Math. Struct. in Comp. Science: *page* 1 of 59.
doi:10.1017/S0960129506005664

Transition systems, link graphs and Petri nets

JAMES J. LEIFER[†] and ROBIN MILNER[‡]

[†]*INRIA, Domaine de Voluceau, BP 105, 78153 Le Chesnay Cedex, France*
Email: `James.Leifer@inria.fr`

[‡]*University of Cambridge, Computer Laboratory, JJ Thomson Avenue, Cambridge CB3 0FD, U.K.*

Received 17 October 2004; revised 2 May 2006

A framework is defined within which reactive systems can be studied formally. The framework is based on *s-categories*, which are 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 on the notion of *relative pushout*, which was 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. Introduction

1.1. *Background*

Process calculi have made progress in modelling interactive concurrent systems (Brookes *et al*. 1984; Bergstra and Klop 1985; Hoare 1985; Milner 1980), systems with mobile connectivity (Milner *et al*. 1992; Fournet and Gonthier 1996), and systems with mobile locality (Berry and Boudol 1992; Cardelli and Gordon 2000). There is some agreement amongst 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 the various kinds of mobility that they model. As shown by Castellani's comprehensive survey, Castellani (2001), widely varying notions of locality have been explored, and this implies a similar variety in treating mobility.

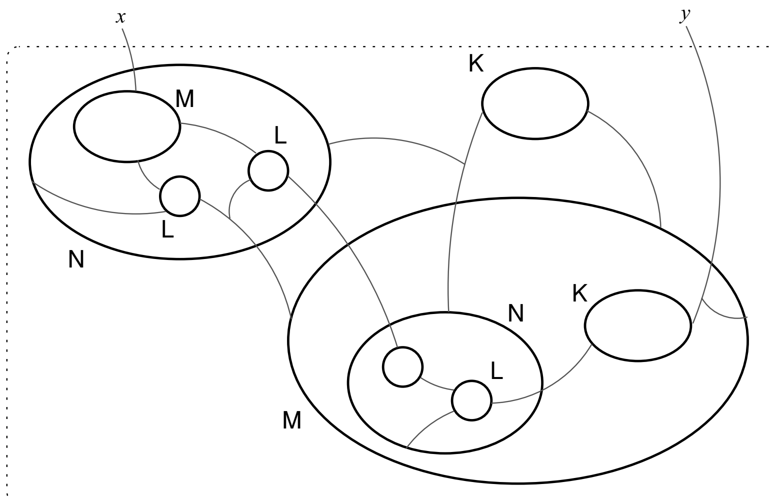


Fig. 1. An example of a bigraph

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 they were, but it appears that mobility is becoming essential to a huge range of applications, so the search for a common theoretical basis should attend to mobility at the outset if it is not to risk irrelevancy.

The authors' response (Leifer and Milner 2000; Leifer 2001) 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 (Milner 2001b; Jensen and Milner 2003) 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 3.1.

These twin proposals have been combined in applications to the π-calculus (Jensen and Milner 2003; Jensen and Milner 2004), the ambient calculus (Jensen and Milner 2004; Jensen 2006) and Petri nets (Milner 2004a), yielding behavioural theory agreeing well with those proposed independently. The theory developed so far 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 (Milner 2004a) 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.

In modelling process calculi, place graphs are useful for many purposes, including sequential control (guarding), scoping of names, and replication. Application to the finite

π-calculus was outlined in Jensen and Milner (2003), which showed there to be a fair correspondence with the transitions and bisimilarity that were originally defined for the calculus. This correspondence has recently been examined in greater detail for finite CCS (Milner 2005), where agreement with the original bisimilarity is found to be exact. In his forthcoming Ph.D. Dissertation (Jensen 2006), Jensen extends these treatments to the full π-calculus and to mobile ambients.

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 on 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 5 we compare this with two alternative approaches: one using a category of graph embeddings and the other a 2-category.

1.2. *Synopsis*

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

Section 2 begins with an overview 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 on IPOs are then derived uniformly from these rules, using RPOs. It is proved that, when enough RPOs exist, bisimilarity is a congruence. Section 2 concludes with a study of how a reactive system may be equipped with different transition systems, and how these may be related to one another.

Section 3 begins with an overview 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 Section 2 is then 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.

Section 4 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 (first reported in Milner (2004a)), for which the work of Section 3 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 5, discusses related and future research.

1.3. *Correction*

We should like to take this opportunity to correct an error in the paper 'Axioms for bigraphical structure' (Milner 2004b). In Table 1 of the paper the following axiom was mistakenly omitted and should be added at the end of the list of categorical axioms:

$$\gamma_{I\otimes J,K} = (\gamma_{I,K} \otimes \mathsf{id}_J)(\mathsf{id}_I \otimes \gamma_{J,K}) \ .$$

2. Reactive systems and transition systems

2.1. *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 $a \longrightarrow a'$, where a and a' are agents. This treatment is often accompanied by *labelled transitions* of the form $a \xrightarrow{\ell} a'$, 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 $\xrightarrow{\tau}$ coincides with the reaction relation $\longrightarrow$.

Labelled transitions express the interaction between a process and its environment. They lead naturally to the definition of behavioural preorders, such as traces, failures and bisimilarity, which often turn out to be congruences. But hitherto the labelled transitions have been tailored individually for each calculus.

We therefore ask whether these labels can be *derived* uniformly from any given set of reaction rules of the form $r \longrightarrow r'$, 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 $a \xrightarrow{L} a'$ 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, that is, preserved by insertion into any surrounding environment.

But we do not want *all* contexts as labels; as Sewell (1998) points out, the behavioural equivalences that result from this choice are unsatisfactory. The problem of finding a satisfactory, and suitably minimal, set of labels, and to do so uniformly, remained an open problem for many years. As a first step, Sewell was able to derive context-labelled transitions uniformly for parametric term-rewriting systems with parallel composition and blocking, and to show that bisimilarity is a congruence (Sewell 1998). His approach did not handle reactive systems with 'connectivity', represented by the (potentially non-linear) sharing of names that arises in many process calculi.

Recently, the current authors 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 (Leifer and Milner 2000). These results were extended and refined in the first author's Ph.D. Dissertation (Leifer 2001), and the theory was also applied to action graphs with rich connectivity in Cattani *et al.* (2002). Meanwhile, the second author developed the bigraph model from action graphs (Milner 2001a; 2001b), 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 the development of *symmetric* action graphs (that is, those with undirected edges) in Gardner (2000).

These applications have motivated the effort to formulate the RPO theory more succinctly (Jensen and Milner 2004), and 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 Section 2 of our paper. It turns out that these two tasks can be addressed well using a variant of a 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, for arrows of suitable domain and codomain, if and only if $|g| \cap |f| = \emptyset$. This structure makes s-categories remarkably well-behaved. They inherit many notions from categories unchanged, and most work is unaffected by the partiality of composition. They also admit direct treatment of the notion of *occurrence* (for example, of a node in a graph), which in Section 3 we find essential to the characterisation of behaviour.

In Section 2.2 we introduce our categorical framework. We then define RPOs and IPOs, and derive their properties. This leads in Section 2.3 to *reactive systems*, and thence to the derivation of *transition systems* based on IPOs. The central theorem, that bisimilarity for these transition systems is a congruence provided enough RPOs exist, is then proved. The remainder of the section deals with useful relationships between transition systems in preparation for Section 3, where we need to refine them by varying their agents or transitions, or both.

2.2. *S-categories and relative pushouts*

In this section and Section 2.3 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 (Milner 2001b; Jensen and Milner 2003) and its applications. More specifically, to keep the paper well-focussed, the abstract development is only taken as far as we need for link graphs, which are constituents of bigraphs. These two sections are an adaptation and extension of work started by the authors (Leifer and Milner 2000), then further developed by the first author in his PhD Dissertation (Leifer 2001) and by the second author in Milner (2001b).

Sections 3 and 4 can be read independently of Section 2 provided the main results of Section 2 are taken on trust and one is prepared 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 2.3.

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 use $\mathsf{dom}(f)$ and $\mathsf{cod}(f)$ to denote the domain I and codomain J of an arrow $f : I \to J$, and we use $\acute{\mathbf{C}}(I \to J)$, or just $I \to J$, to denote the set of arrows from I to J, which is called a *homset*.

We use Id_S to denote the identity function on a set S, and $\varnothing_S$ to denote the empty function from $\varnothing$ to S. We shall use $S \uplus T$ for the union of sets S and T that are known or assumed to be disjoint, and $f \uplus g$ for the 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, we define $R^{\equiv}$ to be the closure of R under $\equiv$, that is, 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 2.1 (precategory, functor). A *precategory* $\acute{\mathbf{C}}$ is defined in exactly the same way 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 it is defined in $\acute{\mathbf{C}}$. A *functor* $\mathscr{F} : \acute{\mathbf{D}} \to \acute{\mathbf{C}}$ 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 $\mathscr{F}(g) \circ \mathscr{F}(f) = \mathscr{F}(g \circ f)$ 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 2.2 (tensor product, monoidal precategory, monoidal functor). A *(strict, symmetric) monoidal* precategory has a partial *tensor product* $\otimes$ on both objects and 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 $f : H \to I,\ g : J \to K$)
(6) $\gamma_{I \otimes J,K} = (\gamma_{I,K} \otimes \mathsf{id}_J) \circ (\mathsf{id}_I \otimes \gamma_{J,K})$.

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 begin by developing the theory for *concrete* bigraphs (those in which nodes have identity), and 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, since 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 *keeping track* of nodes;[†] 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 on a *support* set associated with each arrow. In bigraphs the support of an arrow will be its node set. In general we assume the support to be drawn from some unspecified infinite set. We now give a general definition of precategories $\acute{\mathbf{C}}$ with support; we will continue to use this accented notation for them, dropping the accent only when we have a category.

Definition 2.3 ((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 $\mathsf{dom}(g) = \mathsf{cod}(f)$, so $|g \circ f| = |g| \uplus |f|$.

— 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:

(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) $|\rho \bullet f| = \rho(|f|)$.

If $\acute{\mathbf{C}}$ is monoidal as a precategory, it is a *monoidal* s-category provided the following conditions hold: for $f : H \to I$ and $g : J \to K$, 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.

[†] The first author's development (Leifer 2001, 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.

Exercise. 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 2.4 (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 $f \doteq g$, if $\rho \bullet f = g$ for some support translation ρ. By Definition 2.3, this is an equivalence relation. If $\acute{\mathbf{B}}$ is another s-category, then we say a functor $\mathscr{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ is *supported* if it preserves support equivalence, that is, $f \doteq g$ implies $\mathscr{F}(f) \doteq \mathscr{F}(g)$.

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:[†]

Definition 2.5 (static congruence). Let $\equiv$ be an equivalence defined on every homset of an s-category $\acute{\mathbf{C}}$. We say $\equiv$ is 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 3.2 for the definitions of controls and link graphs.)

The most important example of a static congruence will be support equivalence ($\doteq$), but the following definition shows that any static congruence at least as coarse as support equivalence will yield a well-defined quotient category.

Definition 2.6 (quotient categories). Let $\acute{\mathbf{C}}$ be an s-category and $\equiv$ be a static (monoidal) congruence on $\acute{\mathbf{C}}$ that includes support equivalence, that is, $\doteq \subseteq \equiv$. Then the *quotient* of $\acute{\mathbf{C}}$ by $\equiv$ is a category $\mathbf{C} \stackrel{\text{def}}{=} \acute{\mathbf{C}}/\equiv$ whose objects are the objects of $\acute{\mathbf{C}}$ and whose arrows are equivalence classes of arrows in $\acute{\mathbf{C}}$:

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

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

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

Note that in the last two equations, the right-hand sides are only defined when f and g have disjoint supports. However, this does not adversely affect composition and tensor product in $\mathbf{C}$: as the equivalence classes on the left-hand side are closed under support equivalence, it is always possible to find candidates f and g whose supports are disjoint.

† We use the term *static* congruence to emphasise the fact that these congruences depend only on static structure, in contrast with *dynamic* congruences such as bisimilarity, which depend on transitions.

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 $\acute{\mathbf{C}}$.

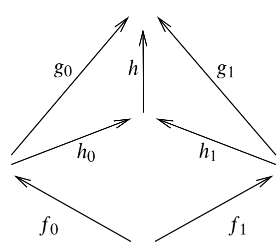
Note that the quotient does not affect objects; thus any tensor product on $\mathbf{C}$ may still be partial on objects. But $\mathbf{C}$ is indeed a category; composition is always well-defined because $f \doteq g$ implies $f \equiv g$, and so also is tensor product provided it is defined on the domains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

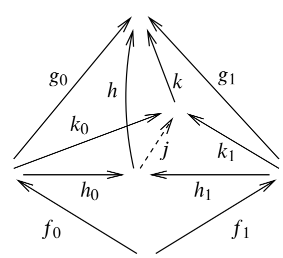
We often abbreviate $[\cdot]_{\doteq}$ to $[\cdot]$; we call it the *support quotient functor*. From the definition, it is clear that a coarser quotient $[\cdot]_{\equiv}$ exists whenever $\equiv$ is a congruence that includes support equivalence. In Section 3 we shall define a coarser quotient functor by this means.

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

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 2.7 (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, it is said to be *consistent*.





Definition 2.8 (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 this 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 that 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 3.2 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 $\vec{g}$ and ask 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 2.9 (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 the first author's dissertation (Leifer 2001) (see also Leifer and Milner (2000)), and the proofs can be routinely adapted for precategories.†

Proposition 2.10 (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

$$\begin{tikzcd} \cdot \arrow[r, "h_0"] & \cdot \arrow[r, "h_1"] & \cdot \\ \cdot \arrow[u, "f_0"] \arrow[r, "g_0"'] & \cdot \arrow[u, "f_1"] \arrow[r, "g_1"'] & \cdot \arrow[u, "f_2"'] \end{tikzcd}$$

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 $\rho \bullet \vec{g}$ is an IPO for $\rho \bullet \vec{f}$.

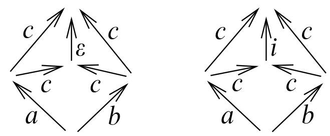
We now consider a property of RPOs that may not be present in all precategories, but will be enjoyed by link graphs. We know that the RPO status of a triple is preserved by isomorphism at its mediating interface, that is, if $(\vec{h}, h)$ is an RPO, then so is $(i \circ \vec{h}, h \circ j)$ where i, 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.

Definition 2.11 (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 $k = h$.

† This adaptation works for the definition of precategory in Definition 2.1, which is satisfied by our s-categories.

