$

For each $x \in K'_i$ we denote the $\simeq$-equivalence class of (i, x) by $\widehat{i, x}$.

links: Choose two arbitrary maps $\eta_i : L_i \to E_{\overline{\imath}} - E_2$ ($i = 0, 1$), called *elision* maps, and define the link maps $C_i : X_i \to Y$ as follows (we give C_0; C_1 is similar):

For $x \in X_0$:

$$C_0(x) \stackrel{\text{def}}{=} \begin{cases} \widehat{0, x} & \text{if } x \in K'_0 \\ A_1(p) & \text{if } x \in K_0 - K'_0, \text{ for } p \in W \uplus P_2 \text{ with } A_0(p) = x \\ \eta_0(x) & \text{if } x \in L_0 \end{cases}$$

For $p \in P_1 - P_2$:

$$C_0(p) \stackrel{\text{def}}{=} \begin{cases} \widehat{1, x} & \text{if } A_1(p) = x \in X_1 \\ A_1(p) & \text{if } A_1(p) \notin X_1 \ . \end{cases}$$

■

Thus there is a distinct IPO for each choice of sets L_i and elision maps η_i. However the IPO will be unique if $L_i = \emptyset$ is forced. This can happen for one of two reasons: either, as previously mentioned, A_i has no idle names (i.e. it is epi); or $E_{\overline{\imath}} - E_2$ is empty (i.e. all edges of $A_{\overline{\imath}}$ are shared), so no elision can exist.

A particular case of $\vec{A}$ with no elisive IPOs is when one member, A_1 say, has no idle names and no edges (closed links). This is because the former prevents elisions from A_1, while the latter entails that C_0 has no edges and so prevents elisions from A_0. Now our principle application of IPOs is to derive transitions for A_0 when A_1 is the redex of a reaction rule, and in many reactive systems the redexes do indeed have this desirable property. We shall see later that this yields rather simple transition systems.

Lemma 6.17 *The definition of $\vec{C}$ is sound and yields a bound.*

Proof In the second clause for $C_0(x)$ we must ensure that $p \in W \uplus P_2$ exists such that $A_0(p) = x$, and that each such p yields the same value $A_1(p)$ in $P_1 - P_2$; in the first clause for $C_0(p)$ we must ensure that $x \in K_1'$. The consistency conditions do indeed ensure this, and they also ensure that $C_0 \circ A_0 = C_1 \circ A_1$. ■

We can now prove the essential theorem that underlies the derivation of labelled transition systems. It states that our construction creates all and only IPOs for $\vec{A}$.

Theorem 6.18 (characterising IPOs for link graphs) *A pair $\vec{C} : \vec{X} \to Y$ is an IPO for $\vec{A} : W \to \vec{X}$ iff it is generated (up to isomorphism) by Construction 6.16.*

Proof (outline)

($\Rightarrow$) Recall that a bound $\vec{B}$ for $\vec{A}$ is an IPO iff it is the legs of an RPO for some bound $\vec{D}$. So assume such a $\vec{B} : \vec{X} \to \widehat{X}$ built by Construction 6.10, and recall the subsets $X_i' \subseteq X_i$ and the equivalence $\cong$ over $: X_0' + X_1'$ defined there. Now apply Construction 6.16 to create a pair $\vec{C} : \vec{X} \to Y$, by choosing the sets $\vec{L}$ and elision maps $\vec{\eta}$ as follows:

$$\begin{aligned} L_i &\stackrel{\text{def}}{=} \{x \in X_i \mid x \text{ idle in } A_i, D_i(x) \in P_{\bar{i}}\} \\ \eta_i : L_i \to P_{\bar{i}} &\stackrel{\text{def}}{=} D_i \restriction L_i \ . \end{aligned}$$

Then indeed $\vec{C}$ coincides with $\vec{B}$. To prove this, first show that K_0', K_1' and $\simeq$ in the IPO construction coincide with X_0', X_1' and $\cong$ in the RPO construction; hence the codomain Y of $\vec{C}$ coincides with the codomain $\widehat{X}$ of $\vec{B}$. Then show that the link maps C_i coincide with B_i. Thus every IPO is a bound built by Construction 6.16.

($\Leftarrow$) To prove the converse, consider any bound $\vec{C} : \vec{X} \to Y$ built by Construction 6.16, for some sets $\vec{L}$ and elision maps $\vec{\eta}$. Now apply Construction 6.10 to yield an RPO $(\vec{B}, B)$ for $\vec{A}$ to $\vec{C}$.

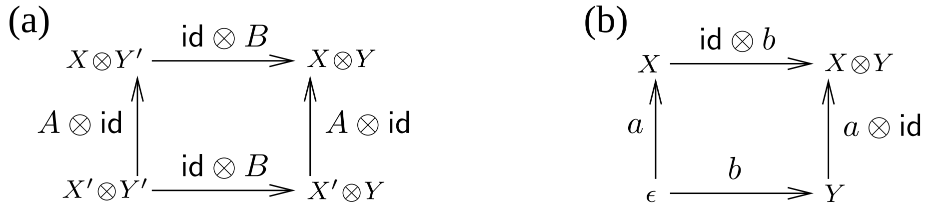
Then indeed $\vec{B}$ coincides with $\vec{C}$ up to isomorphism. To prove this, first show that X_0', X_1' and $\cong$ in the RPO construction coincide with K_0', K_1' and $\simeq$ in the IPO construction; hence the codomain $\widehat{X}$ of $\vec{B}$ coincides with the codomain Y of $\vec{C}$. Then show that the link maps B_i coincide with C_i. Thus every bound built by Construction 6.16 is an IPO. ■

The reader may like to check the IPO construction by confirming that the bound illustrated in Figure 7 is in fact an IPO.

We continue with properties of IPOs that we shall need. First, tensor product preserves IPOs with disjoint support:

Proposition 6.19 (tensor IPO) *In ´LIG($\mathcal{K}$), let $\vec{C}$ be an IPO for $\vec{A}$ and $\vec{D}$ be an IPO for $\vec{B}$, where the supports of the two IPOs are disjoint. Then, provided the tensor products exist, $\vec{C} \otimes \vec{D}$ is an IPO for $\vec{A} \otimes \vec{B}$.*

An important corollary is:



Corollary 6.20 (tensor IPOs with identities) *Let $A : X' \to X$ and $B : Y' \to Y$ have disjoint support, and let $X' \cup X$ be disjoint from $Y' \cup Y$. Then the pair $(A \otimes \mathsf{id}_{Y'}, \mathsf{id}_{X'} \otimes B)$ has an IPO $(\mathsf{id}_X \otimes B, A \otimes \mathsf{id}_Y)$. See diagram* (a).

In particular if $X' = Y' = \epsilon$ then $A = a$ and $B = b$ are ground link graphs, and the IPO is as in diagram (b).

Our next proposition shows exactly when an IPO becomes a pushout.

Proposition 6.21 (unique IPOs are pushouts) *In link graphs, an IPO is unique up to isomorphism iff it is a pushout.*

Proof For the forward implication, we claim first that the RPO $(\vec{B}, B)$ built by Construction 6.10 is rigid, in the sense Definition 3.11, i.e. the last component B is determined by the first two $\vec{B}$. This follows from the fact that the equations defining B in that construction are necessary to ensure that $B \circ B_i = D_i$ ($i = 0, 1$), It follows that in link graphs a pair $\vec{A}$ has a rigid RPO relative to any bound. Proposition 3.12 then yields the required result.

For the reverse implication, it is easy to check that a pushout for $\vec{A}$ provides an RPO relative to any bound, and is therefore an IPO by Proposition 3.10(2). ■

Recall that a link graph is *lean* if it has no idle edges. In Section 7 we shall need to transform IPOs by the addition or subtraction of idle edges. Let us write A^E for the result of adding a set E of fresh idle edges to A. The following is easy to prove from the IPO construction for link graphs:

Proposition 6.22 (IPOs, idle edges and leanness) *For any two pairs $\vec{A}$ and $\vec{B}$:*

1. *If $\vec{B}$ is an IPO for $\vec{A}$, and A_1 is lean, then B_0 is lean.*
2. *For any fresh set E of edges, $\vec{B}$ is an IPO for $\vec{A}$ iff (B_0, B_1^E) is an IPO for (A_0^E, A_1).*

We now turn to abstract link graphs. To get them from concrete bigraphs, we wish to factor out the identity of nodes and edges; we also wish to forget any idle edges. So we define an equivalence $\simeq$ that is a little coarser than support equivalence ($\widehat{=}$):

Definition 6.23 (abstract link graphs and their category) Two concrete link graphs A and B are *lean-support equivalent*, written $A \simeq B$, if after discarding any idle edges they are support equivalent. The category $\text{LIG}(\mathcal{K})$ of *abstract link graphs* has the same objects as $\acute{}\text{LIG}(\mathcal{K})$, and its arrows are lean-support equivalence classes of concrete link graphs. Lean-support equivalence is clearly a static congruence (Definition 3.5). The associated quotient functor, as defined in Definition 3.6, is

$$[\![\cdot]\!] : \acute{}\text{LIG}(\mathcal{K}) \to \text{LIG}(\mathcal{K}) \ . \qquad ■$$

The reason for studying concrete, rather than abstract, link graphs is that they possess RPOs. This will allow us Section 7 to derive a behavioural congruence for ´LIG, and then to show how to transfer it, under certain assumptions, to LIG.

To see why we cannot work directly in LIG, we point out that it lacks structure that is present in ´LIG. For example, the functor $[\![\cdot]\!]$ does not preserve epis. More seriously, LIG lacks RPOs in general; this arises because it lacks any notion of the *occurrence* of a node or edge. A counter-example appears as Example 10 (Figure 12) in [26] (It is presented in terms of bigraphs, but involves only their link graph components.)

7 Reactions and transitions for link graphs

We are now ready to specialise the definitions and theory for reactive systems (RSs) in Section 4, to obtain link-graphical reactive systems (LRSs), which form the objects of a category whose arrows are RS functors.

Definition 7.1 (link-graphical reactive system) A *(concrete) link-graphical reactive system (LRS)* over a signature $\mathcal{K}$ consists of a monoidal reactive system over ´LIG($\mathcal{K}$), with a rule-set $\acute{\mathcal{R}}$ in which every redex in lean. We denote it by

$$\acute{\text{L}}\text{IG}(\mathcal{K}, \acute{\mathcal{R}}) \, . \qquad \blacksquare$$

As an example, Figure 8 shows a likely set of rules for the evaluation of arithmetic nets, whose atoms appeared in Figure 4. The left-hand two rules represent the primitive recursive definition of $+$, while the right-hand two rules deal with the forwarder, $\rightarrow$. In each case we consider the names $\vec{x}$ and $\vec{y}$ to represent inputs and outputs respectively. (In Section 8 we shall capture this distinction by imposing a sort-discipline on link graphs.) The upper left-hand rule introduces a forwarder node. The upper right-hand rule creates a bypass around a forwarder; it is really a family of rules, since '**?**' represents any of the three controls $\{\mathsf{S}, +, \rightarrow\}$ and the dotted link represents any extra inputs to the node with that control. The lower right-hand rule eliminates a forwarder that has finished its work.

Figure 6 shows how a redex, denoted by G, may occur within a ground arithmetic net F; the occurrence is represented by the context H. The reader may like to draw compositions that represent two other redex occurrences within F. This example is close to Hasegawa's sharing graphs [21], which enrich Lafont's interaction nets [27] by permitting shared subevaluations .

We now proceed to consider the derivation of labelled transitions for LRSs. This derivation instantiates the derivation for arbitrary RSs of transitions $a \xrightarrow{L} a'$ based upon IPOs, leading to the standard transition system ST. LRSs thereby inherit the definition of bisimilarity, so we have the following corollary of Theorem 4.12:

Corollary 7.2 (congruence of bisimilarity) *In any concrete LRS equipped with the standard transition system* ST*, bisimilarity of agents is a congruence.*

A natural question arises about identity transitions $a \xrightarrow{\mathsf{id}} a'$; do they differ from reactions $a \longrightarrow a'$? The two clearly coincide in the full transition system FT; but even

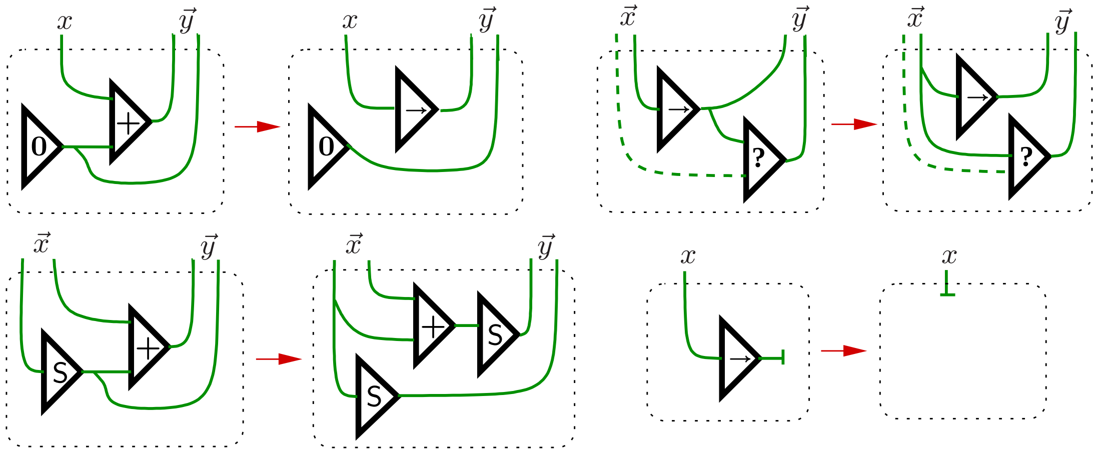


Figure 8: Reaction rules for arithmetic

in ST we would expect them to coincide, since both appear to represent the occurrence of a reaction without external assistance. In fact we have the following:

Proposition 7.3 (identity transitions are reactions) *In a concrete LRS equipped with standard transitions, if no redex has idle names then* $a \xrightarrow{\mathsf{id}} a'$ *iff* $a \longrightarrow a'$*.*

Proof The forward implication is immediate. For the reverse, if $a \longrightarrow a'$ then $a = D \circ r$ and $a' \simeq D \circ r'$ for some rule (r, r'). But r has no idle names, so by Proposition 6.9 it is epi. But then it can be shown (by purely categorical means) that the pair $(D \circ r, r)$ has (id, D) as a pushout, and hence as an IPO; it follows that $a \xrightarrow{\mathsf{id}} a'$. ■

This result is valuable, since we see little value in a redex with idle names. The reader may agree that it would be strange to have a rule where x is idle in the redex but not in the reactum, and if it is idle in both it makes good sense to delete it.

We shall later examine the transition system ST carefully, with the help of a detailed example of condition-event Petri nets. For now, we consider how ST and its induced bisimilarity congruence are transferred to the abstract LRS $\mathrm{LIG}(\mathcal{K},\mathcal{R})$, where $\mathrm{LIG}(\mathcal{K})$ is defined by the quotient functor $[\![\cdot]\!]$ of Definition 6.23, and $\mathcal{R}$ is obtained from $\acute{\mathcal{R}}$ also by $[\![\cdot]\!]$.

Now recall that this functor, the quotient by lean-support equivalence ($\risingdotseq$), is a little coarser than the quotient by support equivalence ($\simeq$), because it discards idle edges. To transfer the congruence result we must prove that $\risingdotseq$ respects ST. For this purpose, we have required all redexes in $\acute{\mathcal{R}}$ to be lean (which is no limitation in practice). We then deduce the crucial property of lean-support equivalence:

Proposition 7.4 (transitions respect equivalence) *In a concrete LRS equipped with standard transitons:*

1. *Every transition label L is lean.*

2. *Transitions respect lean-support equivalence ($\Bumpeq$) in the sense of Definition 4.8. That is, for every transition $a \xrightarrow{L} a'$, if $a \Bumpeq b$ and $L \Bumpeq M$ where M is another label with $M \circ b$ defined, then there exists a transition $b \xrightarrow{M} b'$ for some b' such that $a' \Bumpeq b'$.*

Proof For the first part, use Proposition 6.22(1). For the second part, use Proposition 6.22(2); the assumption that each redex is lean ensures that it cannot share an idle edge with the agent a. ■

We are now ready to transfer transition systems, bisimilarities and congruence results from concrete to abstract LRSs. The following is immediate by invoking Theorem 4.14 and Proposition 7.4, followed by Corollary 7.2:

Corollary 7.5 (behavioural congruence in abstract LRSs) *Let $\acute{\mathbf{A}}$ be a concrete LRS equipped with a TS $\mathcal{L}$ that respects lean-support equivalence. Denote by $\mathbf{A}$ the lean-support quotient of $\acute{\mathbf{A}}$, and denote by $\sim_{\mathcal{L}}$ the bisimilarity induced by $\mathcal{L}$ in both $\acute{\mathbf{A}}$ and $\mathbf{A}$. Then*

1. *$a \sim_{\mathcal{L}} b$ in $\acute{\mathbf{A}}$ iff $[\![a]\!] \sim_{\mathcal{L}} [\![b]\!]$ in $\mathbf{A}$.*
2. *If $\sim_{\mathcal{L}}$ is a congruence in $\acute{\mathbf{A}}$ then it is a congruence in $\mathbf{A}$.*
3. *The bisimilarity induced by* ST *in $\mathbf{A}$ is a congruence.*

This concludes the elementary theory of LRSs. We shall now specialise it by defining the *simple* LRSs, whose redexes have certain structural properties. As predicted in Section 4, working in ´LIG we then show that engaged transitions are adequate for the standard transition system ST. This yields a more tractable TS, which we can again transfer to abstract LRSs over LIG, yielding a bisimilarity that is a congruence.

Recall from Section 6 that a link is *open* if it an outer name, otherwise *closed*, and that these properties are inherited by the points of the link.

Definition 7.6 (simple) A link graph is *simple* if it has no idle names and all its links are open. An LRS is *simple* if all its redexes are simple. ■

We have already argued that the first condition is easy to accept, so the main constraint is openness. It remains to be seen how far we can relax it while retaining our results; meanwhile, many simple LRSs appear to arise naturally.

Simpleness has important consequences:

Proposition 7.7 (simpleness properties)

1. *Every simple link graph is lean.*
2. *If $\vec{B}$ is an IPO for $\vec{A}$ and A_1 is simple, then B_0 is simple and the IPO is a pushout.*
3. *In a simple LRS equipped with* ST*, every label is simple and the IPO underlying every transition is a pushout.*

Proof (1) It is enough to note that a simple link graph has no edges.

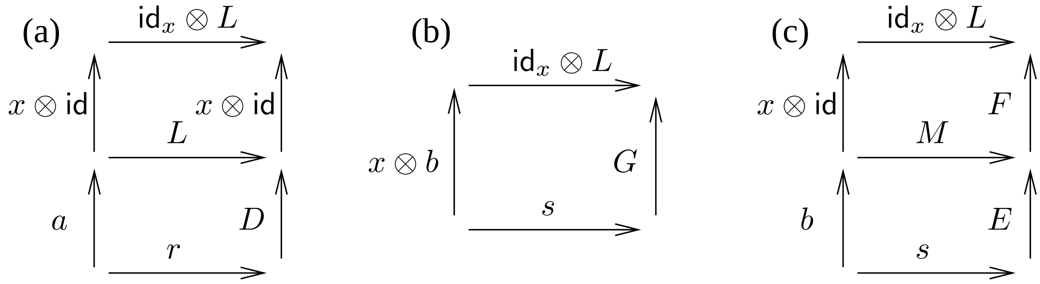
(2) To prove B_0 simple involves a routine check of the RPO construction. Next, we show that the IPO can contain no elisions. Since B_0 has no closed links there can be no elisions from A_0; and there can be no elisions from A_1 since it has no idle names. It follows that up to isomorphism there is a unique IPO for $\vec{A}$, so by Proposition 6.21 it is a pushout.

(3) Apply (2) to the IPO underlying each transition, since its redex is simple. ∎

These results make it easy to verify an important property of idle names. If we encode (say) a version of the π-calculus in link graphs, then a process term T is represented in every ground homset $\mathsf{Gr}(X)$ where X includes all the free names of T; this allows the possibility of bisimilarities $T \sim T'$ where the free names of T and T' differ. But we do not want the truth of this equation to depend on the chosen name-set X. We now show that this is avoided, at least in a simple LRS:

Proposition 7.8 (idle names and bisimilarity) *In a concrete LRS that is simple and equipped with standard transitions,* $a \sim b$ *iff* $x \otimes a \sim x \otimes b$.

Proof For the forward implication, use congruence. For the converse, we shall verify that $\mathcal{S} = \{(a,b) \mid x \otimes a \sim x \otimes b\}$ is a bisimulation up to $\bumpeq$.



Let $a\mathcal{S}b$, and let $a \xrightarrow{L} a'$. We seek a transition $b \xrightarrow{L} b'$ with $(a', b') \in \mathcal{S}^{\bumpeq}$.

The IPO underlying the transition of a is the bottom square of diagram (a), based on a rule (r, r') with $a' \bumpeq D \circ r'$. By Corollary 6.20 the upper square of (a) is also an IPO, hence so is the large square, and it represents a transition

$$x \otimes a \xrightarrow{\mathsf{id}_x \otimes L} x \otimes a' \; .$$

Since $x \otimes a \sim x \otimes b$, there is a rule (s, s') and a transition

$$x \otimes b \xrightarrow{\mathsf{id}_x \otimes L} G \circ s' \sim x \otimes a'$$

with underlying IPO as in diagram (b). Now $x \otimes b = (x \otimes \mathsf{id}) \circ b$, so by taking an RPO (M, E, F) for (b, s) we obtain a pair of IPOs as in (c). By Proposition 7.7(1) M is simple, and by Proposition 7.7(3) the upper square of (c) is a pushout. But by Corollary 6.20 the pair $(x \otimes \mathsf{id}, M)$ has a tensorial IPO $(\mathsf{id}_x \otimes M, x \otimes \mathsf{id})$; up to isomorphism this must coincide with the pushout, so without loss of generality we may assume $M = L$ and $F = \mathsf{id}$. We then find from the lower square that $b \xrightarrow{L} b' \stackrel{\text{def}}{=} E \circ s'$, and since $G = x \otimes E$ we have $G \circ s' = x \otimes b'$. So $(a', b') \in \mathcal{S}^{\bumpeq}$ as required. ∎

We now turn to engaged transitions; recall the discussion of them in Section 4.

Definition 7.9 (engaged transitions) A standard transition of a is said to be *engaged* if it can be based on a reaction with redex r such that $|a| \cap |r| \neq \emptyset$. We denote by ET the transition system of engaged transitions. We write $\sim^{\text{ET}}$ for $\sim^{\text{ET}}_{\text{ST}}$, bisimilarity for ET relative to ST. ■

Now we would like to prove that $\sim^{\text{ET}}$ is adequate for standard bisimilarity (Definition 4.15), i.e. that $\sim^{\text{ET}} = \sim$; for then to establish $a \sim b$ we need only match each *engaged* transition of a (resp. b) by an arbitrary transition of b (resp. a). This is a lighter task than matching *all* transitions.

In proving that $a \sim^{\text{ET}} b$ implies $a \sim b$ we have to show how b can match *all* transitions of a, and the antecedent only tells us how to match the *engaged* ones. However, it turns out that a non-engaged transition of a can be suitably matched by *any* b (whether or not $a \sim^{\text{ET}} b$). This is intuitively not surprising, because a contributes nothing to such a transition, so replacing it by b should not prevent the transition occurring.