Exercise. (Not needed for what follows.) Prove that if $\vec{f}$ has a rigid RPO relative to $\vec{g}$, then all its RPOs relative to $\vec{g}$ are rigid.

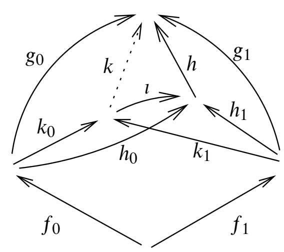
Show that not all RPOs are rigid, as follows. Work in a one-object category whose arrows are strings over the alphabet $\{a, b, c, i\}$ subject to the three equations $ca = cb$, $ic = c$ and $ii = \varepsilon$. The identity is ε, the empty string, and the composition $s \circ t$ of two arrows is the concatenation st. Consider the following diagrams:



Prove that both the diagrams are RPOs, and hence that neither is rigid.

In Section 3.2 we shall show that every link graph RPO is rigid. This is useful as we can then deduce from the following proposition that, in link graphs, a unique IPO is a pushout.

Proposition 2.12 (unique IPOs are pushouts). Assume that whenever $\vec{f}$ has a bound $\vec{g}$ it also has a rigid RPO relative to $\vec{g}$. Then, if $\vec{f}$ has a unique IPO up to isomorphism, this IPO is a pushout.



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 2.10(2); hence by assumption there is an isomorphism $\imath$ as shown such that $\imath \circ k_i = h_i$ $(i = 0, 1)$. Then $h \circ \imath$ satisfies the required property of the mediator k.

Now let k be any such mediator, and let $\imath'$ be the inverse of $\imath$. Then $(k \circ \imath') \circ h_i = k \circ \imath' \circ \imath \circ k_i = k \circ k_i = g_i$ $(i = 0, 1)$. It follows from Proposition 2.10(3) that $(\vec{h}, k \circ \imath')$ is an RPO for $\vec{f}$ to $\vec{g}$. But $(\vec{h}, h)$ is rigid by assumption, hence $k \circ \imath' = h$. So, finally, $k = h \circ \imath$, which shows that the mediator $h \circ \imath$ is unique, as required. □

2.3. *Reactive and transition systems*

We now introduce a kind of dynamical system, of which link graphs will be an instance. In previous work (Leifer and Milner 2000; Leifer 2001) a notion of reactive system was defined. In the terms of the current paper 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 our earlier work (Leifer and Milner 2000; Leifer 2001) 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 on IPOs. Several behavioural preorders and equivalences based on these transitions, including bisimilarity, were shown to be congruences, subject to two conditions: first, that sufficient RPOs exist in the s-category, and second, that decomposition preserves activity – that is, $D \circ C$ active implies both C and D active. In subsequent work, sufficient RPOs were found in interesting cases (Leifer 2001; Cattani *et al.* 2002)).

The present section is essentially a reformulation of Leifer and Milner (2000) and Leifer (2001). 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 2.14 (reactive system). A *reactive system (RS)* is a monoidal s-category $\acute{\mathbf{A}}$ equipped with a set $\acute{\mathscr{R}}$ of *reaction rules* of the form $(r : I, r' : I)$, in which r is the *redex* and r' the *reactum*. We require $\acute{\mathscr{R}}$ to be closed under support equivalence, that is, if (r, r') is a rule, then so is (s, s') whenever $r \bumpeq s$ and $r' \bumpeq s'$.

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

We use $\acute{\mathbf{A}}(\acute{\mathscr{R}})$ to denote this RS, or just $\acute{\mathbf{A}}$ when $\acute{\mathscr{R}}$ is understood. Closing the reaction rules under support equivalence allows us in Definition 2.17 to divide $\acute{\mathbf{A}}$ by $\bumpeq$, forming a quotient RS.

To close $\acute{\mathscr{R}}$ 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, as shown in Appendix C) 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 so that bigraphs capture the standard behavioural equivalences in process calculi, where there is no notion of tracking the identity of components through reaction.

When considering the closure condition, an alternative merits close attention. This 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 (2001), 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 $\mathscr{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ to RSs, requiring it to preserve reaction. Recall from Definition 2.4 that a supported functor is one that preserves support equivalence.

Definition 2.15 (RS functor, sub-RS). A supported monoidal functor $\mathscr{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ of monoidal s-categories is an *RS functor* if it preserves reaction rules, that is, if (r, r') is a rule of $\acute{\mathbf{A}}$, then $(\mathscr{F}(r), \mathscr{F}(r'))$ is a rule of $\acute{\mathbf{B}}$. If $\mathscr{F}$ is injective on objects and arrows, we say $\acute{\mathbf{A}}$ is a *sub-RS* of $\acute{\mathbf{B}}$.

Proposition 2.16 (RS functors preserve reaction). An RS functor $\mathscr{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ preserves reaction, that is, if $g \rightarrowtriangle g'$ in $\acute{\mathbf{A}}$, then $\mathscr{F}(g) \rightarrowtriangle \mathscr{F}(g')$ in $\acute{\mathbf{B}}$.

It is clear that RSs and their functors form a category. An important example of a functor is the support quotient functor extended to RSs as follows.

Definition 2.17 (quotient RS). Let $\acute{\mathbf{A}}$ be a reactive system equipped with $\acute{\mathscr{R}}$. Then its *support quotient* reactive system is based on the support quotient $\mathbf{A} = \acute{\mathbf{A}}/\bumpeq$. Its reaction rules are $\{([r], [r']) \mid (r, r') \in \acute{\mathscr{R}}\}$.

Proposition 2.18 (quotient reflects reaction). The support quotient functor $[\cdot] : \acute{\mathbf{A}} \to \mathbf{A}$ both preserves and reflects reaction, that is, $[g] \rightarrowtriangle [g']$ in $\mathbf{A}$ iff $g \rightarrowtriangle g'$ in $\acute{\mathbf{A}}$.

The quotient functor takes a *concrete* RS based on an s-category to an *abstract* RS based on a category. Later we will 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 the first author’s Dissertation (Leifer 2001).

We now consider how to equip an RS with labelled transitions. Conventionally, a labelled transition takes the form $a \overset{\ell}{\rightarrowtriangle} a'$, 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 (Leifer and Milner 2000) defined a labelled transition to be a triple written $a \overset{L}{\rightarrowtriangle} a'$ 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 or impose an activeness condition:

$$\begin{tikzcd} {} \arrow[r, "L"] & {} \\ {} \arrow[u, "a"] \arrow[r, "r"] & {} \arrow[u, "D"'] \end{tikzcd}$$

Definition 2.19 (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 $a' \doteq D \circ r'$. 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 on them. This leads to the following definition.

Definition 2.20 (transition system). Given an RS $\acute{\mathbf{A}}$, a *(labelled) transition system* $\mathscr{L}$ for $\acute{\mathbf{A}}$ is a pair $(\mathsf{Int}_{\mathscr{L}}, \mathsf{Trans}_{\mathscr{L}})$, where

— $\mathsf{Int}_{\mathscr{L}}$ is a set of interfaces called the *agent interfaces*; the *agents* of $\mathscr{L}$ are defined as $\mathsf{Ag}_{\mathscr{L}} \overset{\text{def}}{=} \{a : I \mid I \in \mathsf{Int}_{\mathscr{L}}\}$.

— $\mathsf{Trans}_{\mathscr{L}}$ is a set of transitions $a \xrightarrow{L} a'$ such that a, a' are agents of $\mathscr{L}$; the *labels* of $\mathscr{L}$ are those that appear in some transition of $\mathsf{Trans}_{\mathscr{L}}$.

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

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

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 in the conditions under which these behavioural relations are congruential, that is, preserved by context. Here we shall limit attention to strong bisimilarity. (Throughout the rest of this paper we shall omit 'strong', since we do not define or use weak bisimilarity.)

Definition 2.21 (bisimilarity, congruence). Let $\acute{\mathbf{A}}$ be a reactive system equipped with a TS $\mathscr{L}$. A *simulation on* $\mathscr{L}$ is a binary relation $\mathscr{S}$ between agents with equal interface such that if $a \mathscr{S} b$ and $a \xrightarrow{L} a'$ in $\mathscr{L}$, then, whenever $L \circ b$ is defined, there exists b' such that $b \xrightarrow{L} b'$ in $\mathscr{L}$ and $a' \mathscr{S} b'$. A *bisimulation* is a symmetric simulation. Then *bisimilarity on* $\mathscr{L}$, denoted by $\sim_{\mathscr{L}}$, is the largest bisimulation on $\mathscr{L}$.

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

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

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

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

Note the slight departure from the usual definition of bisimulation in Park (1980); 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, 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, it is also preserved by tensor product; that is, if $a \sim_{\mathscr{L}} b$, then $a \otimes c \sim_{\mathscr{L}} b \otimes c$. To see this, note that $a \otimes c = (\mathsf{id} \otimes c) \circ a$.

Definition 2.22 (respect). Let $\equiv$ be a static congruence (Definition 2.5) in an RS equipped with $\mathscr{L}$. Suppose that for every transition $a \xrightarrow{L} a'$ in $\mathscr{L}$, if $a \equiv b$ and $L \equiv M$ for another label M of $\mathscr{L}$ with $M \circ b$ defined, then there exist an agent b' and a transition $b \xrightarrow{M} b'$ in $\mathscr{L}$ such that $a' \equiv b'$. We then say that $\equiv$ and $\mathscr{L}$ *respect* one another.

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

It is well known (Milner 1980) that if $\equiv$ is included in (strong) bisimilarity, then to establish bisimilarity it is enough to exhibit a *bisimulation up to* $\equiv$; that is, a symmetric relation $\mathscr{S}$ such that whenever $a \mathscr{S} b$, each transition of a is matched by b in $\mathscr{S}^{\equiv}$. We now deduce from Proposition 2.10(5) that support equivalence can be used in this way.

Proposition 2.23 (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 2.24 (congruence of bisimilarity). In a reactive system that has RPOs and is equipped with the standard transition system, bisimilarity of agents is a congruence; that is, if $a_0 \sim a_1$, then $C \circ a_0 \sim C \circ a_1$.

(a)

$$\begin{tikzcd} {} \arrow[r, "M"] & {} \\ {} \arrow[u, "C \circ a_0"] \arrow[r, "r_0"'] & {} \arrow[u, "E_0"'] \end{tikzcd}$$

(b)

$$\begin{tikzcd} {} \arrow[r, "M"] & {} \\ {} \arrow[u, "C"] \arrow[r, "L"] & {} \arrow[u, "E"'] \\ {} \arrow[u, "a_0"] \arrow[r, "r_0"'] & {} \arrow[u, "D_0"'] \arrow[uu, bend right, "E_0"'] \end{tikzcd}$$

(c)

$$\begin{tikzcd} {} \arrow[r, "L"] & {} \\ {} \arrow[u, "a_1"] \arrow[r, "r_1"'] & {} \arrow[u, "D_1"'] \end{tikzcd}$$

(d)

$$\begin{tikzcd} {} \arrow[r, "M"] & {} \\ {} \arrow[u, "C"] \arrow[r, "L"] & {} \arrow[u, "E"'] \\ {} \arrow[u, "a_1"] \arrow[r, "r_1"'] & {} \arrow[u, "D_1"'] \arrow[uu, bend right, "E_1"'] \end{tikzcd}$$

Proof. The proof is along the same lines as the proof of Theorem 3.9 in Leifer (2001). We establish the following as a bisimulation up to $\doteq$:

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

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 $(b'_0, b'_1) \in \mathscr{S}^{\doteq}$.

There exist a reaction rule (r_0, r'_0) and a context E_0 such that diagram (a) is an IPO; moreover, $b'_0 \doteq E_0 \circ r'_0$. Then, because consistent pairs have RPOs, there exists an RPO for (a_0, r_0) relative to the given bound, and using Proposition 2.10(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), to get diagram (d), in which, by Proposition 2.10(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, $(b'_0, b'_1) \in \mathscr{S}^{\doteq}$, as required, because $b'_0 \doteq E \circ a'_0$ and $b'_1 \doteq E \circ a'_1$ with $a'_0 \sim a'_1$. □

We should mention that we are taking (strong) bisimilarity as a representative of many preorders and equivalences; the first author has proved congruence theorems for several others (Leifer 2001), 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. To this end, 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 $\mathscr{F}(a \xrightarrow{L} a') = \mathscr{F}(a) \xrightarrow{\mathscr{F}(L)} \mathscr{F}(a')$. It is straightforward to check that this is a transition system in the target of $\mathscr{F}$.

Definition 2.25 (functors inducing and respecting transitions). Let $\mathscr{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ be an RS functor, and let $\acute{\mathbf{A}}$ be equipped with a TS $\mathscr{L}$. We say that $\mathscr{F}(\mathscr{L})$ is the TS *induced* on $\acute{\mathbf{B}}$ by $\mathscr{F}$. We say that $\mathscr{F}$ *respects* $\mathscr{L}$ if the static congruence it induces on $\acute{\mathbf{A}}$ respects $\mathscr{L}$.

This definition always makes sense, but it will not always make bisimilarity a congruence in $\acute{\mathbf{B}}$, even if it is one 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 2.26 (functors on bisimilarity). Let $\acute{\mathbf{A}}$ be equipped with a TS $\mathscr{L}$. Let $\mathscr{F}$ be a full RS functor from $\acute{\mathbf{A}}$ to $\acute{\mathbf{B}}$ that is the identity on objects and respects $\mathscr{L}$, and such that $a \doteq b$ implies $\mathscr{F}(a) = \mathscr{F}(b)$. Then the following hold for $\mathscr{F}(\mathscr{L})$:

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

Proof.

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

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

Let $a \sim_{\mathscr{L}} b$ in $\acute{\mathbf{A}}$, and let $p = \mathscr{F}(a)$, $q = \mathscr{F}(b)$ and $p \xrightarrow{M} p'$ in $\acute{\mathbf{B}}$, with $M \circ q$ defined. Then, by the definition of the induced TS, we can find L, a_0 and a_0' such that $M = \mathscr{F}(L)$, $p = \mathscr{F}(a_0)$ and $p' = \mathscr{F}(a_0')$, and $a_0 \xrightarrow{L} a_0'$ in $\acute{\mathbf{A}}$. Moreover, since $\mathscr{F}$ equates support-equivalent arrows, we may assume that L is chosen such that $L \circ b$ and $L \circ a$ are defined. Since $\mathscr{F}$ respects $\mathscr{L}$, there exists a' such that $a \xrightarrow{L} a'$ and $\mathscr{F}(a') = \mathscr{F}(a_0') = p'$. So for some b' we have $b \xrightarrow{L} b'$ with $a' \sim_{\mathscr{L}} b'$. It follows that $q \xrightarrow{M} q'$ in $\acute{\mathbf{B}}$, where $q' = \mathscr{F}(b')$ and $(p', q') \in \mathscr{R}$, so we are done.