Theorem 7.10 (adequacy of engaged transitions) *In a concrete LRS that is simple and equipped with* ST, *the engaged transitions are adequate; that is, engaged bisimilarity $\sim^{\text{ET}}$ coincides with bisimilarity $\sim$.*

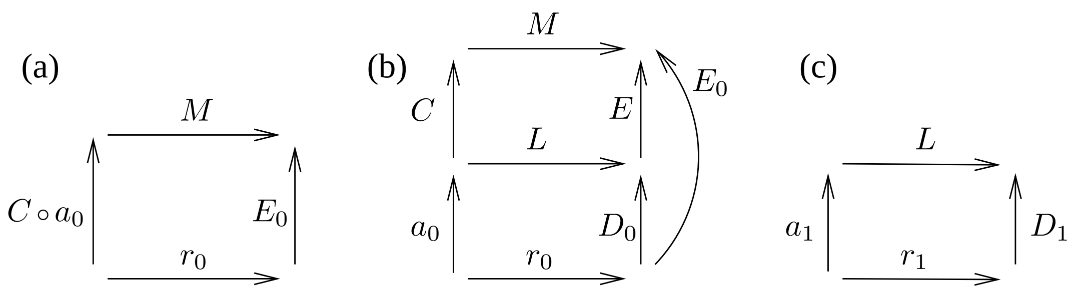
Proof It is immediate that $\sim \subseteq \sim^{\text{ET}}$. For the converse we shall show that

$$\mathcal{S} = \{(C \circ a_0, C \circ a_1) \mid a_0 \sim^{\text{ET}} a_1\}$$

is a standard bisimulation. Then, taking $C = \mathsf{id}$, we deduce $\sim^{\text{ET}} \subseteq \sim$.

Suppose that $a_0 \sim^{\text{ET}} a_1$. Let $C \circ a_0 \xrightarrow{M} b'_0$ be any standard transition, with $M \circ C \circ a_1$ is defined. We must find b'_1 such that $C \circ a_1 \xrightarrow{M} b'_1$ and $(b'_0, b'_1) \in \mathcal{S}$.

There exist a reaction rule (r_0, r'_0) and an underlying IPO as in diagram (a) below; moreover $b'_0 = E_0 \circ r'_0$. Then by taking RPOs we can complete diagram (b) so that every square is an IPO.



Hence $a_0 \xrightarrow{L} a'_0$ where $a'_0 = D_0 \circ r'_0$. Moreover, by Proposition 7.7(3), the lower square in diagram (b) is a pushout. Also $b'_0 = E \circ a'_0$.

Since $M \circ C \circ a_1$ is defined we deduce that $L \circ a_1$ is defined, and we proceed to show in two separate cases the existence of a transition $a_1 \xrightarrow{L} a'_1$, with underlying IPO as shown in diagram (c). (Note that we cannot immediately infer this from $a_0 \sim^{\text{ET}} a_1$, since the transition of a_0 may not lie in ET.) Substituting this diagram for the lower squares in (b), we can infer a transition $C \circ a_1 \xrightarrow{M} b'_1$ where $b'_1 = E \circ a'_1$. In each of the three cases we then argue that $(b'_0, b'_1) \in \mathcal{S}$, thus completing the proof of the theorem.

Case 1 Suppose the transition $a_0 \xrightarrow{L} a_0'$ is not engaged, i.e. $|a_0| \cap |r_0| = \emptyset$. The lower square of (b) is a pushout; hence it is the unique IPO (up to isomorphism) for a_0 and r_0, which by Corollary 6.20 must be a tensor IPO.[6] So up to isomorphism we have $L = \mathsf{id} \otimes r_0$ and $D_0 = a_0 \otimes \mathsf{id}$. Then we calculate

$$\begin{aligned} a_0' &= D_0 \circ r_0' = a_0 \otimes r_0' \\ &= E' \circ a_0 \text{ where } E' = \mathsf{id} \otimes r_0' . \end{aligned}$$

So in this case we take $D_1 = a_1 \otimes \mathsf{id}$ and $r_1 = r_0$ to form the IPO (c); hence

$$a_1 \xrightarrow{L} a_1' \stackrel{\text{def}}{=} E' \circ a_1 .$$

Then for the context $C' \stackrel{\text{def}}{=} E \circ E'$ we have $b_0' = C' \circ a_0$ and $b_1' = C' \circ a_1$; but $a_0 \sim^{\text{ET}} a_1$, so we have $(b_0', b_1') \in \mathcal{S}$ as required.

Case 2 Suppose the transition $a_0 \xrightarrow{L} a_0'$ is engaged, i.e. $|a_0| \cap |r_0| \neq \emptyset$. Then it lies in ET. But $a_0 \sim^{\text{ET}} a_1$, so there is a transition $a_1 \xrightarrow{L} a_1'$ for some a_1' such that $a_0' \sim^{\text{ET}} a_1'$; hence $C \circ a_1 \xrightarrow{M} b_1' \stackrel{\text{def}}{=} E \circ a_1'$, and thus $(b_0', b_1') \in \mathcal{S}$ as required. ∎

We now wish to transfer ET to abstract LRSs, via the functor

$$[\![\cdot]\!] : \acute{}\text{LIG}(\mathcal{K}) \to \text{LIG}(\mathcal{K}) .$$

To do this, we would like to know that ET is *definite* for ST (see Definition 4.16), for then by Proposition 4.17 we can equate the relative bisimilarity $\sim^{\text{ET}}_{\text{ST}}$ with the absolute one $\sim_{\text{ET}}$. For this, we need to know that, from the label L alone, we can determine whether or not a transition $a \xrightarrow{L} a'$ is engaged.

It turns out that this holds in a wide range of LRSs. This is because they all satisfy a simple structural condition, which we now define.

Definition 7.11 (proper LRS) Define $ctrl(G)$, the *control* of a link graph G, to be the multiset of controls of its nodes. A LRS is *proper* if for any two redexes r and s, if $ctrl(r) \subseteq ctrl(s)$ then $ctrl(r) = ctrl(s)$. ∎

Note that this property applies equally to concrete and abstract LRSs, and is indeed preserved and reflected by the quotient functor $[\![\cdot]\!]$. Moreover with the help of Corollaries 4.18 and 7.2, we deduce

Corollary 7.12 (engaged congruence) *In a concrete LRS that is both proper and simple:*

1. *The engaged transition system* ET *is definite for* ST.
2. *Engaged bisimilarity* $\sim_{\text{ET}}$ *coincides with standard bisimilarity.*
3. $\sim_{\text{ET}}$ *is a congruence, i.e.* $a \sim_{\text{ET}} b$ *implies* $C \circ a \sim_{\text{ET}} C \circ b$

[6] A forerunner of this phenomenon, that a non-engaged transition must be based upon a tensor IPO, appears in Leifer's PhD Dissertation [29], Theorem 3.33.

Now recall from Proposition 7.7 that every simple link graph is lean. We therefore specialise Corollary 7.5 to ET under appropriate assumptions:

Corollary 7.13 (engaged congruence in abstract LRSs) *Let $\acute{\mathbf{A}}$ be a concrete LRS that is proper and simple, and let $\mathbf{A}$ be its lean-support quotient. Let $\sim_{\mathrm{ET}}$ denote bisimilarity both for ET in $\acute{\mathbf{A}}$ and for the induced transition system $[\![\mathrm{ET}]\!]$ in $\mathbf{A}$. Then*

1. *$a \sim_{\mathrm{ET}} b$ in $\acute{\mathbf{A}}$ iff $[\![a]\!] \sim_{\mathrm{ET}} [\![b]\!]$ in $\mathbf{A}$.*
2. *Engaged bisimilarity $\sim_{\mathrm{ET}}$ is a congruence in $\mathbf{A}$.*

Proof The quotient functor satisfies the conditions of Theorem 4.14. In particular, by Proposition 7.4 it respects ET, since this is a sub-TS of ST. So the theorem yields (1) immediately. It also yields (2) with the help of Corollary 7.12. ■

Thus we have ensured congruence of engaged bisimilarity in any abstract LRS $\mathrm{LIG}(\mathcal{K})$ satisfying reasonable assumptions.

oOo

Part III
Sorting and condition-event nets

8 Sorted link graphs

Part III is devoted to the application of link graph theory. We begin in the present section with the topic of *sorting*, which is likely to be needed in any significant application. Then in Sections 9 and 10 we apply it, together with our theory of transitions systems, to deriving a behavioural congruence for a class of Petri nets.

Our sorting discipline for link graphs, first proposed for bigraphs in [37], is akin to many-sorted algebra and has a similar purpose: given a signature we wish to limit the entities that can be built with it. In algebra, these are often the algebraic terms that are meaningful for a particular interpretation; here, the same is true of link graphs. For example, in Petri nets it is not meaningful to connect two transition-nodes without an intervening place-node. Using a more sophisticated sorting discipline we can introduce a notion of *name-binding* into bigraphs [26]; this delimits the scope of a name, so that it cannot be linked to a port outside that scope.

In the following Θ will denote a non-empty set of *sorts*, and θ will range over Θ.

Definition 8.1 (sorted link graphs) A signature $\mathcal{K}$ is Θ-*sorted* if it is enriched by an assignment of a sort $\theta \in \Theta$ to each $i \in ar(K)$ for each control K. An interface X is Θ-*sorted* if it is enriched by ascribing a sort to each name $x \in X$.

A link graph is Θ-*sorted over* $\mathcal{K}$ if its interfaces are Θ-sorted, and for each K, i the sort assigned by $\mathcal{K}$ to $i \in ar(K)$ is ascribed to the i^{th} port of every K-node.

We denote by $\acute{\mathrm{L}}\mathrm{IG}(\Theta, \mathcal{K})$ the monoidal precategory of sorted link graphs whose identities, composition and tensor product are defined in the obvious way in terms of the underlying (unsorted) link graphs. ■

Note that sorts are ascribed to points and open links of a link graph, but not to its edges. We say *sorted* instead of Θ-sorted when Θ is understood.

We may wish to consider only those sorted link graphs that obey some condition:

Definition 8.2 (sorting) A *sorting (discipline)* is a triple $\Sigma = (\Theta, \mathcal{K}, \Phi)$ where $\mathcal{K}$ is Θ-sorted, and Φ is a condition on Θ-sorted link graphs over $\mathcal{K}$. The condition Φ must be satisfied by the identities and preserved by both composition and tensor product.

A link graph in $\acute{\mathrm{L}}\mathrm{IG}(\Theta, \mathcal{K})$ is said to be Σ-*sorted* if it satisfies Φ. The Σ-sorted link graphs form a monoidal sub-precategory of $\acute{\mathrm{L}}\mathrm{IG}(\Theta, \mathcal{K})$ denoted by $\acute{\mathrm{L}}\mathrm{IG}(\Sigma)$. Further, if $\acute{\mathcal{R}}$ is a set of Σ-sorted reaction rules then $\acute{\mathrm{L}}\mathrm{IG}(\Sigma, \acute{\mathcal{R}})$ is a Σ-*sorted* LRS. ■

We shall often say *well-sorted* instead of Σ-sorted when Σ is understood.

Even with only a single sort there are important examples; one example is *undirected linear* link graphs, where every open link contains exactly one point, and every closed link exactly two points. (The reader may like to confirm that this sorting satisfies the required conditions.) With two sorts, this condition can be refined to yield *directed linear* link graphs, where each port of each control has a polarity and a link must join ports only when their polarities are opposite. More generally, the purpose of a sorting is to dictate how nodes of a given (sorted) signature may be linked.

What constraints must we place on the sorting $\Sigma = (\Theta, \mathcal{K}, \Phi)$ in order that we may apply our transition theory? These constraints are best understood in terms of the obvious forgetful functor which discards sorts:

$$\mathcal{U} : \acute{}\textsc{Lig}(\Sigma, \acute{}\mathcal{R}) \to \acute{}\textsc{Lig}(\mathcal{U}(\mathcal{K}), \mathcal{U}(\acute{}\mathcal{R}))$$

We shall call $\mathcal{U}$ a *sorting* functor. Such functors have certain properties:

Proposition 8.3 (sorting is faithful) *On interfaces a sorting functor is surjective (but not in general injective); it is also* faithful, *i.e. injective (though not in general surjective) on each homset of link graphs.*

We need more structure than this if we wish to apply our transition theory to a well-sorted LRS. Consider two properties that a functor of precategories may have:

Definition 8.4 (creating RPOs, reflecting pushouts) Let $\mathcal{F}$ be any functor on a precategory $\acute{}\mathbf{A}$. Then $\mathcal{F}$ *creates RPOs* if, whenever $\vec{D}$ bounds $\vec{A}$ in $\acute{}\mathbf{A}$, then any RPO for $\mathcal{F}(\vec{A})$ relative to $\mathcal{F}(\vec{D})$ has a unique $\mathcal{F}$-preimage that is an RPO for $\vec{A}$ relative to $\vec{D}$.

$\mathcal{F}$ *reflects pushouts* if, whenever $\vec{D}$ bounds $\vec{A}$ in $\acute{}\mathbf{A}$ and $\mathcal{F}(\vec{B})$ is a pushout for $\mathcal{F}(\vec{A})$, then $\vec{B}$ is a pushout for $\vec{A}$. ■

Corollary 8.5 (creation ensures RPOs) *If* $\mathcal{F} : \acute{}\mathbf{A} \to \acute{}\mathbf{B}$ *creates RPOs and* $\acute{}\mathbf{A}$ *has RPOs, then* $\acute{}\mathbf{B}$ *has RPOs.*

We shall often confuse Σ with its functor; for example we say 'Σ reflects ...' etc.

It turns out that if a sorting satisfies the two conditions of Definition 8.4 (which appear to be independent, but we need not settle that question here) then we get sufficient structure for our transition theory:

Theorem 8.6 (useful sortings)

1. *If* Σ *creates RPOs then bisimilarity for the standard transition system* ST *over* $\acute{}\textsc{Lig}(\Sigma, \acute{}\mathcal{R})$ *is a congruence.*
2. *If in addition* Σ *reflects pushouts and* $\acute{}\mathcal{R}$ *is simple, then the engaged transitions are adequate for* ST.

Note that *simpleness* of a well-sorted link graph is just as for a pure one. (Indeed sorting functors both preserve and reflect simpleness.) We omit the proof of the theorem; it follows closely the lines of Theorems 4.12 and 7.10; for the latter, the reflection of pushouts enables Proposition 7.7 to be lifted to the well-sorted LRS.

We are now ready to define the sorting discipline we shall use in the remainder of the paper. It may be motivated by our arithmetic nets, in which we want to each link to contain any number of 'input' ports, but at most one 'output' port. The formal definition must also constrain the sorting of interfaces. Recall that in a link graph $G : X \to Y$ a *point* is either an inner in X or a port, while a *closed link* is an edge and an *open link* is an outer name in Y.

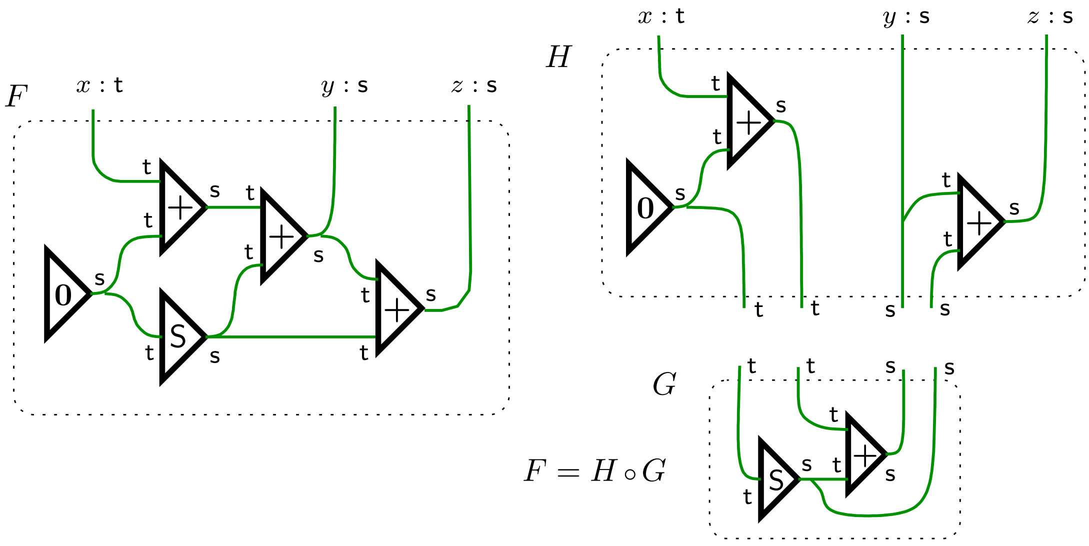


Figure 9: A well-sorted arithmetic net and its decomposition

Definition 8.7 (many-one sorting) In a *many-one sorting* $\Sigma = (\Theta, \mathcal{K}, \Phi)$ the sorts are $\Theta = \{\mathsf{s}, \mathsf{t}\}$, the signature $\mathcal{K}$ is arbitrary with an arbitrary assignment of sorts to control arities, and the condition Φ is as follows:

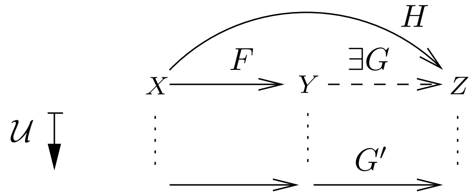
- a closed link has exactly one s-point;
- an open s-link has exactly one s-point;
- an open t-link has no s-points.

There is no constraint on the number of t-points in a link. ■

It is helpful to think of s and t as standing for 'source' and 'target'.

Let us illustrate by considering arithmetic nets. In this case the sorted signature is $\mathcal{K}_{\mathsf{arith}}$ as defined at the beginning of Section 6, enriched by the assignment of s to to output ports and t to input ports; for example, $+$ is assigned the sort-sequence tts. Figure 9 shows the net of Figure 6, but now with sort ascriptions; the reader may like to check that it obeys the many-one sorting discipline.

A many-one sorted LRS has a nice property not shared by all sortings:



Proposition 8.8 (many-one sorted decomposition) *Let $\mathcal{U}$ be a many-one sorting functor, and let*

$$\mathcal{U}(H : X \to Z) = G' \circ \mathcal{U}(F : X \to Y)\ .$$

Then there exists $G : Y \to Z$ such that $\mathcal{U}(G) = G'$ and $H = G \circ F$.

Note that, since $\mathcal{U}$ is faithful, G exists uniquely. (Thus, in category-theoretic terms, the proposition says that every arrow F is opcartesian.) With the help of this proposition is not hard to show that many-one sorting has the structure we need:

Theorem 8.9 (many-sorting structure) *Every many-one sorting discipline creates RPOs and reflects pushouts.*

Proof (outline) For the first property, it can be shown that if we apply Construction 6.10 to a well-sorted pair $\vec{A}$ with a well-sorted bound $\vec{D}$, then the resulting RPO is itself well-sorted; also, the existence of a mediator to any other well-sorted candidate is assured by Proposition 8.8.

The second property can be proved for *any* functor of precategories that is faithful and enjoys the property in Proposition 8.8. ∎

We are now ready to induce a behavioural congruence for condition-event Petri nets, since they can be modelled as a many-one sorted LRS.

9 Condition-event nets as link graphs

We begin this section with a digression from link graphs, in order to discuss the behaviour of Petri nets in their own terms. First we consider some recent papers on behavioural equivalences on Petri nets.

Pomello, Rozenberg and Simone [42] give a comprehensive survey of such equivalences and preorders. They cover those based on observation both of actions and of states, and range from fine relations respecting causality to coarser ones, for example the failures preorder from CSP, the coarsest which respects deadlock. The study of congruence of these relations, i.e. whether they are preserved by contexts, and which contexts *should* preserve them, is reported as being rather incomplete at that date (1992).

Nielsen, Priese and Sassone [40] characterise some behavioural congruences on nets. Given a semantic function $\mathcal{B}$ that assigns an abstract behaviour to each net, they consider the congruence $\approx$ it induces upon nets; this is defined by

$$N_0 \approx N_1 \overset{\text{def}}{\Leftrightarrow} \mathcal{B}(C[N_0]) = \mathcal{B}(C[N_1]) \text{ for every context } C\ .$$

An important contribution of their paper is to define a precise notion of context, by means of a set of *combinators* upon nets. They are then able to characterise the congruences, for each of four semantic functions $\mathcal{B}$, by showing that for each pair N_0, N_1 there is a single easily identified context that is sufficient to determine whether or not $N_0 \approx N_1$.

Priese and Wimmel [43] continue this programme; they enrich the net combinators, and consider a wider range of semantic functions.

The Petri Box calculus of Best, Devillers and Hall [4], like the previous two approaches, emphasises combinators and algebra. By identifying certain net-patterns as operators, it presents a modular semantics of nets in terms of equivalence classes of Boxes (a special class of nets). A main result of the paper is agreement between this denotational semantics and a structured operational semantics of Box expressions.

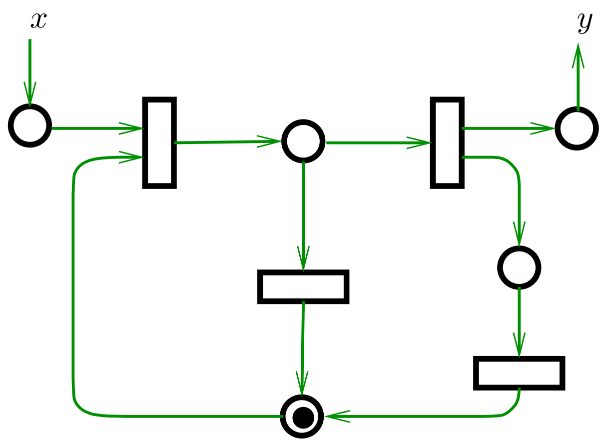


Figure 10: A condition-event net with two observable conditions

Baldan, Corradini, Ehrig and Heckel [1] define a class of *open* Petri nets, having input and output places where tokens may be respectively added and removed at any time. They define a form of composition of two such nets which allows interaction at these places, and define a semantics of a net in terms of its *processes*, i.e. the deterministic nets representing its possible behaviours. The semantics is shown to be compositional, i.e. the composition of two open nets respects their underlying processes.

This brief summary does not do justice to the five papers, which represent well the progress towards a modular treatment of Petri nets. But it helps us to identify differences with the theory of bigraphs (or link graphs), which suggest contributions that can be made by the latter. The first difference is that, since bigraphs and their contexts are the arrows of a (pre)category, whenever a class of agents —e.g. Petri nets— is encoded in bigraphs the contexts and combinators are thereby determined; they need not be defined specifically for each class. The second difference is that the semantic function on bigraphical agents is defined not by specific means, but as the quotient by a generic equivalence relation that pertains to *all* bigraphical systems. Finally, many such equivalences —including bisimulation (which we use in this paper) but also others— are guaranteed by bigraphical theory to be congruences.

After this brief review, let us now consider *condition-event* Petri nets, as illustrated in Figure 10. These are nets in which each place, or condition, may be either marked (i.e. holding a single token) or unmarked. The usual firing rule for condition-event nets is as follows:

> an event with all pre-conditions and no post-conditions marked may 'fire', unmarking its pre-conditions and marking its post-conditions.

The firing rule describes what can happen inside a net, but does not indicate how this net behaviour may be observed or controlled from outside. So we shall set up a simple observational discipline, yielding a labelled transition system and hence inducing a bisimilarity equivalence. This discipline is one of many possible, and it differs from those in the above-cited papers, but is nevertheless quite natural. It provides a good

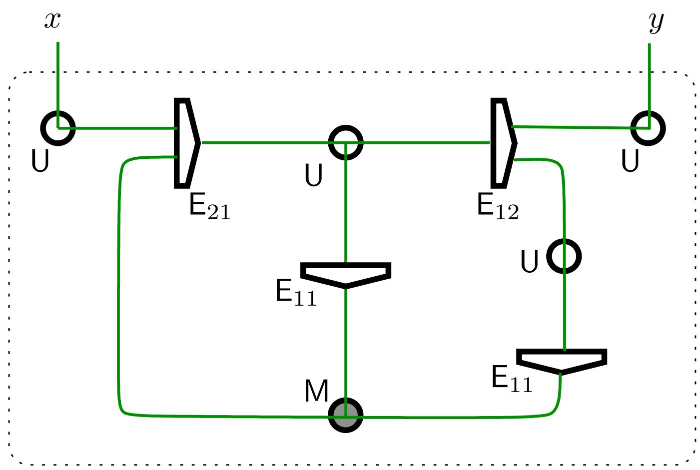


Figure 11: A condition-event net represented as a link graph

case study in link graphs, since we can compare an equivalence expressible in Petri net terms with one induced by link graph theory.