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

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

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

2 Assume that $\sim_{\mathscr{L}}$ is a congruence. In $\acute{\mathbf{B}}$, let $p \sim_{\mathscr{F}(\mathscr{L})} q$ and G be a context such that $G \circ p$ and $G \circ q$ are defined and are agents with respect to $\mathscr{F}(\mathscr{L})$. Then, since $\mathscr{F}$ is full, there exist a, b, C in $\acute{\mathbf{A}}$ with $p = \mathscr{F}(a)$, $q = \mathscr{F}(b)$ and $G = \mathscr{F}(C)$. Moreover, since $\mathscr{F}$ equates support-equivalent arrows, these may be chosen such that $C \circ a$ and $C \circ b$ are defined. From 1-(⇐) we have $a \sim_{\mathscr{L}} b$, hence, by assumption, $C \circ a \sim_{\mathscr{L}} C \circ b$. Applying the functor $\mathscr{F}$, we have from 1-(⇒) that $G \circ p \sim_{\mathscr{F}(\mathscr{L})} G \circ q$ 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 2.27 (relative bisimulation, adequacy). We assume we are given a TS $\mathscr{L}$, with a sub-TS $\mathscr{M}$. A *relative bisimulation for* $\mathscr{M}$ *on* $\mathscr{L}$ is a symmetric relation $\mathscr{S}$ such that whenever $a\mathscr{S}b$, we have that for every transition $a \xrightarrow{L} a'$ in $\mathscr{M}$, with $L \circ b$ defined, there exists b' such that $b \xrightarrow{L} b'$ in $\mathscr{L}$ and $a'\mathscr{S}b'$. We define *relative bisimilarity for* $\mathscr{M}$ *on* $\mathscr{L}$, denoted by $\sim_{\mathscr{L}}^{\mathscr{M}}$, to be the largest relative bisimulation for $\mathscr{M}$ on $\mathscr{L}$.

We say $\mathscr{M}$ is *adequate (for $\mathscr{L}$)* if $\sim_{\mathscr{L}}^{\mathscr{M}}$ coincides with $\sim_{\mathscr{L}}$ on the agents of $\mathscr{M}$, and write this as $\sim_{\mathscr{L}}^{\mathscr{M}} = \sim_{\mathscr{L}} \restriction \mathsf{Int}_{\mathscr{M}}$.

(We have not seen this notion of relative bisimulation anywhere else in the literature.)

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

Relative bisimilarity is valuable when $\mathscr{M}$ is adequate for $\mathscr{L}$, 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 $\mathscr{M}$-transitions than in $\mathscr{L}$-transitions, so 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, that is, 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, Jensen and Milner (2003) shows how to extend the result to a class of *bigraphical* reactive systems (BRSs) broad enough to include the π-calculus (Milner *et al.* 1992) and the ambient calculus (Cardelli and Gordon 2000). 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 2.28 (definite, full sub-TS). Let $\mathscr{M} \prec \mathscr{L}$. Then we say that $\mathscr{M}$ is *definite* for $\mathscr{L}$ if, for some subset Ls of the labels of $\mathscr{L}$, we have

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

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

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

Thus, a definite sub-TS of $\mathscr{L}$ 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 4.2. For a definite sub-TS (hence also for a full sub-TS) we immediately find that a relative bisimilarity is an absolute one.

Proposition 2.29 (definite sub-TS). If $\mathscr{M}$ is definite for $\mathscr{L}$, then $\sim_{\mathscr{M}} = \sim_{\mathscr{L}}^{\mathscr{M}}$.

Corollary 2.30 (adequate sub-congruence). Let $\mathscr{M}$ be definite and adequate for $\mathscr{L}$. Then

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

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

Proposition 2.31 (full sub-congruence). If $\mathscr{M}$ is full for $\mathscr{L}$, then it is also adequate for $\mathscr{L}$, and hence the results of Corollary 2.30 hold.

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

3. Link graphs and their dynamics

3.1. *Introduction to link graphs*

Bigraphical reactive systems (Milner 2001a; 2001b; 2001c; Jensen and Milner 2003; 2004) 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 of basing all distributed computation on a 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 (Milner 1996). They use ideas from many sources: the Chemical Abstract machine (Cham) of Berry and Boudol (1992), the π-calculus of Milner *et al.* (1992), the interaction nets of Lafont (1990), the mobile ambients of Cardelli and Gordon (2000), the explicit fusions of Gardner and Wischik (2000) developed from the fusion calculus of Parrow and Victor (1998), and Nomadic Pict of Wojciechowski and Sewell (1999). They also use the theoretical basis set out in Section 2.

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, that is, forbid certain links to cross its boundary; and it may define what should be replicated or discarded by certain reactions. When none of these are needed, the theory is simpler. But it has been set up (Jensen and Milner 2004) so that the *placing* (that is, 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, 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. Link graphs are reminiscent of several categories of linking, of which many are classified in Bruni *et al.* (2002).

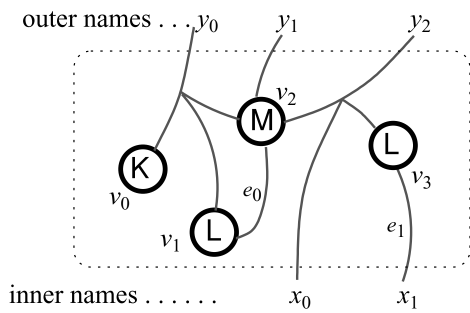


Fig. 2. A link graph $G : \{x_0, x_1\} \to \{y_0, y_1, y_2\}$

The exact definition of link graphs has been a crucial part of the development of bigraphs. In Appendix B the reader will find a discussion of the definition, including a comparison with a previous version.

In Sections 3.2 and 3.3, respectively, we set out the structure and dynamic theory of link graphs, in preparation for their application in Section 4.

3.2. *Link graphs*

In this section we define the notion of a *link graph* formally. In Section 3.3 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.

Definition 3.1 (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 4,

In developing link graphs and LRSs we shall use two running examples with the following signatures:

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

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 already in the literature.

We now proceed to define link graphs over a signature $\mathscr{K}$. 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 that 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 $\mathscr{X}$ of *names*. Formally, we have the following definition.

Definition 3.2 (interface). An *interface* $X, Y, \ldots$ is a finite set of names drawn from $\mathscr{X}$. We refer to the empty interface as the *origin*.

Definition 3.3 (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* ($ctrl : V \to \mathscr{K}$) and a *link map* ($link : X \uplus P \to E \uplus Y$), where $P \stackrel{\text{def}}{=} \sum_{v \in V} ar(ctrl(v))$ is the set of *ports* of A.

We say that the inner names X and ports P are the *points* of A, and that the edges E and outer names Y are 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.

We 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 $y : \varnothing \to \{y\}$, is just a link graph with a single idle name (see Definition 3.8). The *closure* $/x : \{x\} \to \varnothing$ 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.

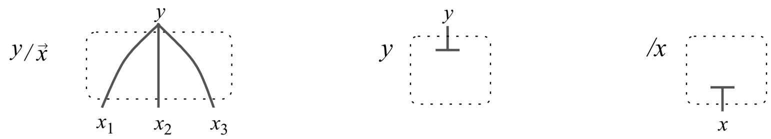


Fig. 3. Elementary wirings

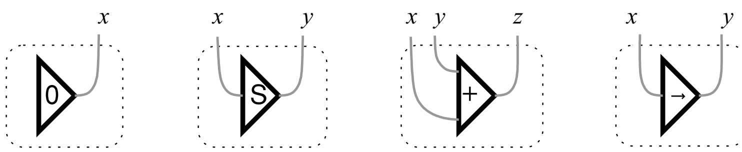


Fig. 4. Atoms for arithmetic nets

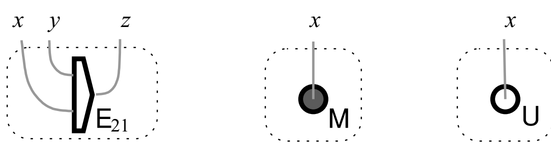


Fig. 5. Atoms for condition-event nets

The second kind of elementary link graph is the *atom* $K\vec{x} : \varnothing \to \{\vec{x}\}$, 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 $\mathbf{0}x$, and so on, for arithmetic nets, and $\mathsf{E}_{21}xyz$, and so on, 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 3.4 (s-category of link graphs). The s-category $\acute{}\text{LIG}(\mathscr{K})$ over a signature $\mathscr{K}$ 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

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

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 $\mathsf{id}_X \stackrel{\text{def}}{=} (\varnothing, \varnothing, \varnothing_{\mathscr{K}}, \mathsf{Id}_X) : X \to X$. A *ground* link graph $G : \varnothing \to X$ 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.

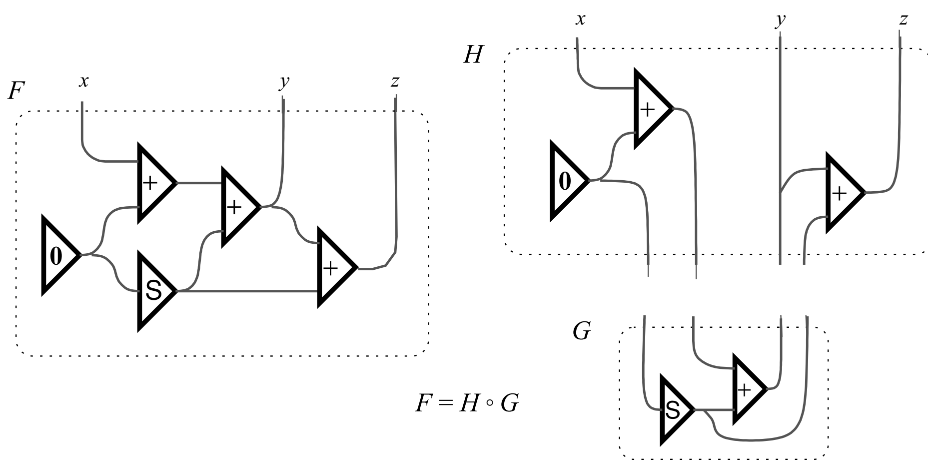


Fig. 6. A ground link graph and its decomposition

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’, that is, synonymous outer names. A previous version of bigraphs (Milner 2001b) 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($\mathscr{K}_{\mathsf{arith}}$). 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 3.8).

Definition 3.5 (tensor product). The *tensor product* $\otimes$ in ´Lig($\mathscr{K}$) 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.

Proposition 3.6 (link graphs are monoidal). The s-category ´Lig($\mathscr{K}$) is monoidal, with origin $\epsilon = \emptyset$.

We say that a tensor product of linkers is a *substitution*, and use σ, τ to range over substitutions. A tensor product of linkers and closures is called a *wiring*, and 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 the tensor product of link graphs that merges outer names, that is, does not require them to be disjoint.

Definition 3.7 (parallel product). The *parallel product* $\mid$ in $'\mathrm{LIG}(\mathscr{K})$ 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 we have

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

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 3.8 (idle, open, closed, peer, lean). A link with no preimage under the link map is *idle*. An outer name is an *open* link, and an edge is a *closed* link. A point (that is, an inner name or port) is *open* if its link is open, otherwise it is *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 3.9 (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.

We need some more notation for what follows.

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 use V_i to denote the node set of A_i $(i = 0, 1)$, and V_2 for $V_0 \cap V_1$. 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 also sometimes use p_i for points, that is, $p_i \in W \uplus P_i$, but the context will resolve any ambiguity.

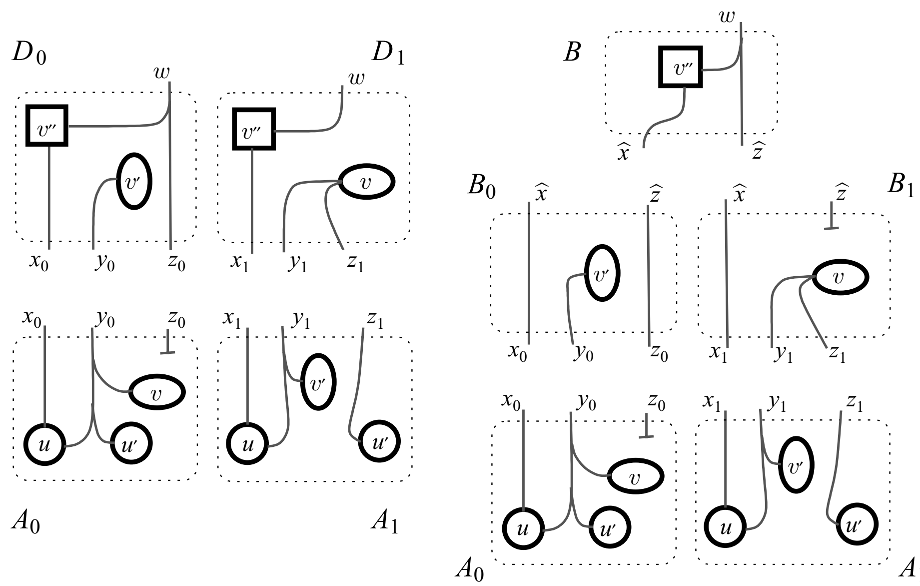


Fig. 7. A bound $\vec{D}$ for $\vec{A}$, and an RPO $(\vec{B}, B)$ for $\vec{A}$ to $\vec{D}$

We now turn to constructing RPOs for concrete link graphs. An informal intuition will help in understanding the construction. Suppose that $\vec{D}$ is a bound for $\vec{A}$ and that 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 $B_0 \circ A_0 = B_1 \circ A_1$. As an example, consider Figure 7 (where controls are omitted for clarity). The left-hand side of the diagram shows a bound $\vec{D}$ for $\vec{A}$, and the right-hand side shows an RPO $(\vec{B}, B)$ for $\vec{A}$ to $\vec{D}$. Note especially that A_0 has an idle name z_0; we shall see later how this affects the family of IPOs for $\vec{A}$.

Formally, the construction of the RPO is as follows.