How may we conduct experiments, or observations, on a condition-event net? One way, akin to the approach of Baldan et al [1], is to make certain conditions externally accessible, allowing the observer both to detect and to change the state (marked or unmarked) of the place. For example, the net in Figure 10 has two accessible conditions, named x and y. In general, given a state g, i.e. a net together with a marking of its conditions, the transition $g \xrightarrow{+x} \overline{g}$ or $g \xrightarrow{-x} \overline{g}$ represents the addition or subtraction of a token at x. Since we are dealing with condition-event nets, in any given state exactly one of these experiments is possible for each accessible condition. A third kind of transition, $g \xrightarrow{\tau} \overline{g}$, represents (the firing of) an internal event and involves no external participation. These three kinds of transition are the basis of a *raw* TS $\mathcal{L}_{\mathsf{p}}$, with which we shall equip our LRS of Petri nets, in order to compare it with another TS $\mathcal{L}_{\mathsf{g}}$ which we shall derive from reaction rules by the methods discussed in Parts I and II of this paper.

We now set up condition-event nets as link graphs. There are many ways to do it; we choose one that gives a smooth treatment. Figure 11 shows the net of Figure 10 as a link graph, using the signature $\mathcal{K}_{\mathsf{petri}}$ defined at the start of Section 6 and illustrated in Figure 5. Recall the three kinds of control: M ('marked') and U ('unmarked') for conditions, and E_{hk} for events. The shape and shading of nodes will save us from writing controls in diagrams. A condition-node has a single port, which we site in its centre. An E_{hk} event-node has $h+k$ ports; h for pre-conditions, and k for post-conditions. You may check that the above net has two open and three closed links.

Now we enrich $\mathcal{K}_{\mathsf{petri}}$ by assigning the sort s to all condition ports and t to event ports. This leads us to the sorting discipline

$$\Sigma_{\mathsf{petri}} \stackrel{\text{def}}{=} (\Theta_{\mathsf{petri}}, \mathcal{K}_{\mathsf{petri}}, \Phi_{\mathsf{petri}})$$

where $\Theta_{\mathsf{petri}} = \{\mathsf{s}, \mathsf{t}\}$ and Φ_{petri} is the many-one sorting condition of Definition 8.7.

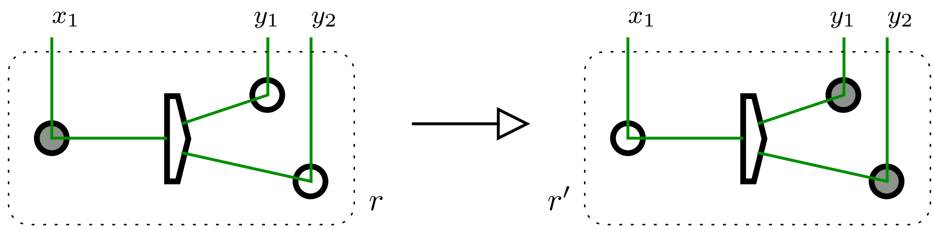


Figure 12: A link-graph reaction rule for condition-event nets

Then the concrete precategory of many-one sorted condition-event nets is

$$\acute{\mathbf{CE}} \stackrel{\text{def}}{=} \acute{\mathrm{L}}\textsc{ig}(\Sigma_{\mathsf{petri}})$$

and we denote its lean-support quotient by **CE**. Although these nets share many-one sorting with arithmetic nets, there is a considerable difference; this arises from the fact that in arithmetic nets every node possesses exactly one s-port, while in $\acute{\mathbf{CE}}$ the event nodes have none. This illustrates the versatility of many-one sorting.

In general an interface may contain both s-names and t-names. But in the example both x and y are s-names, because each is a link containing a condition. So let us define an s-*interface* to be one containing only s-names; then we can model condition-event nets in $\acute{\mathbf{CE}}$ and **CE** as link graphs with s-interfaces, and call them s-*nets*.

Without further ado we now set up in $\acute{\mathbf{CE}}$ a raw transition system $\mathcal{L}_{\mathsf{p}}$, whose interfaces are s-interfaces and whose transitions $a \xrightarrow{\ell} b$ are those we have already described with $\ell = +x, -x$ or τ. We also close the transitions under support equivalence. This induces a TS $[\![\mathcal{L}_{\mathsf{p}}]\!]$ in **CE**. Let us use $\sim_{\mathsf{p}}$ for the associated bisimilarity in both cases. Since no RPO theory is involved, we readily find

Proposition 9.1 (raw bisimilarity)

1. *$a \xrightarrow{\ell} a'$ in $\acute{\mathbf{CE}}$ iff $[\![a]\!] \xrightarrow{\ell} [\![a']\!]$ in* **CE***.*
2. *$a \sim_{\mathsf{p}} b$ in $\acute{\mathbf{CE}}$ iff $[\![a]\!] \sim_{\mathsf{p}} [\![b]\!]$ in* **CE***.*

To compare this raw TS and bisimilarity with a contextual one, we must add reaction rules to $\acute{\mathbf{CE}}$, to make it an LRS. To match the firing rule, for each pair h, k we introduce a reaction rule for E_{hk} as illustrated in Figure 12 for $h = 1, k = 2$. As required by Definitions 4.2 and 7.1, we close this set under support translation and make each rule lean (no idle edges). Having thus established $\acute{\mathbf{CE}}$ as a concrete LRS, we equip it with the standard transition system ST. We can then apply Corollary 7.2 to establish that the associated bisimilarity $\sim_{\mathrm{g}}$, is a congruence.

Now we wish to refine the transition system in two steps. The first step is to reduce its transitions to the engaged ones.

Proposition 9.2 (adequacy for nets) *The engaged transition system* ET *over $\acute{\mathbf{CE}}$ is definite and adequate for* ST*; therefore its bisimilarity coincides with $\sim_{\mathrm{g}}$.*

Proof It is easy to show that $\acute{\mathbf{C}}\mathbf{E}$ is simple, as defined in Definition 7.6. It is also proper, according to Definition 7.11. Therefore by Corollary 7.12 we may reduce ST to ET without affecting the induced bisimilarity $\sim_{\mathrm{g}}$. ■

The second refinement step is to reduce the agents to s-nets. We define the TS $\mathcal{L}_{\mathrm{g}}$ to consist of s-interfaces together with all engaged transition between s-nets. Now, since every redex and reactum is an s-net, we find that in any standard transition $a \xrightarrow{L} a'$, if a is an s-net then so are L and a'. It follows that $\mathcal{L}_{\mathrm{g}}$ is a full sub-TS of ET. Therefore by Proposition 4.19 and Corollary 4.18 we have the following:

Corollary 9.3 (bisimulation congruence for concrete s-nets) *Bisimilarity for the transition system $\mathcal{L}_{\mathrm{g}}$ coincides with $\sim_{\mathrm{g}}$ on s-nets and is a congruence.*

We have now taken the theory of $\mathcal{L}_{\mathrm{g}}$ for concrete s-nets as far as we need, except for characterising its transitions. We leave that task to Section 10. Here, noting that $\mathcal{L}_{\mathrm{g}}$ respects lean-support equivalence, we relate it to the TS $[\![\mathcal{L}_{\mathrm{g}}]\!]$ induced on abstract s-nets, using Corollaries 9.3 and 7.5:

Corollary 9.4 (bisimulation congruence for abstract s-nets) *Denote by $\sim_{\mathrm{g}}$ the bisimilarity induced on* **CE** *by the abstract TS $[\![\mathcal{L}_{\mathrm{g}}]\!]$. Then*

1. *$a \sim_{\mathrm{g}} b$ in $\acute{\mathbf{C}}\mathbf{E}$ iff $[\![a]\!] \sim_{\mathrm{g}} [\![b]\!]$ in* **CE**.
2. *The bisimilarity $\sim_{\mathrm{g}}$ is a congruence in* **CE**.

10 Coincidence of bisimilarities

We are now ready to examine the behaviour of s-nets. In $\acute{\mathbf{C}}\mathbf{E}$ this is given both by a raw TS $\mathcal{L}_{\mathrm{p}}$ with associated bisimilarity $\sim_{\mathrm{p}}$ and by a contextual TS $\mathcal{L}_{\mathrm{g}}$ with associated bisimilarity $\sim_{\mathrm{g}}$. These induce in **CE** the TSs $[\![\mathcal{L}_{\mathrm{p}}]\!]$ and $[\![\mathcal{L}_{\mathrm{g}}]\!]$, whose associated bisimilarities are again denoted by $\sim_{\mathrm{p}}$ and $\sim_{\mathrm{g}}$.

Our main concern is to compare these two *abstract* bisimilarities, but we shall do the work mainly in *concrete* s-nets since it involves a little RPO theory. At the end the comparison is transported easily to abstract s-nets.

Our first task is to characterise the labels of $\mathcal{L}_{\mathrm{g}}$. We omit the detailed analysis; it uses the fact that transitions are engaged (Proposition 9.2) and that labels are simple (Proposition 7.7) and have s-interfaces. It turns out that, up to isomorphism, a label takes two forms: either it is an identity, or it is an open s-net with exactly one E-node, linked to zero or more M-nodes as preconditions and U-nodes as post-conditions.

For the identity labels, we recall from Proposition 6.21 that $a \xrightarrow{\mathsf{id}} a'$ iff $a \longrightarrow a'$; an id label signifies a transition with no help from the context.

Figure 13 shows a non-identity label; the dashed link indicates an identity on zero or more names. A label can be thought of as a redex-fragment, lacking some conditions; in the example it requires its client agent to provide one marked pre-condition and one unmarked post-condition. Figure 14 shows the anatomy of a transition $a \xrightarrow{L} a'$ with this label. Note that a' takes the form $\overline{L} \circ \overline{a}$. In what follows we shall often use the

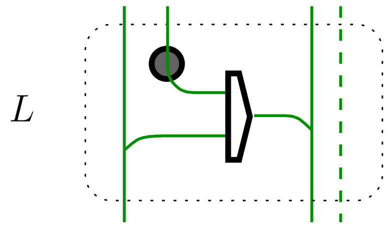


Figure 13: A typical label in $\mathcal{L}_{\mathrm{g}}$

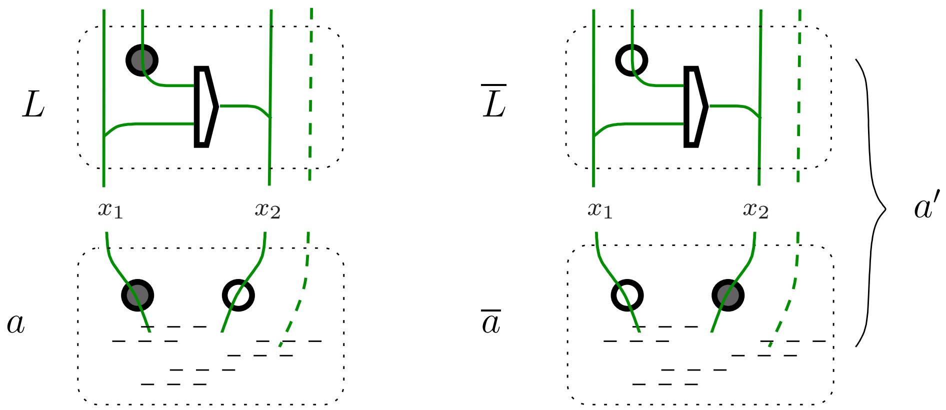


Figure 14: Anatomy of a transition $a \xrightarrow{L} a'$ in $\mathcal{L}_{\mathrm{g}}$

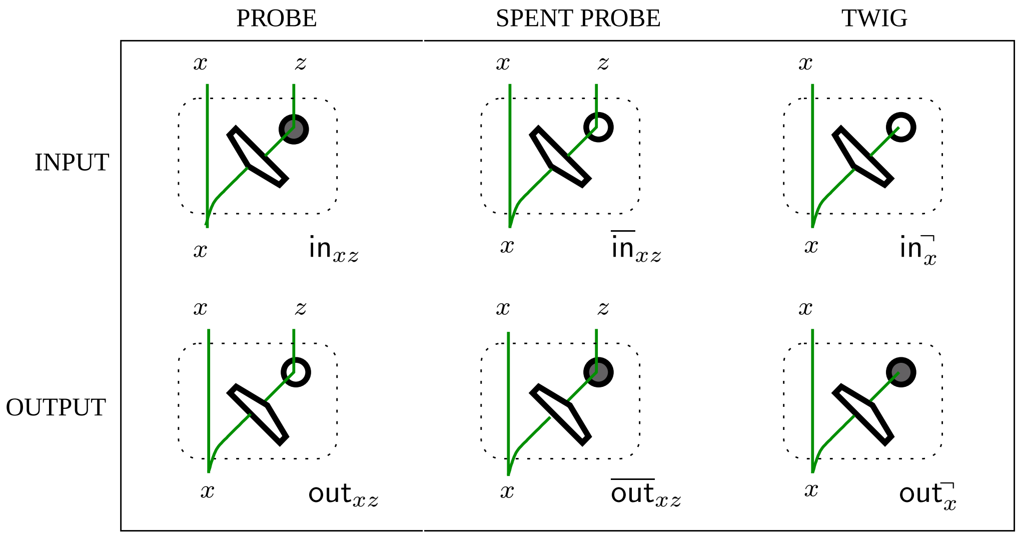


Figure 15: Probes for observing conditions in a s-net

notation $\overline{a}$ to denote a s-net that differs from a only by the marking of some conditions; we call it a *residual* of a.

We see that a single transition may change the marking of several named conditions of a, however far apart they may lie in a. Any other agent b with the same interface as a will have a similar transition, provided only that it has the same initial marking of its named conditions.

The two TSs $\mathcal{L}_\mathsf{p}$ and $\mathcal{L}_\mathsf{g}$ are significantly different, so it is not clear that they will induce the same bisimilarity. We shall now prove that they do so.

We shall first show that $\sim_\mathsf{g} \subseteq \sim_\mathsf{p}$ in ´**CE**. This asserts that if we can distinguish two s-nets a and b by using 'experiments' ℓ of the form $+x$, $-x$ or τ, then we can also do so using 'experiments' L that are link graph contexts. So, among the labels L generated by our theory (see Figure 13), we need to find those that can do the job of the experiments $+x$, $-x$ and τ.

It turns out that labels to mimic an experiment $+x$ or $-x$ need only involve E_{11} events, those with one pre- and one post-condition; they take the form $P \otimes \mathsf{id}$, where P is respectively an *input* or *output probe*. The probes are denoted by in_{xz} and out_{xz}, and are shown in the first column of Figure 15. The second column shows the *spent* probes $\overline{P}$, residuals of the probes. The third column shows the spent probes with their conditions closed; they are defined by $\mathsf{in}^{\neg}_x \stackrel{\text{def}}{=} /z \circ \overline{\mathsf{in}}_{xz}$ and $\mathsf{out}^{\neg}_x \stackrel{\text{def}}{=} /z \circ \overline{\mathsf{out}}_{xz}$. We shall call them *twigs* because, up to the equivalence $\sim_\mathsf{g}$, they can be 'broken off'. The intuition is simply that a twig occurring anywhere in a net can never fire. We express this formally as follows:

Lemma 10.1 *For any* s*-agent f having x in its outer face,* $\mathsf{in}^{\neg}_x \circ a \sim_\mathsf{g} \mathsf{out}^{\neg}_x \circ a \sim_\mathsf{g} a$.

Here we have abbreviated $\mathsf{in}^{\neg}_x \otimes \mathsf{id}$ to $\mathsf{in}^{\neg}_x$; we shall use such abbreviations in what follows, but only in a composition which determines the identity id.

Now to prove that $\sim_{\mathsf{g}} \subseteq \sim_{\mathsf{p}}$ it is enough to show that $\sim_{\mathsf{g}}$ is an $\mathcal{L}_{\mathsf{p}}$-bisimulation. For this, suppose that $a \sim_{\mathsf{g}} b$, and let $a \xrightarrow{\ell} \overline{a}$ in $\mathcal{L}_{\mathsf{p}}$. We must find $\overline{b}$ such that $b \xrightarrow{\ell} \overline{b}$ and $\overline{a} \sim_{\mathsf{g}} \overline{b}$. If $\ell = \tau$ this is easy, because then our assumption implies that $a \longrightarrow \overline{a}$, and hence $a \xrightarrow{\mathsf{id}} \overline{a}$ in $\mathcal{L}_{\mathsf{g}}$; but then by bisimilarity in $\mathcal{L}_{\mathsf{g}}$ we have $b \xrightarrow{\mathsf{id}} \overline{b} \sim_{\mathsf{g}} \overline{a}$, and by reversing the reasoning for a we get that $b \xrightarrow{\tau} \overline{b}$ and we are done.

Now let $\ell = +x$ (the case for $-x$ is dual), so that $a \xrightarrow{+x} \overline{a}$. This means that a has an unmarked condition named x, so that in $\mathcal{L}_{\mathsf{g}}$ we have

$$a \xrightarrow{\mathsf{in}_{xz} \otimes \mathsf{id}} a' = \overline{\mathsf{in}}_{xz} \circ \overline{a} \;.$$

Hence by bisimilarity in $\mathcal{L}_{\mathsf{g}}$ we have

$$b \xrightarrow{\mathsf{in}_{xz} \otimes \mathsf{id}} b' = \overline{\mathsf{in}}_{xz} \circ \overline{b}$$

where $a' \sim_{\mathsf{g}} b'$ and $\overline{b}$ is the residual of b under the transition. This residual $\overline{b}$ differs from b only in having a marked condition named x that was unmarked in b, and hence we also have $b \xrightarrow{+x} \overline{b}$ in $\mathcal{L}_{\mathsf{p}}$. It remains only to show that $\overline{a} \sim_{\mathsf{g}} \overline{b}$. We deduce this using the congruence of $\sim_{\mathsf{g}}$ and Lemma 10.1:

$$\begin{array}{rll} \overline{a} & \sim_{\mathsf{g}} & \mathsf{in}^{\neg}_{x} \circ \overline{a} \;=\; /z \circ \overline{\mathsf{in}}_{xz} \circ \overline{a} \;=\; /z \circ a' \\ & \sim_{\mathsf{g}} & /z \circ b' \;=\; /z \circ \overline{\mathsf{in}}_{xz} \circ \overline{b} \;=\; \mathsf{in}^{\neg}_{x} \circ \overline{b} \\ & \sim_{\mathsf{g}} & \overline{b} \;. \end{array}$$

Therefore we have proved what we wished:

Lemma 10.2 $\sim_{\mathsf{g}} \subseteq \sim_{\mathsf{p}}$ *in* ´**CE**.

To complete our theorem we must prove the converse, $\sim_{\mathsf{p}} \subseteq \sim_{\mathsf{g}}$. It will be enough to prove that

$$\mathcal{S} \stackrel{\text{def}}{=} \{\, (C \circ a, C \circ b) \mid a \sim_{\mathsf{p}} b \,\}$$

is a bisimulation up to $\simeq$. We get the required result by considering the case $C = \mathsf{id}$.

We shall make use of the close correspondence between transitions in the concrete and abstract LRSs, respectively ´**CE** and **CE**. Further we shall use the convenient fact that, in ´**CE**, every IPO is actually a pushout by Proposition 7.7(3).

So let us assume that $a \sim_{\mathsf{p}} b$, and that $C \circ a \xrightarrow{M} a''$ in $\mathcal{L}_{\mathsf{g}}$. (This covers the case that $M = \mathsf{id}$.) Then there is a reaction rule r and context D such that (M, D) forms a pushout for $(C \circ a, r)$, as shown in the left-hand diagram of Figure 16, and $a'' \simeq D \circ r'$. We now take the pushout (L, F) for (a, r), and properties of pushouts yield the right-hand diagram, in which the upper square is also a pushout. So there is a transition $a \xrightarrow{L} a'$, where $a' \simeq F \circ r'$; note also that $a'' \simeq C' \circ a'$. Up to isomorphism, L is either an identity or a non-identity label.

If $L = \mathsf{id}$ then $a \longrightarrow a'$, hence $a \xrightarrow{\tau} a'$ in $\mathcal{L}_{\mathsf{p}}$. Since $a \sim_{\mathsf{p}} a'$ we have $b \xrightarrow{\tau} b'$ with $a' \sim_{\mathsf{p}} b'$. Then also $b \xrightarrow{L} b'$, with underlying pushout as in the left-hand diagram of Figure 17. We then proceed, as in the non-identity case below, to construct the right-hand diagram and to find b'' with $C \circ b \xrightarrow{M} b''$ and $(a'', b'') \in \mathcal{S}^{\simeq}$.

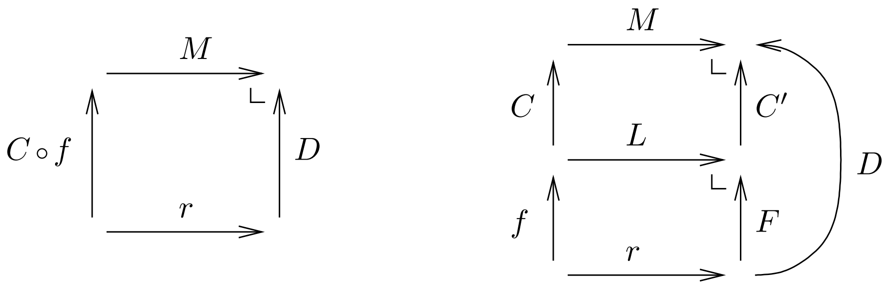


Figure 16: Pushouts underlying transitions of $C \circ a$ and a

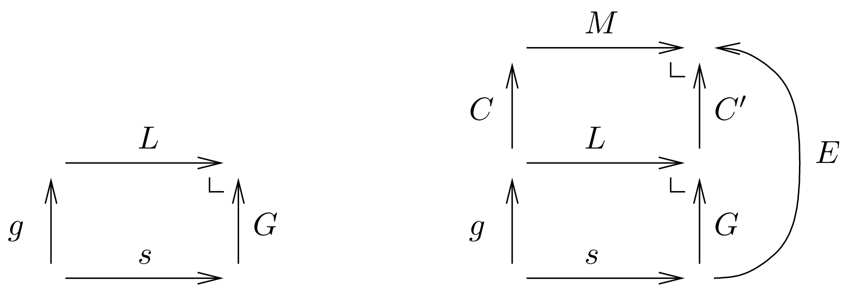


Figure 17: Pushouts underlying transitions of b and $C \circ b$

If L is a non-identity label we consider the anatomy of the transition $a \xrightarrow{L} a'$, as exemplified in Figure 14. We know that the residual $\overline{a}$ differs from a only in the changed marking of zero or more named conditions. It follows therefore that in $\mathcal{L}_{\mathsf{p}}$ there is a sequence of transitions

$$a \xrightarrow{\ell_1} a_1 \ldots \xrightarrow{\ell_n} a_n = \overline{a} \qquad (n \geq 0)$$

where $\ell_i \in \{+x_i, -x_i\}$; each transition marks or unmarks a single named condition. Moreover $a' = \overline{L} \circ \overline{a}$. Since $a \sim_{\mathsf{p}} b$ there exists a similar sequence

$$b \xrightarrow{\ell_1} b_1 \ldots \xrightarrow{\ell_n} b_n = \overline{b}$$

with $\overline{a} \sim_{\mathsf{p}} \overline{b}$. This implies that b has the same initial marking as a for the named conditions involved in the transitions. But we know that $L \circ b$ is defined (since we assumed $M \circ C \circ b = C' \circ L \circ b$ to be defined), so in $\mathcal{L}_{\mathsf{g}}$ there is a transition $b \xrightarrow{L} b' = \overline{L} \circ \overline{b}$. Its underlying pushout is shown in the left-hand diagram of Figure 17. Also it has an underlying reaction rule (s, s'), with $b' \simeq G \circ s'$.