Construction 3.10 (RPOs in link graphs). An RPO $(\vec{B} : \vec{X} \to \hat{X}, B : \hat{X} \to Z)$ for a pair $\vec{A} : W \to \vec{X}$ of link graphs relative to a bound $\vec{D} : \vec{X} \to Z$ 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 $(V_{\bar{\imath}} - V_2) \uplus V_3$ for some V_3. (Recall that $\bar{\imath}$ is the complement $1 - i$ of i.) Define the nodes of B_i and B to be $V_{\bar{\imath}} - V_2$ $(i = 0, 1)$ and V_3, respectively. Edges are treated exactly analogously, and ports inherit the analogous treatment from nodes.

interface: We construct the outer names $\hat{X}$ of $\vec{B}$ as follows. First, we 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 $X'_0 + X'_1$, we 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 we 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 use $\widehat{i,x}$ to denote the name in $\hat{X}$ corresponding to the $\cong$-equivalence class of (i, x).

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

$$\begin{array}{ll} \text{For } x \in X_0 : & B_0(x) \stackrel{\text{def}}{=} \begin{cases} \widehat{0,x} & \text{if } x \in X'_0 \\ D_0(x) & \text{if } x \notin X'_0 \end{cases} \\ \text{For } p \in P_1{-}P_2 : & B_0(p) \stackrel{\text{def}}{=} \begin{cases} \widehat{1,x} & \text{if } A_1(p) = x \in X_1 \\ D_0(p) & \text{if } A_1(p) \notin X_1 \ . \end{cases} \end{array}$$

Finally, we define B to simulate both D_0 and D_1:

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

To prove that this definition is 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 3.11. The definition in Construction 3.10 is sound.

Proof. The second clause defining $B_0(x)$ is sound, since if $x \notin X'_0$, 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 $p \in P_1{-}P_2$, we have $x \in X'_1$; since otherwise $D_1(x) \in E_0{-}E_2$, 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 for our construction lies in the following lemma and theorem, both of which are proved in Appendix A.

Lemma 3.12. $(\vec{B}, B)$ is a candidate RPO for $\vec{A}$ relative to $\vec{D}$.

Theorem 3.13 (RPOs in link graphs). ´Lig($\mathscr{K}$) 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 3.10 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 in Appendix C.

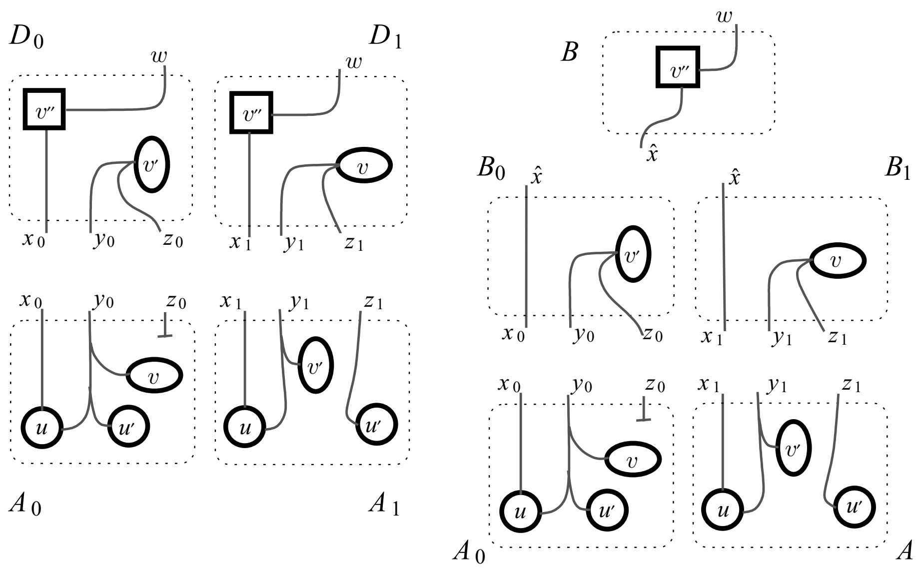


Fig. 8. The RPO $(\vec{B}, B)$ for $\vec{A}$ to $\vec{D}$ varies as $\vec{D}$ varies

Now, in order to prepare for the derivation of labelled transition systems, we proceed to characterise all the IPOs for a given pair $\vec{A}: W \to \vec{X}$ 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 in the pair $\vec{A}$ of Figure 7 the bigraph A_0 is not epi; it has an idle name z_0. The effect is that a different bound $\vec{D}$ for $\vec{A}$, as shown in Figure 8, yields an RPO that treats the idle name z_0 differently. Roughly, since D_0 now 'elides' z_0 into a closed link, so B_0 also elides z_0. These 'elisions' are the only way that IPOs can vary for a fixed pair $\vec{A}$, as we shall see in Construction 6.16, which characterises the family of IPOs for a given pair $\vec{A}$.

Before tackling that construction we need to establish necessary and sufficient conditions under which a pair $\vec{A}$ is consistent, that is, possesses any bound at all.

Definition 3.14 (consistency conditions for link graphs). We define three *consistency* conditions on a pair $\vec{A} : W \to \vec{X}$ of link graphs. Let $P_2 = P_0 \cap P_1$ and $E_2 = E_0 \cap E_1$ be the shared ports and edges, respectively. 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 $A_{\bar{\imath}}(p) = A_{\bar{\imath}}(p_2)$ also, then $p \in W \uplus P_2$ and $A_i(p) = A_i(p_2)$.

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, furthermore, 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. Thus they precisely characterise when a process A_0 may engage in a labelled transition due to a redex A_1. Recall that arrow composition, when defined, does not change node or edge identities. As a result, the choice of redex A_1 fixes how a subset of its nodes and edges (the intersection of supports, $|A_0| \cap |A_1|$) will be aligned with the corresponding ones in A_0 in all possible labelled transitions. Of course, different alignments are possible by considering support translations of A_1.

The consistency conditions closely resemble the *gluing conditions* in graph rewriting (Ehrig 1979). Informally, the gluing conditions operate on the same kind of data as our consistency conditions, namely a redex, a process and an embedding aligning the redex with the process. The gluing conditions then characterise when the process can reduce (by giving a pushout complement). We do not attempt to make the analogy mathematically rigourous since the categories that each system uses are quite different: in link graphs, the objects are graph interfaces and the arrows are contexts, whereas in graph rewriting, the objects are graphs and the arrows are graph embeddings; furthermore, graph rewriting has traditionally been concerned with reductions, not labelled transitions. Recent work (Ehrig 2002; Ehrig and König 2004; Sassone and Sobocinski 2004), however, bridges these gaps and therefore opens up the possibility for a formal comparison.

Returning to our consistency conditions, we see that they are easily implied by the existence of a bound. We treat the converse case afterwards.

Proposition 3.15 (consistency in link graphs). If the pair $\vec{A}$ has a bound, the consistency conditions hold.

As an example, consider the pair $\vec{A} : \varnothing \to \vec{X}$ in Figure 7, where $X_0 = \{x_0, y_0, z_0\}$ and $X_1 = \{x_1, y_1, z_1\}$. Assuming controls are consistently allocated, the pair is consistent, with bound $\vec{D}$ as shown. It is worth checking the consistency conditions.

Now, assuming the consistency conditions of Definition 3.14, 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 3.16 (IPOs in link graphs). We 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 $V_{\bar{\imath}} - V_2$ and $E_{\bar{\imath}} - E_2$.

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

$$K_i' \stackrel{\text{def}}{=} \{x_i \in K_i \mid \forall p \in P_2.\ A_i(p) = x_i \Rightarrow A_{\bar{i}}(p) \in X_{\bar{i}}\}.$$

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 use $\widehat{i, x}$ to denote the $\simeq$-equivalence class of (i, x).

links: Choose two arbitrary maps $\eta_i : L_i \to E_{\bar{i}} - E_2$ $(i = 0, 1)$, 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 (that is, it is epi); or $E_{\bar{i}} - E_2$ is empty (that is, all edges of $A_{\bar{i}}$ 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 3.17. 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 $P_1 - P_2$, and 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 3.18 (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 3.16.

Proof (outline).

(⇒) 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 we assume such a $\vec{B} : \vec{X} \to \widehat{X}$ built by Construction 3.10, and recall the subsets $X'_i \subseteq X_i$ and the equivalence $\cong$ over $: X'_0 + X'_1$ defined there. Now we apply Construction 3.16 to create a pair $\vec{C} : \vec{X} \to Y$ by choosing the sets $\vec{L}$ and elision maps $\vec{\eta}$ as follows:

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

Then $\vec{C}$ does indeed coincide with $\vec{B}$. To prove this, we 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; so the codomain Y of $\vec{C}$ coincides with the codomain $\widehat{X}$ of $\vec{B}$. Then we show that the link maps C_i coincide with B_i. Thus every IPO is a bound built by Construction 3.16.

(⇐) To prove the converse, we consider any bound $\vec{C} : \vec{X} \to Y$ built by Construction 3.16, for some sets $\vec{L}$ and elision maps $\vec{\eta}$. Now we apply Construction 3.10 to yield an RPO $(\vec{B}, B)$ for $\vec{A}$ to $\vec{C}$.
Then $\vec{B}$ does indeed coincide with $\vec{C}$ up to isomorphism. To prove this, we 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; so the codomain $\widehat{X}$ of $\vec{B}$ coincides with the codomain Y of $\vec{C}$. Then we show that the link maps B_i coincide with C_i. Thus, every bound built by Construction 3.16 is an IPO. □

The reader may like to check the IPO construction by confirming that the two bounds $\vec{B}$ for $\vec{A}$ shown in Figures 7 and 8 are both IPOs. In fact, they constitute the entire family of RPOs for $\vec{A}$ since the pair has only one idle name z_0, and there is only one closed link in B_0 to which it may or may not be elided.

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

Proposition 3.19 (tensor IPO). In $'\textsc{Lig}(\mathscr{K})$, 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 follows.

(a)
$$\begin{tikzcd} X \otimes Y' \arrow[r, "\mathsf{id} \otimes B"] & X \otimes Y \\ X' \otimes Y' \arrow[u, "A \otimes \mathsf{id}"] \arrow[r, "\mathsf{id} \otimes B"] & X' \otimes Y \arrow[u, "A \otimes \mathsf{id}"'] \end{tikzcd}$$

(b)
$$\begin{tikzcd} X \arrow[r, "\mathsf{id} \otimes b"] & X \otimes Y \\ \epsilon \arrow[u, "a"] \arrow[r, "b"] & Y \arrow[u, "a \otimes \mathsf{id}"'] \end{tikzcd}$$

Corollary 3.20 (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 3.21 (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 3.10 is rigid in the sense of Definition 2.11, that is, 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 2.12 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 2.10(2). □

Recall that a link graph is *lean* if it has no idle edges. In Section 3.3 we shall need to transform IPOs by the addition or subtraction of idle edges. We write A^E for the result of adding a set E of fresh idle edges to A. The following proposition is easy to prove from the IPO construction for link graphs.

Proposition 3.22 (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 $\risingdotseq$ that is a little coarser than support equivalence ($\doteq$).

Definition 3.23 (abstract link graphs and their category). Two concrete link graphs A and B are *lean-support equivalent*, written $A \risingdotseq B$, if after discarding any idle edges they are support equivalent. The category $\mathrm{L{\small IG}}(\mathscr{K})$ of *abstract link graphs* has the same objects as $\acute{}\mathrm{L{\small IG}}(\mathscr{K})$, and its arrows are lean-support equivalence classes of concrete link graphs. Lean-support equivalence is clearly a static congruence (Definition 2.5). The associated quotient functor, as defined in Definition 2.6, is

$$[\![\cdot]\!] : \acute{}\mathrm{L{\small IG}}(\mathscr{K}) \to \mathrm{L{\small IG}}(\mathscr{K})\ .$$

The reason for studying concrete, rather than abstract, link graphs is that they possess RPOs. This will allow us in Section 3.3 to derive a behavioural congruence for ´Lɪɢ, and then to show how to transfer it, under certain assumptions, to Lɪɢ.

To see why we cannot work directly in Lɪɢ, we point out that it lacks some structure present in ´Lɪɢ. For example, the functor $[\![\cdot]\!]$ does not preserve epis. More seriously, Lɪɢ 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 Jensen and

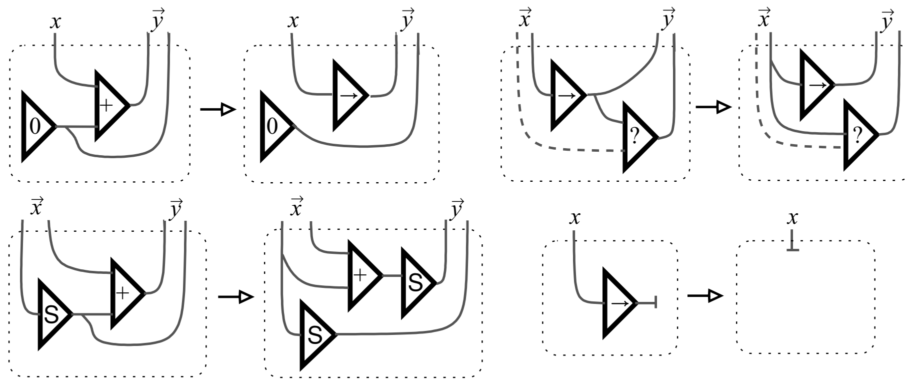


Fig. 9. Reaction rules for arithmetic

Milner (2004). (It is presented in terms of bigraphs, but involves only their link graph components.)

3.3. *Reactions and transitions for link graphs*

We are now ready to specialise the definitions and theory for reactive systems (RSs) in Section 2.3 to obtain link-graphical reactive systems (LRSs), which form the objects of a category whose arrows are RS functors.

Definition 3.24 (link-graphical reactive system). A *(concrete) link-graphical reactive system (LRS)* over a signature $\mathcal{K}$ consists of a monoidal reactive system over $'\mathrm{L{\scriptsize IG}}(\mathcal{K})$, with a rule-set $'\mathcal{R}$ in which no redex has an idle link. We denote it by

$$'\mathrm{L{\scriptsize IG}}(\mathcal{K}, '\mathcal{R}) \ .$$

As an example, Figure 9 shows a likely set of rules for the evaluation of arithmetic nets, whose atoms appeared in Figure 4. The two rules on the left-hand side represent the primitive recursive definition of $+$, while the two rules on the right 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 4 we shall capture this distinction by imposing a sort-discipline on link graphs.) The top left-hand rule introduces a forwarder node. The top right-hand rule creates a bypass around a forwarder; it is really a family of rules, since '**?**' represents any of the three controls $\{\mathsf{S}, \mathbf{+}, \rightarrow\}$, and the dotted link represents any extra inputs to the node with that control. The bottom 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 (Hasegawa 1999), which enrich Lafont's interaction nets (Lafont 1990) 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 on IPOs, leading to the standard transition system ST. LRSs thereby inherit the definition of bisimilarity, so we have the following corollary of Theorem 2.24.