Now we form the right-hand diagram of Figure 17 by replacing this pushout for the lower square in right-hand diagram of Figure 16. Since both small squares are pushouts, so is the large square; therefore it underlies an $\mathcal{L}_{\mathsf{g}}$-transition

$$C \circ b \xrightarrow{M} b'' \stackrel{\text{def}}{=} E \circ s' \ .$$

To complete our proof we need only show that the pair (a'', b'') lies in $\mathcal{S}^{\bumpeq}$. We already know that $a'' \bumpeq C' \circ a' = C' \circ \overline{L} \circ \overline{a}$. We can now compute

$$b'' = E \circ s' = C' \circ G \circ s' \bumpeq C' \circ b' = C' \circ \overline{L} \circ \overline{b}\ ,$$

and hence $(a'', b'') \in \mathcal{S}^{\bumpeq}$ since $\overline{a} \sim_{\mathsf{p}} \overline{b}$. It follows that $\sim_{\mathsf{p}} \subseteq \sim_{\mathsf{g}}$.

So we have proved the coincidence of our two bisimilarities:

Theorem 10.3 (coincidence of concrete bisimilarities) *In* ´**CE** *the two bisimilarities* $\sim_{\mathsf{g}}$ *and* $\sim_{\mathsf{p}}$ *for concrete* s*-nets coincide. Hence, since* $\sim_{\mathsf{g}}$ *is a congruence, so also is* $\sim_{\mathsf{p}}$ *is a congruence.*

It remains to transfer this to abstract s-nets. But this is immediate by Proposition 9.1 and Corollary 9.4, and finally we have the result we set out to prove:

Corollary 10.4 (coincidence of abstract bisimilarities) *In* **CE** *the two bisimilarities* $\sim_{\mathsf{g}}$ *and* $\sim_{\mathsf{p}}$ *for abstract* s*-nets coincide. Hence, since* $\sim_{\mathsf{g}}$ *is a congruence, so also is* $\sim_{\mathsf{p}}$ *is a congruence.*

It is worth noting that since $\mathcal{L}_{\mathsf{p}}$ and $\sim_{\mathsf{p}}$ were defined without reference to link graphs, it was not clear which contexts would preserve $\sim_{\mathsf{p}}$, i.e. in what sense $\sim_{\mathsf{p}}$ would be a congruence. Thus link graph theory can claim to have provided a convincing answer to these questions, by means of an alternative characterisation of $\sim_{\mathsf{p}}$.

---

oOo

11 Related and future research

We conclude by commenting on related work that has not already been mentioned in the text; at the same time we point to some future directions for our own research.

In this paper we have limited our attention to link graphs, which are one constituent of bigraphs, and have applied them to Petri nets where the other constituent —place graphs— is not needed. The technical report by Jensen and Milner [26] pursues a similar programme for full bigraphs, giving a full analysis of a finite asynchronous π-calculus as reported earlier at a conference [25]. In his forthcoming PhD Dissertation [24] Jensen will carry out this analysis not only for the full π-calculus but also for the ambient calculus.

Leifer, in his Dissertation [29], extended the present congruence results for strong bisimilarity to many other behavioural relations, including weak bisimilarity and the failures preorder; these results will be published separately. Jensen in his Dissertation is also extending Leifer's treatment of weak transitions.

The long tradition of graph-rewriting is based upon the *double pushout* (DPO) construction originated by Ehrig [14]. Our use of (relative) pushouts to derive transitions is quite distinct from the DPO construction, whose purpose is to explain the anatomy of graph-rewriting rules (not labelled transitions) working in a category of graph embeddings where the objects are graphs and the arrows are embeddings. This contrasts with our contextual s-categories, where the objects are interfaces and arrows are graphs. But there are links between these formulations, both via cospans [18] and via a categorical isomorphism between graph embeddings and a coslice over our contextual s-categories [12]. Ehrig [15] has investigated these links further. This has led to paper by Ehrig and König [16] in which the RPO technique is transferred to graph-embedding categories.

Sassone and Sobocinski [48] have generalised RPOs to *groupoid* RPOs, in a 2-category whose 2-cells (i.e. arrows between arrows) are isomorphisms. They advocate treating graphical and other dynamic entities as arrows in such a 2-category; the 2-cells keep track of the identity of nodes (which is essential for RPOs to exist) and have the potential to serve as witnesses for rich structural congruences. An advantage of their approach over s-categories is that composition is total, though this comes at the cost of a more complicated notion of "2-RPO". Our s-categories are well-behaved, and lend themselves easily to the detailed analysis of transitions in the particular case of bigraphs and link graphs, e.g. the characterisation of all IPOs for a given span (Theorem 6.18). Thus for our own work the motivation to pass to 2-categories is hitherto weak; however, the 2-categorical approach clearly deserves further investigation for these and other non-trivial applications.

The 'dualism' of graphs-as-arrows versus graphs-as-objects deserves further comment. From the graph-rewriting perspective the latter is considered basic, and indeed embeddings as arrows are a natural way to distinguish different occurrences of one entity within another. From the process calculus perspective, it is normal to represent processes as terms of an algebra; one reason is the composition of such terms aligns well with the composition of programs, and indeed there are good examples of programming languages derived from process calculi. 'Bigraphs-as-arrows' can be seen as an instance (or an enrichment) of Lawvere's algebraic theories [28], the standard categorical treatment of algebra. In this spirit, Milner [35] has completely axiomatised the algebra of pure bigraphs.

The case-studies on deriving transitions from reaction rules, in both the π-calculus and Petri nets, have shown an interesting mismatch with existing (or putative) transitions defined ab initio for these calculi, even when the bisimilarities agree. One phenomenon, seen here for Petri nets, is that the derived transitions have redundancy. This is because we derive transitions for each reaction rule separately; no advantage is gained from treating a whole rule-set. An interesting future study would be to somehow detect and eliminate redundancies, arriving at simpler transition systems.

We have discussed a way of deriving a non-trivial transitional theories for graphical models of mobile systems, and this has served to calibrate such a model against process calculi. But for many applications it will be important to look beyond theories of an algebraic character, and pursue the kind of spatio-temporal logic proposed by Cardelli and Caires [8, 9]. Such logics admit a partial —rather than holistic— analysis of complex systems, and they also lend themselves to powerful mechanical assistance (model-checking). The present work will then be useful in studying the extent to which the algebraic and logical theories agree.

References

[1] Baldan, P., Corradini, A., Ehrig, H. and Heckel, R. (2001), Compositional modeling of reactive systems using open nets. Proc. CONCUR 2001, 12th International Conference on Concurrency theory, LNCS 2154, Springer-Verlag, pp502–518.

[2] Bergstra, J.A. and Klop, J.W. (1985), Algebra for communicating processes with abstraction. Theoretical Computer Science 37, pp77–121.

[3] Berry, G. and Boudol, G. (1992), The chemical abstract machine. Journal of Theoretical Computer Science, Vol 96, pp217–248.

[4] Best, E., Devillers, R. and Hall, J.G. (1999), The box algebra: a model of nets and process expressions. 20th International Conference on Application and Theory of Petri Nets, LNCS 1639, Springer-Verlag, pp344–363.

[5] Boudol, G. (1992), Asynchrony and the π-calculus. Rapport de Recherche RR–1702, INRIA Sophia Antipolis.

[6] Brookes, S.D., Hoare, C.A.R. and Roscoe, A.W. (1984), A theory of communicating sequential processes. J. ACM 31, pp560–599.

[7] Cardelli, L. (2003) Bioware languages. In Computer Systems: Papers for Roger Needham, Springer Monographs in Computer Science, to appear, 6pp.

[8] Caires, L. and Cardelli, L. (2001), A spatial logic for concurrency (Part I). Proc. 4th International Symposium on Theoretical Aspects of Computer Software, LNCS 2215, Springer Verlag, pp1–37.

[9] Caires, L. (2004), Behavioural and spatial observations in a logic for the π-calculus. Proc. FOSSACS 2004, LNCS 2987, pp72–89.

[10] Cardelli, L. and Gordon, A.D. (2000), Mobile ambients. Foundations of System Specification and Computational Structures, LNCS 1378, pp140–155.

[11] Castellani, I. (2001), Process algebras with localities. *Handbook of Process Algebra*, eds Bergstra, Ponse and Smolka, Elsevier, pp947–1045.

[12] Cattani, G.L., Leifer, J.J. and Milner, R. (2000), Contexts and Embeddings for closed shallow action graphs. University of Cambridge Computer Laboratory, Technical Report 496. [Submitted for publication.] Available at `http://pauillac.inria.fr/~leifer`.

[13] Drewes, F., Hoffmann, B. and Plump, D. (2000) Hierarchical graph transformation. In: Foundations of Software Science and Computation Structures, LNCS 1784, Springer Verlag.

[14] Ehrig, H. (1979) Introduction to the theory of graph grammars. Graph Grammars and their Application to Computer Science and Biology, LNCS 73, Springer Verlag, pp1–69.

[15] Ehrig, H. (2002) Bigraphs meet double pushouts. EATCS Bulletin 78, October 2002, pp72–85.

[16] Ehrig, H. and König, B. (2004), Deriving bisimulation congruences in the DPO approach to graph-rewriting. Proc. FOSSACS 2004, LNCS 2987, pp151–156.

[17] Fournet, C. and Gonthier, G. (1996), The reflexive Cham and the join calculus. Proc. 23rd Annual ACM Symposium on Principles of Programming Languages, Florida, pp372–385.

[18] Gadducci, F., Heckel, R. and Llabrés Segura, M. (1999), A bi-categorical axiomatisation of concurrent graph rewriting. Proc. 8th Conference on Category Theory in Computer Science (CTCS'99), Vol 29 of Electronic Notes in TCS, Elsevier Science.

[19] Gardner, P.A. (2000), From process calculi to process frameworks. Proc. CONCUR 2000, 11th International Conference on Concurrency Theory, pp69–88.

[20] Gardner, P.A. and Wischik, L. (2000), Explicit fusions. Proc. MFCS 2000. LNCS 1893, pp373–383.

[21] Hasegawa, M. (1999), Models of sharing graphs (A categorical semantics of Let and Letrec). PhD Dissertation, Report ECS-LFCS-97-360, Division of Informatics, Edinburgh University. Springer Series of Distinguished Dissertations in Computer Science.

[22] Hoare. C.A.R. (1985), *Communicating Sequential Processes*. Prentice Hall.

[23] Honda, K. and Tokoro, M. (1991). An object calculus for asynchronous communications. In ECOOP'91, Workshop on Object-based Concurrent Programming, LNCS 512.

[24] Jensen, O.H, (2004), Forthcoming PhD Dissertation.

[25] Jensen, O.H. and Milner, R. (2003), Bigraphs and transitions. In 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages.

[26] Jensen, O.H. and Milner, R. (2004), Bigraphs and mobile processes (revised). Technical Report 580, University of Cambridge Computer Laboratory. Available from `http://www.cl.cam.ac.uk/users/rm135`.

[27] Lafont, Y. (1990) Interaction nets. Proc. 17th ACM Symposium on Principles of Programming Languages (POPL'90), pp95–108.

[28] Lawvere, F.W. (1963), Functorial semantics of algebraic theories. Dissertation, Columbia University. Announcement in Proc. Nat. Acad. Sci. 50, 1963, pp869–873.

[29] Leifer, J.J. (2001), Operational congruences for reactive systems. PhD Dissertation, University of Cambridge Computer Laboratory. Distributed in revised form as Technical Report 521. Available from `http://pauillac.inria.fr/~leifer`.

[30] Leifer, J.J. and Milner, R. (2000), Deriving bisimulation congruences for reactive systems. Proc. CONCUR 2000, 11th International Conference on Concurrency theory, pp243–258. Available at `http://pauillac.inria.fr/~leifer`.

[31] Merro, M. and Hennessy, M. (2002), Bisimulation Congruences in Safe Ambients. Proc. 29th International Symposium on Principles of Programming Languages, Oregon, pp71–80.

[32] Milner, R. (1980) *A Calculus of Communicating Systems*. LNCS 92, Springer Verlag.

[33] Milner, R. (1996), Calculi for interaction. Acta Informatica 33, pp707–737.