Corollary 3.25 (congruence of bisimilarity). In any concrete LRS equipped with the standard transition system ST, bisimilarity of agents is a congruence.

It is natural to ask whether identity transitions $a \xrightarrow{\mathsf{id}} a'$ differ from reactions $a \longrightarrow a'$. The two clearly coincide in the full transition system FT; but even 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.

Proposition 3.26 (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 $a' \doteq D \circ r'$ for some rule (r, r'). But r has no idle names, so by Proposition 3.9, 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. Thus 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 $\mathrm{LIG}(\mathscr{K},\mathscr{R})$, where $\mathrm{LIG}(\mathscr{K})$ is defined by the quotient functor $[\![\cdot]\!]$ of Definition 3.23, and $\mathscr{R}$ is also obtained from $\acute{}\mathscr{R}$ by $[\![\cdot]\!]$.

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

Proposition 3.27 (transitions respect equivalence). In a concrete LRS equipped with standard transitions:

1 Every transition label L is lean.
2 Transitions respect lean-support equivalence ($\simeq$) in the sense of Definition 2.20. That is, for every transition $a \xrightarrow{L} a'$, if $a \simeq b$ and $L \simeq M$ where M is another label with $M \circ b$ defined, then there exists a transition $b \xrightarrow{M} b'$ for some b' such that $a' \simeq b'$.

Proof. For the first part, use Proposition 3.22(1). For the second part, use Proposition 3.22(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 corollary is immediate by invoking Theorem 2.26 and Proposition 3.27, followed by Corollary 3.25.

Corollary 3.28 (behavioural congruence in abstract LRSs). Let $\acute{\mathbf{A}}$ be a concrete LRS equipped with a TS $\mathscr{L}$ that respects lean-support equivalence. We use $\mathbf{A}$ to denote the lean-support quotient of $\acute{\mathbf{A}}$, and $\sim_{\mathscr{L}}$ for the bisimilarity induced by $\mathscr{L}$ in both $\acute{\mathbf{A}}$ and $\mathbf{A}$. Then:

1 $a \sim_{\mathscr{L}} b$ in $\acute{\mathbf{A}}$ iff $[\![a]\!] \sim_{\mathscr{L}} [\![b]\!]$ in $\mathbf{A}$.
2 If $\sim_{\mathscr{L}}$ 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 2.3, 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 3.2 that a link is *open* if it is an outer name, and *closed* otherwise, and that these properties are inherited by the points of the link.

Definition 3.29 (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 3.30 (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 3.21 it is a pushout.
3 Apply part 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. (Recall from Figure 3 and the discussion following Definition 3.3 that $x : \emptyset \to \{x\}$ is a link graph consisting of the single idle name x.).

Proposition 3.31 (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, we use congruence. For the converse, we shall verify that $\mathscr{S} = \{(a,b) \mid x \otimes a \sim x \otimes b\}$ is a bisimulation up to $\doteq$.

(a)
$$\begin{tikzcd}
{} \arrow[r, "\mathsf{id}_x \otimes L"] & {} \\
{} \arrow[u, "x \otimes \mathsf{id}"] \arrow[r, "L"] & {} \arrow[u, "x \otimes \mathsf{id}"'] \\
{} \arrow[u, "a"] \arrow[r, "r"] & {} \arrow[u, "D"']
\end{tikzcd}$$

(b)
$$\begin{tikzcd}
{} \arrow[r, "\mathsf{id}_x \otimes L"] & {} \\
{} \arrow[u, "x \otimes b"] \arrow[r, "s"] & {} \arrow[u, "G"']
\end{tikzcd}$$

(c)
$$\begin{tikzcd}
{} \arrow[r, "\mathsf{id}_x \otimes L"] & {} \\
{} \arrow[u, "x \otimes \mathsf{id}"] \arrow[r, "M"] & {} \arrow[u, "F"'] \\
{} \arrow[u, "b"] \arrow[r, "s"] & {} \arrow[u, "E"']
\end{tikzcd}$$

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

The IPO underlying the transition of a is the bottom square of diagram (a), based on a rule (r, r') with $a' \doteq D \circ r'$. By Corollary 3.20, 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

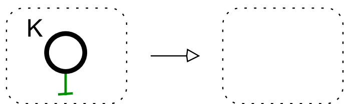
$$x \otimes b \xrightarrow{\mathsf{id}_x \otimes L} G \circ s' \sim x \otimes a'$$

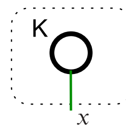
with underlying IPO as in diagram (b). Now $x \otimes b = (x \otimes \mathsf{id}) \circ b$, so, by taking an RPO (M, E, F) for (b, s), we obtain a pair of IPOs as in (c). By Proposition 3.30(1), M is simple, and by Proposition 3.30(3), the upper square of (c) is a pushout. But, by Corollary 3.20, the 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 $b \xrightarrow{L} b' \overset{\text{def}}{=} E \circ s'$, and since $G = x \otimes E$, we have $G \circ s' = x \otimes b'$. So $(a', b') \in \mathscr{S}^{\doteq}$ as required. $\square$

We now turn to engaged transitions (see the discussion of engaged transitions in Section 2.3).

Definition 3.32 (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 use ET to denote 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 the engaged transitions are adequate for standard bisimilarity (Definition 2.27), that is, that $\sim^{\text{ET}} = \sim$, since that would mean that in order to establish $a \sim b$ we need only match each *engaged* transition of a (respectively, b) by an arbitrary transition of b (respectively, a). This is a lighter task than matching *all* transitions. If an LRS is not simple, its engaged transitions may not be adequate. For example, take two controls K and N with arity 1, and let there be a single reaction rule as shown on the left in the following diagram:





$/x \circ \mathsf{K}_x \longrightarrow \emptyset$ $\qquad$ $L = /x \circ (\mathsf{id}_x \mid \mathsf{K}_x)$

The rule is not simple, because the redex is not open. Now consider two agents, N_x and x, the latter being just an idle name. Observe that, in the context L shown, N_x has no reaction, while x has a reaction; in fact $L \circ \mathsf{N}_x \not\longrightarrow$, while $L \circ x = /x \circ \mathsf{K}_x \longrightarrow \emptyset$. Thus $\mathsf{N}_x \not\sim x$. This is reflected directly by the fact that N_x has no L-transition, while $x \xrightarrow{L} \emptyset$ by an elisive transition. However, this transition is not engaged; in fact, neither N_x nor x has any engaged transition, so $\mathsf{N}_x \sim^{\text{ET}} x$.

We shall now prove that, in a simple LRS, the engaged transitions are indeed adequate, that is, $a \sim^{\text{ET}} b$ implies $a \sim b$. To do this, 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, in a simple LRS we find that non-engaged transition of a can be suitably matched by *any* b (whether or not $a \sim^{\text{ET}} b$).

Theorem 3.33 (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

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

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 $M \circ C \circ a_1$ defined. We must find b'_1 such that $C \circ a_1 \xrightarrow{M} b'_1$ and $(b'_0, b'_1) \in \mathscr{S}$.

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

(a)
$$\begin{tikzcd} \cdot \arrow[r, "M"] & \cdot \\ \cdot \arrow[u, "C \circ a_0"] \arrow[r, "r_0"'] & \cdot \arrow[u, "E_0"'] \end{tikzcd}$$

(b)
$$\begin{tikzcd} \cdot \arrow[r, "M"] & \cdot \\ \cdot \arrow[u, "C"] \arrow[r, "L"] & \cdot \arrow[u, "E"'] \\ \cdot \arrow[u, "a_0"] \arrow[r, "r_0"'] & \cdot \arrow[u, "D_0"'] \arrow[uu, bend right, "E_0"'] \end{tikzcd}$$

(c)
$$\begin{tikzcd} \cdot \arrow[r, "L"] & \cdot \\ \cdot \arrow[u, "a_1"] \arrow[r, "r_1"'] & \cdot \arrow[u, "D_1"'] \end{tikzcd}$$

Hence, $a_0 \xrightarrow{L} a_0'$, where $a_0' = D_0 \circ r_0'$. Moreover, by Proposition 3.30(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 $(b_0', b_1') \in \mathscr{S}$, thus completing the proof of the theorem.

Case 1: Suppose the transition $a_0 \xrightarrow{L} a_0'$ is not engaged, that is, $|a_0| \cap |r_0| = \emptyset$. 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 3.20 must be a tensor IPO.† 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 $(b_0', b_1') \in \mathscr{S}$, as required.

Case 2: Suppose the transition $a_0 \xrightarrow{L} a_0'$ is engaged, that is, $|a_0| \cap |r_0| \neq \emptyset$. 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 $(b_0', b_1') \in \mathscr{S}$, as required. □

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

$$[\![\cdot]\!] : {}^{\prime}\mathrm{L{\small IG}}(\mathscr{K}) \to \mathrm{L{\small IG}}(\mathscr{K}) \ .$$

To do this, we would like to know that ET is *definite* for ST (see Definition 2.28), since then, by Proposition 2.29, 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, and which is sufficient to ensure definiteness.

Definition 3.34 (proper LRS). Define $ctrl(G)$, the *control* of a link graph G, to be the multiset of controls of its nodes. An LRS is *proper* if for any two redexes r and s, if $ctrl(r) \subseteq ctrl(s)$, then $ctrl(r) = ctrl(s)$.

† A forerunner of this phenomenon, that a non-engaged transition must be based on a tensor IPO, appears as Theorem 3.33 in the first author's Ph.D. Dissertation (Leifer 2001).

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 2.30 and 3.25, we deduce the following corollary.

Corollary 3.35 (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, that is, $a \sim_{\text{ET}} b$ implies $C \circ a \sim_{\text{ET}} C \circ b$

Proof. For the first part we must prove that if a standard transition $a \xrightarrow{L} a'$ is engaged, then every standard transition with label L is engaged. To produce a contradiction, suppose this is not the case, that is, suppose that some other standard transition $b \xrightarrow{L} b'$ is not engaged. Let r and s be the redexes underlying the transitions of a and b, respectively. Then, from the construction of RPOs, we have $|L| \subsetneq |r|$ and $|L| = |s|$. Hence $|s| \subsetneq |r|$. So, since nodes in an IPO for $\vec{A}$ inherit their controls from $\vec{A}$, it follows that $ctrl(s) \subsetneq ctrl(r)$ (as multisets), contradicting the assumption that the LRS is proper.

Since simpleness implies the adequacy of ET, the last two parts follow directly. □

Now recall from Proposition 3.30 that every simple link graph is lean. We therefore specialise Corollary 3.28 to ET under appropriate assumptions.

Corollary 3.36 (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 $\sim_{\text{ET}}$ denote bisimilarity both for ET in $\acute{\mathbf{A}}$ and for the induced transition system $[\![\text{ET}]\!]$ in $\mathbf{A}$. Then:

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

Proof. The quotient functor satisfies the conditions of Theorem 2.26. In particular, by Proposition 3.27 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 3.35. □

Thus we have ensured congruence of engaged bisimilarity in any abstract LRS $\text{L}\textsc{ig}(\mathscr{K})$ satisfying reasonable assumptions.

4. Sorted link graphs

4.1. *Sorting and condition-event nets*

Section 4 is devoted to the application of link graph theory. We begin in this subsection with the topic of *sorting*, which is likely to be needed in any significant application. Then in Sections 4.2 and 4.3 we apply link graph theory, together with our theory of transitions systems, to deriving a behavioural congruence for a class of Petri nets. The work on Petri nets was first reported by the second author in Milner (2004a); the present approach improves on this by adding an adequacy result (see Proposition 4.11).

Our sorting discipline for link graphs, which was first proposed for bigraphs (Milner 2001b), 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 (Jensen and Milner 2004); this sets limits on the scope of a name, so that it cannot be linked to a port outside that scope.

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

Definition 4.1 (sorted link graphs). A signature $\mathscr{K}$ 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* $\mathscr{K}$ if its interfaces are Θ-sorted, and for each K, i the sort assigned by $\mathscr{K}$ to $i \in ar(K)$ is ascribed to the i^{th} port of every K-node.

We use $\acute{}\mathrm{L}\textsc{ig}(\Theta, \mathscr{K})$ to denote 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 4.2 (sorting). A *sorting (discipline)* is a triple $\Sigma = (\Theta, \mathscr{K}, \Phi)$ where $\mathscr{K}$ is Θ-sorted, and Φ is a condition on Θ-sorted link graphs over $\mathscr{K}$. The condition Φ must be satisfied by the identities and preserved by both composition and tensor product.

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

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

Even if we confine ourselves to a single sort, there are some 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 $\Sigma = (\Theta, \mathscr{K}, \Phi)$ in order that we may apply our transition theory? These constraints are best understood in terms of the obvious forgetful functor that discards sorts:

$$\mathscr{U} : \acute{}\mathrm{L}\textsc{ig}(\Sigma, \acute{}\mathscr{R}) \to \acute{}\mathrm{L}\textsc{ig}(\mathscr{U}(\mathscr{K}), \mathscr{U}(\acute{}\mathscr{R}))\,.$$

We shall say $\mathscr{U}$ is a *sorting* functor. Such functors have certain properties.

Proposition 4.3 (sorting is faithful). On interfaces, a sorting functor is surjective (but not in general injective). It is also *faithful*, that is, 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 the two properties in the following definition that a functor of precategories may have.

Definition 4.4 (creating RPOs, reflecting pushouts). Let $\mathscr{F}$ be any functor on a precategory $\acute{\mathbf{A}}$. Then $\mathscr{F}$ *creates RPOs* if, whenever $\vec{D}$ bounds $\vec{A}$ in $\acute{\mathbf{A}}$, then any RPO for $\mathscr{F}(\vec{A})$ relative to $\mathscr{F}(\vec{D})$ has a unique $\mathscr{F}$-preimage that is an RPO for $\vec{A}$ relative to $\vec{D}$.

$\mathscr{F}$ *reflects pushouts* if, whenever $\vec{D}$ bounds $\vec{A}$ in $\acute{\mathbf{A}}$ and $\mathscr{F}(\vec{B})$ is a pushout for $\mathscr{F}(\vec{A})$, we have that $\vec{B}$ is a pushout for $\vec{A}$.

Corollary 4.5 (creation ensures RPOs). If $\mathscr{F} : \acute{\mathbf{A}} \to \acute{\mathbf{B}}$ 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 ... ', and so on.

It turns out that if a sorting satisfies the two conditions of Definition 4.4 (which appear to be independent, but we need not settle that question here), then we get sufficient structure for our transition theory.

Theorem 4.6 (useful sortings).

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

Note that *simpleness* for a well-sorted link graph is just the same as for a pure one. (Indeed, sorting functors both preserve and reflect simpleness.) We omit the proof of the theorem: it follows closely along the lines of the proofs of Theorems 2.24 and 3.33; for the latter, the reflection of pushouts enables Proposition 3.30 to be lifted to the well-sorted LRS.

We are now ready to define the sorting discipline that we shall use in the remainder of the paper. It may be motivated by our arithmetic nets, in which we want 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.

Definition 4.7 (many-one sorting). In a *many-one sorting* $\Sigma = (\Theta, \mathscr{K}, \Phi)$ the sorts are $\Theta = \{\mathsf{s}, \mathsf{t}\}$, the signature $\mathscr{K}$ 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.

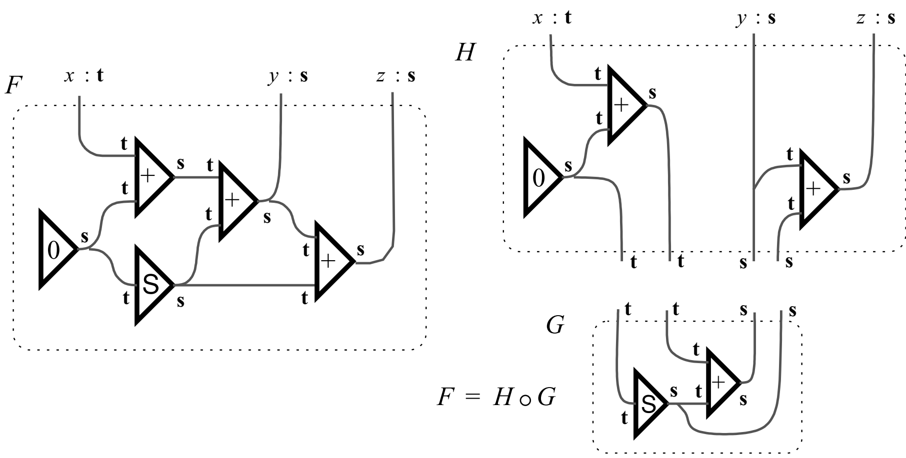


Fig. 10. A well-sorted arithmetic net and its decomposition

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

We can illustrate this using arithmetic nets. In this case the sorted signature is $\mathscr{K}_{\mathsf{arith}}$ as defined at the beginning of Section 3.2 enriched by the assignment of s to output ports and t to input ports; for example, **+** is assigned the sort-sequence tts. Figure 10 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 that is not shared by all sortings:

$$\mathscr{U}\downarrow \qquad \begin{tikzcd} X \arrow[r, "F"] \arrow[rr, bend left, "H"] \arrow[d, dotted, no head] & Y \arrow[r, dashed, "\exists G"] \arrow[d, dotted, no head] & Z \arrow[d, dotted, no head] \\ {} \arrow[r] & {} \arrow[r, "G'"] & {} \end{tikzcd}$$

Proposition 4.8 (many-one sorted decomposition). Let $\mathscr{U}$ be a many-one sorting functor, and let

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

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

Note that, since $\mathscr{U}$ 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, it is not hard to show that many-one sorting has the structure we need.

Theorem 4.9 (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 3.10 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 4.8.

The second property can be proved for *any* functor of precategories that is faithful and enjoys the property in Proposition 4.8. □

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.

4.2. *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 *et al.* (1992) gives a comprehensive survey of such equivalences and preorders. It covers those based on observation both of actions and of states, and range from fine relations respecting causality to coarser ones such as the failures preorder from CSP, which is the coarsest respecting deadlock. The study of congruence of these relations, that is, whether they are preserved by contexts, and which contexts *should* preserve them, was reported as being rather incomplete in 1992.

Nielsen *et al.* (1995) characterises some behavioural congruences on nets. Given a semantic function $\mathscr{B}$ that assigns an abstract behaviour to each net, the congruence $\approx$ it induces on nets is considered; this is defined by

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

An important contribution of the paper is to define a precise notion of context by means of a set of *combinators* on nets. The authors are then able to characterise the congruences, for each of four semantic functions $\mathscr{B}$, 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 (1998) continues this programme by enriching the net combinators, and considering a wider range of semantic functions.

The Petri Box calculus of Best *et al.* (1999), 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.

Baldan *et al.* (2001) defines a class of *open* Petri nets, having input and output places where tokens may, respectively, be added and removed at any time. The authors define a form of composition of two such nets that allows interaction at these places, and define a semantics of a net in terms of its *processes*, that is, the deterministic nets representing its possible behaviours. The semantics is shown to be compositional, that is, the composition of two open nets respects their underlying processes.

This brief summary does not do justice to the five papers, which form a good representation of the progress towards a modular treatment of Petri nets. But it does help us to identify where it differs from the theory of bigraphs (or link graphs), which

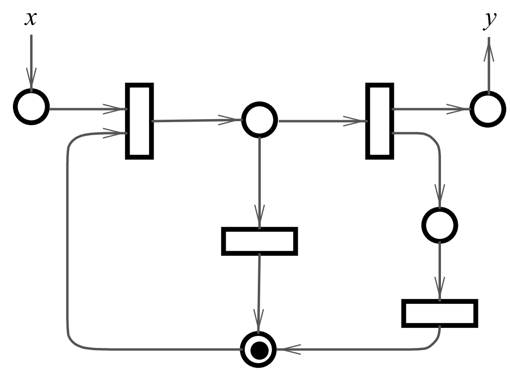


Fig. 11. A condition-event net with two observable conditions

suggests 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, such as 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, we now consider *condition-event* Petri nets, as illustrated in Figure 11. These are nets in which each place, or condition, may be either marked (that is, 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 differs from those in the above-cited papers, but is, nevertheless, quite natural. It provides a good 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.* (2001), 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 11 has two accessible conditions, named x and y. In general, given a state g, that is, 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.

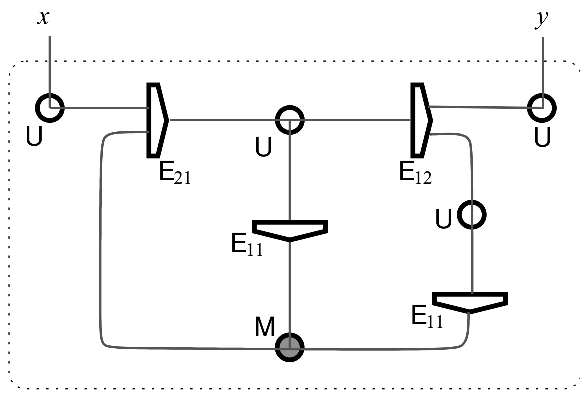


Fig. 12. A condition-event net represented as a link graph

These three kinds of transition are the basis of a *raw* TS $\mathscr{L}_{\mathsf{p}}$, with which we shall equip our LRS of Petri nets, in order to compare it with another TS $\mathscr{L}_{\mathsf{g}}$, which we shall derive from reaction rules by the methods discussed in Sections 2 and 3 of this paper.

We now set up condition-event nets as link graphs. There are many ways to do this; we choose one that gives a smooth treatment. Figure 12 shows the net of Figure 11 as a link graph, using the signature $\mathscr{K}_{\mathsf{petri}}$ defined at the start of Section 3.2 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. Thus, for example, the three arcs that impinge on the marked node in Figure 11 are represented in Figure 12 as a single arc connected to the single port of the M-node. 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 $\mathscr{K}_{\mathsf{petri}}$ by assigning the sort s to all condition ports and t to event ports. This leads us to the sorting discipline

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

where $\Theta_{\mathsf{petri}} = \{\mathsf{s}, \mathsf{t}\}$ and Φ_{petri} is the many-one sorting condition of Definition 4.7. Then the concrete precategory of many-one sorted condition-event nets is

$$\acute{}\mathbf{CE} \stackrel{\text{def}}{=} \acute{}\textsc{Lig}(\Sigma_{\mathsf{petri}}),$$

and we denote its lean-support quotient by **CE**. Although these nets share many-one sorting with arithmetic nets, there is a considerable difference, which 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. However, we do not yet claim any general taxonomy of sorting disciplines, since to develop this will require the study of a wider range of applications.

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 we define an

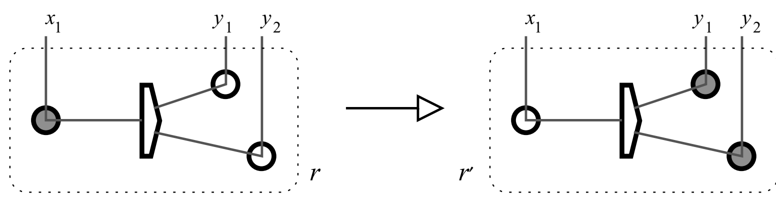


Fig. 13. A link-graph reaction rule for condition-event nets

s-*interface* to be one containing only s-names. We can then model condition-event nets in ´**CE** and **CE** as link graphs with s-interfaces, and will call them s-*nets*.

Without further ado, we now set up in ´**CE** a raw transition system $\mathscr{L}_{\mathsf{p}}$ 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 $[\![\mathscr{L}_{\mathsf{p}}]\!]$ in **CE**. We will use $\sim_{\mathsf{p}}$ for the associated bisimilarity in both cases. Since no RPO theory is involved, we readily get the following proposition.

Proposition 4.10 (raw bisimilarity).
1 $a \xrightarrow{\ell} a'$ in ´**CE** iff $[\![a]\!] \xrightarrow{\ell} [\![a']\!]$ in **CE**.
2 $a \sim_{\mathsf{p}} b$ in ´**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 ´**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 13 for $h = 1, k = 2$. As required by Definitions 2.14 and 3.24, we close this set under support translation and make each rule lean (no idle edges). Having thus established ´**CE** as a concrete LRS, we equip it with the standard transition system ST. We can then apply Corollary 3.25 to establish that the associated bisimilarity $\sim_{\mathsf{g}}$, is a congruence.

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

Proposition 4.11 (adequacy for nets). The engaged transition system ET over ´**CE** is definite and adequate for ST, so its bisimilarity coincides with $\sim_{\mathsf{g}}$.

Proof. It is easy to show that ´**CE** is simple, as defined in Definition 3.29. It is also proper, according to Definition 3.34. Therefore, by Corollary 3.35, we may reduce ST to ET without affecting the induced bisimilarity $\sim_{\mathsf{g}}$. □

The second refinement step is to reduce the agents to s-nets. We define the TS $\mathscr{L}_{\mathsf{g}}$ 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 $\mathscr{L}_{\mathsf{g}}$ is a full sub-TS of ET. Therefore by Proposition 2.31 and Corollary 2.30 we have the following corollary.

Corollary 4.12 (bisimulation congruence for concrete s-nets). Bisimilarity for the transition system $\mathscr{L}_{\mathsf{g}}$ coincides with $\sim_{\mathsf{g}}$ on s-nets and is a congruence.

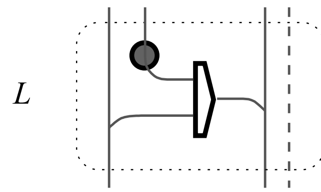


Fig. 14. A typical label in $\mathscr{L}_{\mathsf{g}}$

We have now taken the theory of $\mathscr{L}_{\mathsf{g}}$ for concrete s-nets as far as we need, except for characterising its transitions. We leave that task to Section 4.3. Here, noting that $\mathscr{L}_{\mathsf{g}}$ respects lean-support equivalence, we can relate it to the TS $[\![\mathscr{L}_{\mathsf{g}}]\!]$ induced on abstract s-nets, using Corollaries 4.12 and 3.28.

Corollary 4.13 (bisimulation congruence for abstract s-nets). We use $\sim_{\mathsf{g}}$ to denote the bisimilarity induced on **CE** by the abstract TS $[\![\mathscr{L}_{\mathsf{g}}]\!]$. Then:

1 $a \sim_{\mathsf{g}} b$ in ´**CE** iff $[\![a]\!] \sim_{\mathsf{g}} [\![b]\!]$ in **CE**.
2 The bisimilarity $\sim_{\mathsf{g}}$ is a congruence in **CE**.

4.3. *Coincidence of bisimilarities*

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

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 $\mathscr{L}_{\mathsf{g}}$. We omit the detailed analysis; it uses the fact that transitions are engaged (Proposition 4.11) and that labels are simple (Proposition 3.30) 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, which is linked to zero or more M-nodes as preconditions and U-nodes as post-conditions.

For the identity labels, we recall from Proposition 3.21 that $a \xrightarrow{\mathsf{id}} a'$ iff $a \longrightarrow a'$; an id label signifies a transition with no help from the context.

Figure 14 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 15 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 notation $\overline{a}$ to denote an 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

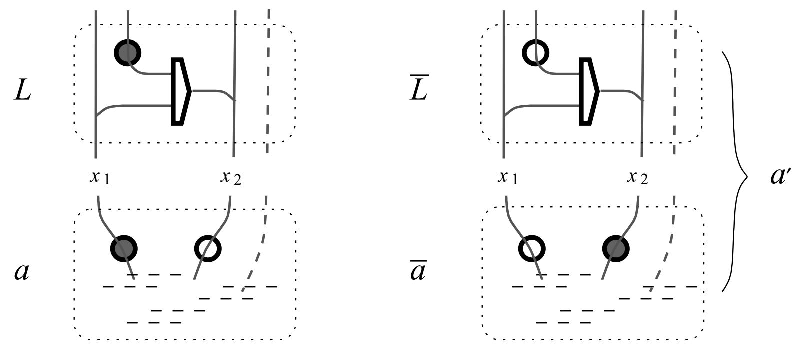


Fig. 15. Anatomy of a transition $a \xrightarrow{L} a'$ in $\mathscr{L}_{\mathsf{g}}$

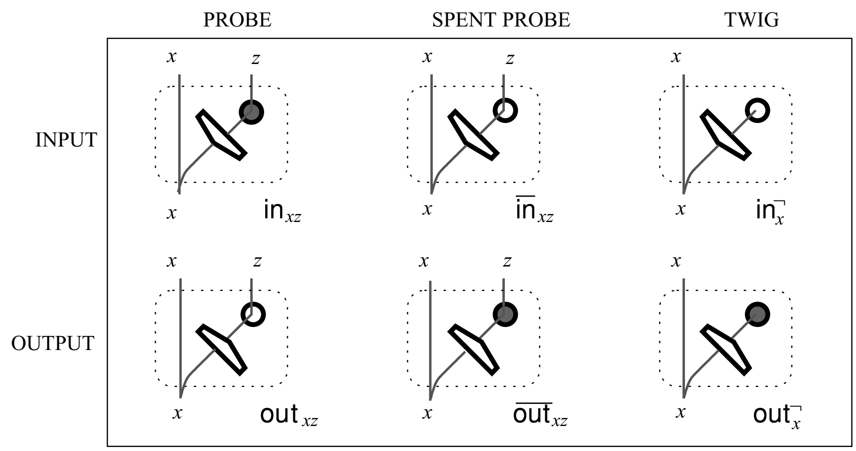


Fig. 16. Probes for observing conditions in a s-net

a will have a similar transition, provided only that it has the same initial marking of its named conditions.