[34] Milner, R. (2004), Bigraphs for Petri nets. In Proc. Advanced Course on Petri Nets, Eichstätt 2003, LNCS 3098.

[35] Milner, R. (2004), Axioms for bigraphical structure. Technical Report 581, University of Cambridge Computer Laboratory. (Submitted for publication.)

[36] Milner, R. (2001) Computational flux. Proc 28th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, pp220-221.

[37] Milner, R. Bigraphical reactive systems: basic theory. Technical Report 503, University of Cambridge Computer Laboratory (2001). Available from `http://www.cl.cam.ac.uk/users/rm135`.

[38] Milner, R. (2001) Bigraphical reactive systems. Proc. 12th International Conference on Concurrency Theory, LNCS2154, pp16–35.

[39] Milner, R., Parrow, J. and Walker D. (1992), A calculus of mobile processes, Parts I and II. Journal of Information and Computation, Vol 100, pp1–40 and pp41–77.

[40] Nielsen, M., Priese, L. and Sassone, V., Characterizing behavioural congruences for Petri nets. Proc. CONCUR'95, LNCS 962 (1995) 175–189.

[41] Park, D.M.R. (1980), Concurrency and automata on infinite sequences. LNCS 104, Springer Verlag.

[42] Pomello, L., Rozenberg, G. and Simone, C., A survey of equivalence notions for net-based systems. Advances in Petri Nets '92, LNCS 609 (1992) 410–472.

[43] Priese, L. and Wimmel, H., A uniform approach to true-concurrency and interleaving semantics for Petri nets. Theoretical Computer Science 206 (1998) 219–206.

[44] Parrow, J. and Victor, B. (1998), The fusion calculus: expressiveness and symmetry in mobile processes. Proc. LICS'98, IEEE Computer Society Press.

[45] Priami, C. (1995), Stochastic π-calculus. Computer Journal, Vol 38 (7), pp578–589.

[46] Priami, C., Regev, A., Silverman, W. and Shapiro, E. (2001), Application of stochastic process algebras to bioinformatics of molecular processes. Information Processing Letters, Vol 80, pp 25–31.

[47] Rounds, W.H. and Song, H. (2003), The Φ-calculus - a language for distributed control of reconfigurable embedded systems. In Hybrid Systems: Computation and control, LNCS 2263, Springer-Verlag, pp435–449.

[48] Sassone, V. and Sobocinski, P. (2002), Deriving bisimulation congruences: a 2-categorical approach. Electronic Notes in Theoretical Computer Science, Vol 68 (2).

[49] Sewell, P. (1998), From rewrite rules to bisimulation congruences. Proc CONCUR'98, LNCS 1466, pp269–284. Full version to appear in Theoretical Computer Science, Vol 272/1–2.

[50] Regev, A., Silverman, W. and Shapiro, E. (2001), Representation and simulation of biochemical processes using the π-calculus process algebra. Proc. Pacific Symposium of Biocomputing 2001 (PSB2001), Vol 6, pp459–470.

[51] Smith, H. and Fingar, P. (2002), Business Process Management: the third wave. Amazon.com.

[52] Wojciechowski, P.T. and Sewell, P. (1999), Nomadic Pict: Language and infrastructure design for mobile agents. Proc. ASA/MA '99, Palm Springs, California.

Appendix

A Proofs

Lemma 6.12 *$(\vec{B}, B)$ is a candidate RPO for $\vec{A}$ relative to $\vec{D}$.*

Proof To prove $B_0 \circ A_0 = B_1 \circ A_1$, by symmetry it will be enough to consider cases for $p \in W \uplus P_0$, and for the value of $A_0(p)$.

Case $p \in P_0 - P_2$, $A_0(p) = e \in E_0$. Then $(B_1 \circ A_1)(p) = B_1(p) = D_1(p) = (D_1 \circ A_1)(p) = (D_0 \circ A_0)(p) = A_0(p) = (B_0 \circ A_0)(p)$.

Case $p \in P_0 - P_2$, $A_0(p) = x_0 \in X_0$. Then $(B_1 \circ A_1)(p) = B_1(p) = \widehat{x_0} = B_0(x_0) = (B_0 \circ A_0)(p)$.

Case $p \in W \uplus P_2$, $A_0(p) = e \in E_0 - E_2$. Then $(B_0 \circ A_0)(p) = A_0(p) = e$. Also $(D_1 \circ A_1)(p) = (D_0 \circ A_0)(p) = e$, so for some $x_1 \in X_1$ we have $A_1(p) = x_1$ and $D_1(x_1) = e$, hence $x_1 \notin X_1'$. Then $(B_1 \circ A_1)(p) = B_1(x_1) = D_1(x_1) = e$.

Case $p \in W \uplus P_2$, $A_0(p) = e \in E_2$. Then $(D_1 \circ A_1)(p) = (D_0 \circ A_0)(p) = e$, so also $A_1(p) = e$. Hence $(B_1 \circ A_1)(p) = e = (B_0 \circ A_0)(p)$.

Case $p \in W \uplus P_2$, $A_0(p) = x_0 \in X_0'$. Then $D_0(x_0) \in E_3 \uplus Z$, and so $(D_1 \circ A_1)(p) = (D_0 \circ A_0)(p) \in E_3 \uplus Z$; hence for some $x_1 \in X_1'$ we have $A_1(p) = x_1$ and $D_1(x_1) = D_0(x_0)$. Hence $(B_0 \circ A_0)(p) = B_0(x_0) = D_0(x_0) = D_1(x_1) = B_1(x_1) = (B_1 \circ A_1)(p)$.

Case $p \in W \uplus P_2$, $A_0(p) = x_0 \in X_0 - X_0'$. Then $D_0(x_0) = e \in E_1 - E_2$; hence $(D_1 \circ A_1)(p) = (D_0 \circ A_0)(p) = e$, so $A_1(p) = e$. So $(B_1 \circ A_1)(p) = e = D_0(r_0) = B_0(x_0) = (B_0 \circ A_0)(p)$.

We now prove $B \circ B_0 = D_0$ by case analysis.

Case $x \in X_0'$. Then $(B \circ B_0)(x) = B(\widehat{0, x}) = D_0(x)$.

Case $x \in X_0 - X_0'$. Then $B_0(x) = D_0(x) \in E_0 - E_2$, hence $(B \circ B_0)(x) = D_0(x)$.

Case $p \in P_1 - P_2$, $D_0(p) \in E_1 - E_2$. Since $D_0 \circ A_0 = D_1 \circ A_1$ we have $A_1(p) \notin X_1$, so $B_0(p) = D_0(p) \in E_1 - E_2$; hence $(B \circ B_0)(p) = B_0(p) = D_0(p)$.

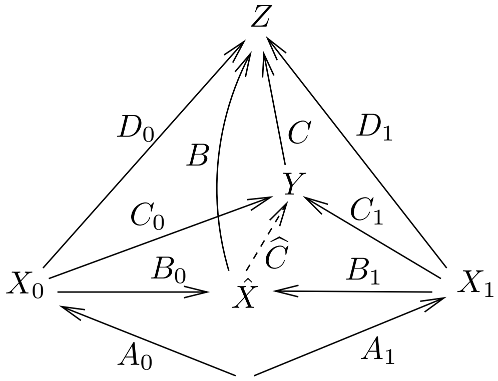
Case $p \in P_1 - P_2$, $D_0(p) \in E_3 \uplus Z$. Since $D_0 \circ A_0 = D_1 \circ A_1$ there exists $x \in X_1$ with $A_1(p) = x$; moreover we readily deduce $x \in X_1'$, so $B_0(p) = \widehat{1, x}$. Hence $(B \circ B_0)(p) = B(\widehat{1, x}) = D_1(x) = (D_1 \circ A_1)(p) = (D_0 \circ A_0)(p) = D_0(p)$.

Case $p \in P_3$. Then $(B \circ B_0)(p) = B(p) = D_0(p)$. ■

Theorem 6.13 (RPOs in link graphs) *In ´LIG, Whenever a pair $\vec{A}$ of link graphs has a bound $\vec{D}$, there exists an RPO $(\vec{B}, B)$ for $\vec{B}$ relative to $\vec{D}$, and Construction 6.10 yields such an RPO.*

Proof We have already proved that the triple $(\vec{B}, B)$ built in Construction 6.10 is an RPO candidate. Now consider any other candidate $(\vec{C}, C)$ with intervening interface

Y. C_i has nodes $V_{\bar{i}} - V_2 \uplus V_4$ ($i = 0, 1$) and C has nodes V_5, where $V_4 \uplus V_5 = V_3$. We have to construct a unique mediating arrow $\widehat{C}$, as shown in the diagram.



We define $\widehat{C}$ with nodes V_4 as follows:

$$\text{for } \hat{x} = \widehat{i,x} \in \hat{X} : \quad \widehat{C}(\hat{x}) \stackrel{\text{def}}{=} C_i(x)$$
$$\text{for } p \in P_4 : \quad \widehat{C}(p) \stackrel{\text{def}}{=} C_i(p) \ .$$

Note that the equations $\widehat{C} \circ B_i = C_i$ ($i = 0, 1$) determine $\widehat{C}$ uniquely, since they force the above definition. We now prove the equations (considering $i = 0$):

Case $x \in X_0'$. Then $(\widehat{C} \circ B_0)(x) = \widehat{C}(\widehat{0,x}) = C_0(x)$.

Case $x \in X_0 - X_0'$. Then $D_0(x) \in E_1 - E_2$, so $B_0(x) = D_0(x)$, hence $(\widehat{C} \circ B_0)(x) = D_0(x)$. Also since $C \circ C_0 = D_0 \in E_1 - E_2$ we have $C_0(x) = D_0(x)$.

Case $p \in P_1 - P_2$, $D_0(p) \in E_1 - E_2$. Since $D_0 \circ A_0 = D_1 \circ A_1$ we have $A_1(p) \notin X_1$, so $B_0(p) = D_0(p)$, hence $(\widehat{C} \circ B_0)(p) = D_0(p)$. Also $C_0(p) = (C \circ C_0)(p) = D_0(p)$.

Case $p \in P_1 - P_2$, $D_0(p) \in E_3 \uplus Z$. Then $A_1(v) = x \in X_1'$ with $D_1(x) = D_0(p)$, and $B_0(p) = \widehat{1,x}$. So $(\widehat{C} \circ B_0)(p) = \widehat{C}(\widehat{1,x}) = C_1(x) = (C_0 \circ A_0)(p) = C_0(p)$.

Case $p \in P_4$. Then $(\widehat{C} \circ B_0)(p) = \widehat{C}(p) = C_0(p)$.

It remains to prove that $C \circ \widehat{C} = B$. The following cases suffice:

Case $\hat{x} = \widehat{0,x} \in X$, $B(\hat{x}) \in E_4$. Then $(C \circ \widehat{C})(\hat{x}) = \widehat{C}(\hat{x}) = C_0(x) = D_0(x) = B(\hat{x})$.

Case $\hat{x} = \widehat{0,x} \in X$, $B(\hat{x}) \in E_5 \uplus Z$. Then $D_0(x) = B(\hat{x}) \in E_5 \uplus Z$, so for some $y \in Y$ we have $C_0(x) = y$ and $C(y) = B(\hat{x})$. But by definition $\widehat{C}(\hat{x}) = y$, so $(C \circ \widehat{C})(\hat{x}) = C(y) = (C \circ C_0)(x) = D_0(x) = B(\hat{x})$.

Case $p \in P_4$, $B(v) \in E_4$. Then $(C \circ \widehat{C})(p) = \widehat{C}(p) = C_0(p) = D_0(p) = B(p)$.

Case $p \in P_4$, $B(p) \in E_5 \uplus Z$. Then $B(p) = D_0(p) = C(y)$, where $C_0(p) = y \in Y$, and by definition $\widehat{C}(p) = C_0(p)$, so $(C \circ \widehat{C})(p) = C(y) = B(p)$.

Case $p \in P_5$. Then $(C \circ \widehat{C})(p) = C(p) = D_0(p) = B(p)$.

Hence $\widehat{C}$ is the required unique mediator; so $(\vec{B}, B)$ is an RPO. ∎