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

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 14), 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 in_{xz} and out_{xz}, and are shown in the first column of Figure 16. The second column shows the *spent* probes $\overline{P}$, the 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 4.14. For any s-agent a having x in its outer face, $\mathsf{in}_x^{\neg} \circ a \sim_{\mathsf{g}} \mathsf{out}_x^{\neg} \circ a \sim_{\mathsf{g}} a$.

Proof (outline). For example, we must prove that there is a bisimulation of the form $\mathscr{S} = \{(\mathsf{in}_x^{\neg} \circ a, a) \mid a \text{ an agent}\}$. For this, we must use the fact that $\mathsf{in}_x^{\neg}$ cannot contribute to transitions; that is, $\mathsf{in}_x^{\neg} \circ a \xrightarrow{L} b'$ iff there is a transition $a \xrightarrow{L} a'$ with $b' = \mathsf{in}_x^{\neg} \circ a'$. We leave details to the reader. □

Here we have abbreviated $\mathsf{in}_x^{\neg} \otimes \mathsf{id}$ to $\mathsf{in}_x^{\neg}$, and we will continue to use such abbreviations in what follows, but only in a composition that 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 $\mathscr{L}_{\mathsf{p}}$-bisimulation. For this, suppose that $a \sim_{\mathsf{g}} b$, and let $a \xrightarrow{\ell} \overline{a}$ in $\mathscr{L}_{\mathsf{p}}$. 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 our assumption then implies that $a \longrightarrow \overline{a}$, and hence $a \xrightarrow{\mathsf{id}} \overline{a}$ in $\mathscr{L}_{\mathsf{g}}$; but then, by bisimilarity in $\mathscr{L}_{\mathsf{g}}$, 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 in $\mathscr{L}_{\mathsf{g}}$ we have

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

Hence, by bisimilarity in $\mathscr{L}_{\mathsf{g}}$, 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 $\mathscr{L}_{\mathsf{p}}$. 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 4.14:

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

Therefore we have proved the following lemma, as desired.

Lemma 4.15. $\sim_{\mathsf{g}} \subseteq \sim_{\mathsf{p}}$ 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

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

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, ´**CE** and **CE**, respectively. Furthermore, we shall use the convenient fact that, by Proposition 3.30(3), in ´**CE**, every IPO is actually a pushout.

So let us assume that $a \sim_{\mathsf{p}} b$, and that $C \circ a \xrightarrow{M} a''$ in $\mathscr{L}_{\mathsf{g}}$. (This covers the case $M = \mathsf{id}$.) Then there is a reaction rule r and context D such that (M, D) forms a pushout for

$$
\begin{tikzcd}
{} \arrow[r, "M"] & {} \\
{} \arrow[u, "C \circ f"] \arrow[r, "r"'] \arrow[ur, phantom, "\llcorner", very near end] & {} \arrow[u, "D"']
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{} \arrow[r, "M"] & {} \\
{} \arrow[u, "C"] \arrow[r, "L"] \arrow[ur, phantom, "\llcorner", very near end] & {} \arrow[u, "C'"'] \\
{} \arrow[u, "f"] \arrow[r, "r"'] \arrow[ur, phantom, "\llcorner", very near end] & {} \arrow[u, "F"'] \arrow[uu, bend right=60, "D"']
\end{tikzcd}
$$

Fig. 17. Pushouts underlying transitions of $C \circ a$ and a

$$
\begin{tikzcd}
{} \arrow[r, "L"] & {} \\
{} \arrow[u, "g"] \arrow[r, "s"'] \arrow[ur, phantom, "\llcorner", very near end] & {} \arrow[u, "G"']
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{} \arrow[r, "M"] & {} \\
{} \arrow[u, "C"] \arrow[r, "L"] \arrow[ur, phantom, "\llcorner", very near end] & {} \arrow[u, "C'"'] \\
{} \arrow[u, "g"] \arrow[r, "s"'] \arrow[ur, phantom, "\llcorner", very near end] & {} \arrow[u, "G"'] \arrow[uu, bend right=60, "E"']
\end{tikzcd}
$$

Fig. 18. Pushouts underlying transitions of b and $C \circ b$

$(C \circ a, r)$, as shown in the left-hand diagram of Figure 17, and $a'' \doteq D \circ r'$. We now take the pushout (L, F) for (a, r), and properties of pushouts then yield the right-hand diagram, in which the upper square is also a pushout. So there is a transition $a \xrightarrow{L} a'$, where $a' \doteq F \circ r'$; note also that $a'' \doteq C' \circ a'$. 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 $\mathscr{L}_{\mathsf{p}}$. 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 18. 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 $(a'', b'') \in \mathscr{S}^{\doteq}$.

If L is a non-identity label, we consider the anatomy of the transition $a \xrightarrow{L} a'$, as exemplified in Figure 15. 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 $\mathscr{L}_{\mathsf{p}}$ there is a sequence of transitions

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

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 $\mathscr{L}_{\mathsf{g}}$ 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 18. Also, it has an underlying reaction rule (s, s'), with $b' \doteq G \circ s'$.

Now we form the right-hand diagram of Figure 18 by replacing this pushout for the lower square in the right-hand diagram of Figure 17. Since both small squares are pushouts, the large square is also, and thus it underlies an $\mathscr{L}_{\mathsf{g}}$-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 $\mathscr{S}^{\hat{=}}$. We already know that $a'' \mathrel{\hat{=}} C' \circ a' = C' \circ \overline{L} \circ \overline{a}$. We can now compute

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

and hence $(a'', b'') \in \mathscr{S}^{\hat{=}}$ 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 4.16 (coincidence of concrete bisimilarities). In $\acute{\mathbf{C}}\mathbf{E}$ the two bisimilarities $\sim_{\mathsf{g}}$ and $\sim_{\mathsf{p}}$ for concrete s-nets coincide. Hence, since $\sim_{\mathsf{g}}$ is a congruence, $\sim_{\mathsf{p}}$ is a congruence too.

It just remains to transfer this to abstract s-nets. But this is immediate by Proposition 4.10 and Corollary 4.13, and we have finally arrived at the result we set out to prove.

Corollary 4.17 (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 is $\sim_{\mathsf{p}}$.

It is worth noting that since $\mathscr{L}_{\mathsf{p}}$ and $\sim_{\mathsf{p}}$ were defined without reference to link graphs, it was not clear which contexts would preserve $\sim_{\mathsf{p}}$, that is, 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 $\sim_{\mathsf{p}}$.

5. Related and future research

We conclude by commenting on some related work that has not already been mentioned in the text, and at the same time 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 the second author (Jensen and Milner 2004) pursues a similar programme for full bigraphs, giving a full analysis of a finite asynchronous π-calculus as reported earlier at a conference (Jensen and Milner 2003). Jensen's forthcoming Ph.D. Dissertation (Jensen 2006) will carry out this analysis not only for the full π-calculus but also for the ambient calculus.

The first author, in his Ph.D. Dissertation (Leifer 2001), 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 the first author's treatment of weak transitions.

The long tradition of graph-rewriting is based on the *double pushout* (DPO) construction originated by Ehrig (Ehrig 1979). 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 the arrows are graphs. But there are links between these formulations, both via cospans (Gadducci 1999) and via a categorical isomorphism between graph embeddings and a coslice over our contextual s-categories (Cattani *et al.* 2002). Ehrig has investigated these links further (Ehrig 2002). This led to the paper Ehrig and König (2004), which was further generalised by Sassone and Sobocinski (2004), in which the RPO technique is transferred to graph-embedding categories.

The bisimilarity based on the full transition system has been investigated in Montanari and Sassone (1992), and has the advantage of a simple characterisation: it is the coarsest congruence that is also a bisimulation. Sewell has argued that this approach is unsatisfactory (Sewell 1998), and we agree with his view; we also aimed to achieve an intuitively acceptable bisimilarity that is based on a smaller TS than the full one. To our knowledge, the question of how to reduce significantly the full TS while retaining the bisimilarity that it induces remains open.

Sassone and Sobocinski have generalised RPOs to *groupoid* RPOs, in a 2-category whose 2-cells (that is, arrows between arrows) are isomorphisms (Sassone and Sobocinski 2002). 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, for example, the characterisation of all IPOs for a given span (Theorem 3.18).The 2-categorical approach clearly deserves further development; but until it can be shown to improve the treatment of results such as the characterisation of IPOs and the adequacy theorem, there is good reason to retain the present approach.

The idea of finding conditions under which a labelled transition relation yields an operational congruence has been thoroughly studied in work on *structural operational semantics* (SOS) (Groote and Vaandrager 1992). The principle is to postulate *rule formats*, which are conditions on an inductive presentation of a labelled transition relation that ensure that operational equivalences are congruences. The work on SOSs takes almost the opposite approach to the one we consider here by postulating that a labelled transition relation is already given and then showing that if it satisfies a particular format *then* the congruences follow. By contrast, we regard reaction rules as primitive, not labelled transitions, and aim to synthesise a labelled transition relation from the reaction rules for which operational equivalences are congruences.

The idea that the labels of a process can be those contexts that enable reactions has been a longstanding intuition in the field. It appeared formally in Abramsky's applicative bisimilarity (Abramsky and Ong 1993) and was the basis of Bernstein's work on rule formats (Bernstein 1998) and Jeffrey and Rathke's LTS for a variant of the λ-calculus with fresh name creation (Jeffrey and Rathke 1999).

The tile model (Gadducci and Montanari 2000) generalises rule formats by using a rich 2-categorical structure to describe the composition of the parts of an LTS derivation.

By taking the appropriate notion of *tile dynamic bisimulation* to close up by context composition, it is possible to recover open bisimilarity in the π-calculus (Bruni *et al.* 2005) as well as giving a semantics to logic programming (Bruni *et al.* 2001).

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 (Lawvere 1963), the standard categorical treatment of algebra. In this spirit, the second author has completely axiomatised the algebra of pure bigraphs (Milner 2004b).

The case-studies on deriving transitions from reaction rules, in both the π-calculus and Petri nets, have shown an interesting mismatch between these derived transitions and 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 to arrive 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 a kind of spatio-temporal logic (Caires and Cardelli 2001; Caires 2004). 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.

Appendix A. Proofs

Lemma 3.12. $(\vec{B}, B)$ is a candidate RPO for $\vec{A}$ relative to $\vec{D}$.

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

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

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

Case: $p \in W \uplus P_2$, $A_0(p) = e \in E_0 - E_2$.
In this case $(B_0 \circ A_0)(p) = A_0(p) = e$. 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 $x_1 \notin X_1'$, and thus $(B_1 \circ A_1)(p) = B_1(x_1) = D_1(x_1) = e$.

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

Case: $p \in W \uplus P_2$, $A_0(p) = x_0 \in X'_0$.
In this case $D_0(x_0) \in E_3 \uplus Z$, 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: $p \in W \uplus P_2$, $A_0(p) = x_0 \in X_0 - X'_0$.
In this case $D_0(x_0) = e \in E_1 - E_2$. 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: $x \in X'_0$.
In this case $(B \circ B_0)(x) = B(\widehat{0,x}) = D_0(x)$.

Case: $x \in X_0 - X'_0$.
In this case $B_0(x) = D_0(x) \in E_0 - E_2$, so $(B \circ B_0)(x) = D_0(x)$.

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 can 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: $p \in P_3$.
In this case $(B \circ B_0)(p) = B(p) = D_0(p)$. □

Theorem 3.13 (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 3.10 yields such an RPO.

Proof. We have already proved that the triple $(\vec{B}, B)$ built in Construction 3.10 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.

$$
\begin{tikzcd}
 & & Z & & \\
 & & Y \arrow[u, "C"'] & & \\
X_0 \arrow[uurr, "D_0"] \arrow[urr, "C_0"] \arrow[rr, "B_0"'] & & \hat{X} \arrow[uu, bend left, "B"] \arrow[u, dashed, "\widehat{C}"'] & & X_1 \arrow[uull, "D_1"'] \arrow[ull, "C_1"'] \arrow[ll, "B_1"] \\
 & & {} \arrow[ull, "A_0"] \arrow[urr, "A_1"'] & &
\end{tikzcd}
$$

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

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

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: $x \in X'_0$.
In this case $(\widehat{C} \circ B_0)(x) = \widehat{C}(\widehat{0,x}) = C_0(x)$.

Case: $x \in X_0 - X'_0$.
In this case $D_0(x) \in E_1 - E_2$, so $B_0(x) = D_0(x)$, and thus $(\widehat{C} \circ B_0)(x) = D_0(x)$. 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 $B_0(p) = D_0(p)$, and we then get $(\widehat{C} \circ B_0)(p) = D_0(p)$. Also, $C_0(p) = (C \circ C_0)(p) = D_0(p)$.

Case: $p \in P_1 - P_2$, $D_0(p) \in E_3 \uplus Z$.
In this case $A_1(v) = x \in X'_1$ 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: $p \in P_4$.
In this case $(\widehat{C} \circ B_0)(p) = \widehat{C}(p) = C_0(p)$.

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

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

Case: $\hat{x} = \widehat{0,x} \in X$, $B(\hat{x}) \in E_5 \uplus Z$.
In this case $D_0(x) = B(\hat{x}) \in E_5 \uplus Z$, 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: $p \in P_4$, $B(v) \in E_4$.
In this case $(C \circ \widehat{C})(p) = \widehat{C}(p) = C_0(p) = D_0(p) = B(p)$.

Case: $p \in P_4$, $B(p) \in E_5 \uplus Z$.
In this case $B(p) = D_0(p) = C(y)$, 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: $p \in P_5$.
In this case $(C \circ \widehat{C})(p) = C(p) = D_0(p) = B(p)$.

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

Appendix B. Commentary

A tractable definition of link graphs is crucial to the development of bigraphs. There is at least one alternative definition, which was adopted in the initial formulation of link graphs (Milner 2001b). (In that paper they were called monographs.) Here we compare that definition with the present one, explaining why we have adopted the latter.

The reader may note that the present definition is not self-dual; that is, inner and outer names are not treated symmetrically, so the inversion of a link graph is no longer a link graph. The key point is that each link may contain many inner names, but at most one outer name, which is the image under the link map of each point in the link.

The earlier definition of link graphs (Milner 2001b) is indeed self-dual. Instead of using a link map, it defines the links of $G : X \to Y$ to be the cells of a partition of the set $X + P + Y$, where P are the ports of G. In other words, the link map is replaced by an equivalence relation over ports, inner names and outer names, and a link is an equivalence class. If $H : Y \to Z$ has ports Q, then the link equivalence for $H \circ G$ is the smallest equivalence over $X + P + Y + Q + Z$ that includes the link equivalences of G and H. This easily leads to an s-category. Moreover, in this s-category there is no role for the set E of edges (closed links) here associated with a link graph, so the support of a link graph with the earlier definition is just the set of its nodes.

Much of the theory presented here was developed for the earlier version of link graphs by the second author, who thought, as the reader may also, that it was the obvious and best formulation. Equivalence relations are well understood and there are well-known techniques for manipulating them. Two factors, however, led to the present formulation. The first is that proofs using equivalence relations turn out to be significantly more complex than those for link maps. To measure this, the reader may like to compare the work of Section 2.3 and Appendix A here with the corresponding work for the earlier definition of bigraphs in Section 4 and Appendices B and C in Milner (2001b). Apparently, for our purposes, maps are easier than equivalences!

The second factor against the earlier definition of link graphs is that RPOs do not always exist, as they do in the present formulation. They do indeed exist for a pair $\vec{A}$ with bound $\vec{D}$, provided that at least one of the arrow $\vec{A}$ is an epimorphism. Now, a link graph with the earlier definition is epi iff it has no idle names and no aliases (an alias is a link with two or more outer names). Figure 15 of Milner (2001b) provides a counter-example showing the lack of an RPO when neither of $\vec{A}$ is epi. Thus, by admitting aliases (which we must if link graphs are to be self-dual), we also admit the partial lack of RPOs.

This second factor is not crippling, since when we use RPOs to generate labelled transition systems one of the pair $\vec{A}$ is always a redex, and there are good arguments for excluding reaction rules with non-epi redexes. However, we expect other uses of RPOs to arise, since an RPO merely exhibits the excess of one member of a span $\vec{A}$ over the other, and we may require this analysis more generally.

On balance, the combination of heavy proofs and only partial existence of RPOs has led us to prefer the present formulation. In making this choice we certainly exclude some possible applications: for example, aliases correspond to the explicit fusions introduced in Gardner (2000). However, the range of applications is still very large, and we are keen to explore this range with the most tractable conceptual tools.

Appendix C. No RPOs in abstract link graphs

Abstract link graphs, which were introduced in Definition 3.23, consist of lean-support equivalence classes of concrete link graphs; that is, they forget the identity of nodes and edges. As promised in Section 2.3 and at the end of Section 3.2, we now provide an example to show that RPOs do not always exist in abstract link graphs.

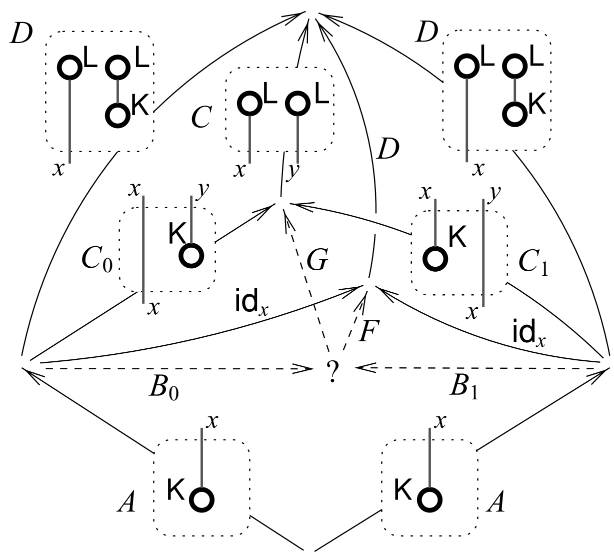


Fig. 19. A case where no RPO exists for abstract link graphs

Figure 19 shows a bound (D, D) for a pair (A, A), where $A : \varnothing \to x$ and $D : x \to \varnothing$. (We are writing $\{x\}$ as x.) It also shows two relative bounds for the pair, relative to (D, D). The first of these is $(\mathsf{id}_x, \mathsf{id}_x, D)$; the second, $(\vec{C}, C)$, involves the creation of a new K node. Thus the two relative bounds treat nodes differently: intuitively, the first regards the K-nodes of the pair (A, A) as identical, while the second regards them as distinct. The large square commutes in the category Lɪɢ of abstract link graphs only because an automorphism of the composite link graph $D \circ A$ is treated as an identity.

We now show that no RPO can exist for (A, A) to (D, D). We suppose $(\vec{B}, B)$ is such an RPO, and derive a contradiction from the assumption that mediating arrows F and G exist to our two relative bounds. (The upper component B is not shown in the figure as the argument does not require it.) Since $F \circ B_0 = \mathsf{id}_x$, we have $B_0(x) = z$ and $F(z) = x$ for some z. Since $B_0 \circ A = B_1 \circ A$, we also have $B_1(x) = z$. Since $G \circ B_i = C_i$ $(i = 0, 1)$, we deduce that $G(z) = x = y$, which gives a contradiction since the link map of G is single-valued. Thus RPOs do not exist in general in abstract link graphs.

Acknowledgements

The authors wish to thank the anonymous referees for their detailed reading and helpful comments, which have led to many clarifications and improvements in this paper.

References

Abramsky, S. and Ong, C.-H. L. (1993) Full abstraction in the lazy lambda calculus. *Information and Computation* **105** (2) 159–267.

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. *Springer-Verlag Lecture Notes in Computer Science* **2154** 502–518.

Bergstra, J. A. and Klop, J. W. (1985) Algebra for communicating processes with abstraction. *Theoretical Computer Science* **37** 77–121.

Bernstein, K. (1998) A congruence theorem for structured operational semantics of higher-order languages. *Proc. LICS'98, 13th Annual IEEE Symposium on Logic in Computer Science* 153–164.

Berry, G. and Boudol, G. (1992) The chemical abstract machine. *Theoretical Computer Science* **96** 217–248.

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. *Springer-Verlag Lecture Notes in Computer Science* **1639** 344–363.

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

Brookes, S. D., Hoare, C. A. R. and Roscoe, A. W. (1984) A theory of communicating sequential processes. *J. ACM* **31** 560–599.

Bruni, R., Gadducci, F. and Montanari, U. (2002) Normal forms for algebras of connections. *Theoretical Computer Science* **286** (2) 247–292.

Bruni, R., Montanari, U. and Rossi, F. (2001) An interactive semantics of logic programming. *Theory and Practice of Logic Programming* **1** (6) 647–690.

Bruni, R., Montanari, U. and Sassone, V. (2005) Observational congruences for dynamically reconfigurable tile systems. *Theoretical Computer Science* **335** (2-3) 331–372.

Cardelli, L. (2004) Bioware languages. In: *Computer Systems: Theory, Technology, and Applications — A Tribute to Roger Needham*, Springer-Verlag 59–65.

Caires, L. and Cardelli, L. (2001) A spatial logic for concurrency (Part I). Proc. 4th International Symposium on Theoretical Aspects of Computer Software. *Springer-Verlag Lecture Notes in Computer Science* **2215** 1–37.

Caires, L. (2004) Behavioural and spatial observations in a logic for the π-calculus. Proc. FOSSACS 2004. *Springer-Verlag Lecture Notes in Computer Science* **2987** 72–89.

Cardelli, L. and Gordon, A. D. (2000) Mobile ambients. Foundations of System Specification and Computational Structures. *Springer-Verlag Lecture Notes in Computer Science* **1378** 140–155.

Castellani, I. (2001) Process algebras with localities. In: *Handbook of Process Algebra*, Elsevier 947–1045.

Cattani, G. L., Leifer, J. J. and Milner, R. (2002) Contexts and embeddings for closed shallow action graphs. *Formal Aspects of Computing* **13** (3–5).

Drewes, F., Hoffmann, B. and Plump, D. (2000) Hierarchical graph transformation. In: Foundations of Software Science and Computation Structures. *Springer-Verlag Lecture Notes in Computer Science* **1784**.

Ehrig, H. (1979) Introduction to the theory of graph grammars. Graph Grammars and their Application to Computer Science and Biology. *Springer-Verlag Lecture Notes in Computer Science* **73** 1–69.

Ehrig, H. (2002) Bigraphs meet double pushouts. *EATCS Bulletin* **78** 72–85.

Ehrig, H. and König, B. (2004) Deriving bisimulation congruences in the DPO approach to graph-rewriting. Proc. FOSSACS 2004. *Springer-Verlag Lecture Notes in Computer Science* **2987** 151–156.

Fournet, C. and Gonthier, G. (1996) The reflexive Cham and the join calculus. *Proc. 23rd Annual ACM Symposium on Principles of Programming Languages, Florida* 372–385.

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). *Electronic Notes in Theoretical Computer Science* **29**.

Gadducci, F. and Montanari, U. (2000) The tile model. In: Plotkin, G., Stirling, C. and Tofte, M. (eds.) *Proof, Language and Interaction: Essays in Honour of Robin Milner*, MIT Press 133–166. (Also Technical Report TR-27/96, Dipartimento di Informatica, Università di Pisa, 1996).

Gardner, P. A. (2000) From process calculi to process frameworks. *Proc. CONCUR 2000, 11th International Conference on Concurrency Theory* 69–88.

Gardner, P. A. and Wischik, L. (2000) Explicit fusions. Proc. MFCS 2000. *Springer-Verlag Lecture Notes in Computer Science* **1893** 373–383.

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

Hasegawa, M. (1999) *Models of sharing graphs (a categorical semantics of let and letrec)*, Ph.D. Dissertation, Division of Informatics, Edinburgh University. (Also available as Technical Report ECS-LFCS-97-360, and in Springer Series of Distinguished Dissertations in Computer Science.)

Hirsch, D. and Montanari, U. (2001) Synchronised hyperedge replacement with name mobility. Proc. 12th International Conference on Concurrency Theory (CONCUR 2001). *Springer-Verlag Lecture Notes in Computer Science* **2154** 121–136.

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

Honda, K. and Tokoro, M. (1991) An object calculus for asynchronous communications. ECOOP'91, Workshop on Object-based Concurrent Programming. *Springer-Verlag Lecture Notes in Computer Science* **512**.

Jeffrey, A. and Rathke, J. (1999) Towards a theory of bisimulation for local names. *Proc. LICS 1999*, IEEE Press 56–66.

Jensen, O. H. (2006) Forthcoming Ph.D. Dissertation.

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

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`.

Lafont, Y. (1990) Interaction nets. *Proc. 17th ACM Symposium on Principles of Programming Languages (POPL'90)* 95–108.

Lawvere, F. W. (1963) *Functorial semantics of algebraic theories*, Ph.D. Dissertation, Columbia University. (Announcement in *Proc. Nat. Acad. Sci.* **50** 869–873.)

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

Leifer, J. J. and Milner, R. (2000) Deriving bisimulation congruences for reactive systems. *Proc. CONCUR 2000, 11th International Conference on Concurrency theory* 243–258. (Available at `http://pauillac.inria.fr/~leifer`.)

Merro, M. and Hennessy, M. (2002) Bisimulation congruences in safe ambients. *Proc. 29th International Symposium on Principles of Programming Languages, Oregon* 71–80.

Milner, R. (1980) A Calculus of Communicating Systems. *Springer-Verlag Lecture Notes in Computer Science* **92**.

Milner, R. (1996) Calculi for interaction. *Acta Informatica* **33** 707–737.

Milner, R. (2001a) Computational flux. *Proc. 28th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* 220–221.

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

Milner, R. (2001c) Bigraphical reactive systems. Proc. 12th International Conference on Concurrency Theory. *Springer-Verlag Lecture Notes in Computer Science* **2154** 16–35.

Milner, R. (2004a) Bigraphs for Petri nets. Proc. Advanced Course on Petri Nets, Eichstätt 2003. *Springer-Verlag Lecture Notes in Computer Science* **3098**.

Milner, R. (2004b) Axioms for bigraphical structure. Technical Report 581, University of Cambridge Computer Laboratory. (To appear in *Mathematical Structures in Computer Science*.)

Milner, R. (2005) Pure bigraphs. Technical Report 614, University of Cambridge Computer Laboratory.

Milner, R., Parrow, J. and Walker, D. (1992) A calculus of mobile processes, Parts I and II. *Information and Computation* **100** 1–77.

Montanari, U. and Sassone, V. (1992) Dynamic congruence vs. progressing bisimulation for CCS. *Fundamenta Informaticae* **16** 171–196.

Nielsen, M., Priese, L. and Sassone, V. (1995) Characterizing behavioural congruences for Petri nets. Proc. CONCUR'95. *Springer-Verlag Lecture Notes in Computer Science* **962** 175–189.

Park, D. M. R. (1980) Concurrency and automata on infinite sequences. *Springer-Verlag Lecture Notes in Computer Science* **104**.

Pomello, L., Rozenberg, G. and Simone, C. (1992) A survey of equivalence notions for net-based systems. In: Advances in Petri Nets '92. *Springer-Verlag Lecture Notes in Computer Science* **609** 410–472.

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

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

Priami, C. (1995) Stochastic π-calculus. *Computer Journal* **38** (7) 578–589.

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

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. *Springer-Verlag Lecture Notes in Computer Science* **2263** 435–449.

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

Sassone, V. and Sobocinski, P. (2004) Congruences for contextual graph-rewriting. BRICS Technical Report RS-04-11.

Sewell, P. (1998) From rewrite rules to bisimulation congruences. Proc CONCUR'98. *Springer-Verlag Lecture Notes in Computer Science* **1466** 269–284. (Full version in *Theoretical Computer Science* **272** (1–2).)

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)* **6** 459–470.

Smith, H. and Fingar, P. (2002) *Business Process Management: the third wave*, Meghan-Kiffer Press.

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