

Bigraphs for Petri Nets

Robin Milner

University of Cambridge, The Computer Laboratory,
J J Thomson Avenue, Cambridge CB3 0FD, UK

Abstract. A simple example is given of the use of bigraphical reactive systems (BRSs). It provides a behavioural semantics for condition-event Petri nets whose interfaces are named condition nodes, using a simple form of BRS equipped with a labelled transition system and its associated bisimilarity equivalence. Both of the latter are derived from the standard net firing rules by a uniform technique in bigraphs, which also ensures that the bisimilarity is a congruence. Furthermore, this bisimilarity is shown to coincide with one induced by a natural notion of *experiment* on condition-event nets, defined independently of bigraphs.

The paper is intended as a bridge between Petri net theory and bigraphs, as well as a pedagogical exercise in the latter.

1 Introduction

This paper conducts a simple exercise in bigraphical reactive systems (BRSs) [4], consisting of a behavioural study of condition-event Petri nets [12]. The exercise has two very different purposes. The first is pedagogical: condition-event nets can be modelled as a *link-graph* reactive system (LRS), which is a simple form of BRS, so they illustrate the use of bigraphs while avoiding some of their complexity. The other purpose is to promote future research: since bigraphs model systems that can reconfigure both their placing and their linking, the exercise illustrates a framework in which Petri nets may be generalised to deal with mobile informatic systems.

The exercise involves the interpretation of condition-event nets in terms of *bisimilarity* [8]. As in process calculi, it may sometimes be useful to employ an abstract model of the behaviour of a Petri net in which two nets are regarded as equivalent if they cannot be distinguished by certain forms of *experiment*. If an experiment e can be carried out on a system in state g , changing its state to g' , we write

$$g \xrightarrow{e} g'$$

and call it a *labelled transition* between the two states. If we fix a vocabulary of labels e and define the possible transitions for each one, we have a (*labelled*) *transition system* (TS) \mathcal{L} . Then a symmetric binary relation \mathcal{R} between two system states is said to be a *bisimulation* (for \mathcal{L}) if

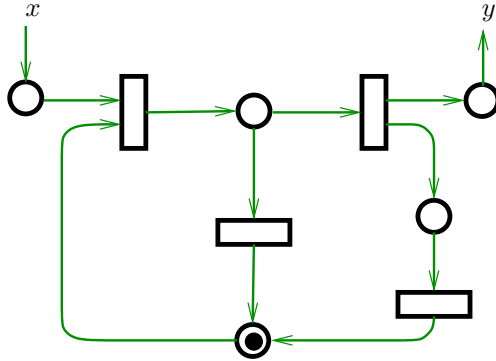
whenever $f\mathcal{R}g$ and $f \xrightarrow{e} f'$, there exists
a state g' such that $g \xrightarrow{e} g'$ and $f'\mathcal{R}g'$.

In other words: given two related processes, whatever one of them can do, the other can also do without losing the relationship. The bisimulation property is preserved by union

of relations, so there is a largest bisimulation which is the union of all bisimulations, and it is easily found to be an equivalence relation. We call it *bisimilarity* (for \mathcal{L}).

All this holds for any interpretation of ‘experiment’. We call bisimulations (and bisimilarity) *weak* or *strong*, and denote the equivalence by \approx or \sim , according to whether or not a single experiment e can be accompanied by any finite amount of internal activity. In the strong case we consider each individual internal action as an experiment, even though it is indistinguishable from any other such action. Both weak and strong bisimilarity abstract away from the causal behaviour of systems, but the weak form is more generous in turning a blind eye to internal activity.

We now consider what might be an experiment on a condition-event net. There are various ways to make parts of a net externally accessible, in order to observe – or induce – behaviour of the net from the outside. Authors (some of whom are cited in the next section) have considered making accessible certain *events* or *actions*, or alternatively certain *conditions* or *states*. This exercise is of the latter kind; we allow an experimenter to change certain conditions from holding to not holding or vice versa, by removing or adding a token. This choice was made because it makes the exercise simple, but the alternatives may well yield to a similar approach.



Consider the above net, for example. At different times the experimenter will be able to add or remove a token at x or at y . In general, given a state g , i.e. a *marking* of the net, the transition $g \xrightarrow{+x} g'$ or $g \xrightarrow{-x} g'$ 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, $g \xrightarrow{\tau} g'$, represents an internal event involving no external participation.

These three kinds of transition are the basis of a TS; we shall call it \mathcal{L}_p , and its induced bisimilarity \sim_p . In the rest of this paper we shall compare this TS and its bisimilarity with another one, which arises from setting up condition-event nets as an LRS and then deriving a TS \mathcal{L}_g by a construction [5, 4] that is uniform over all LRSs (and BRSs). We shall find that the labels of \mathcal{L}_g differ from those of \mathcal{L}_p , but that the two bisimilarities \sim_p and \sim coincide. This gives us confidence that the dynamics of nets may be faithfully presented in bigraph theory.

2 Related Work on Petri Nets

In the introduction we declared two goals: first, to give a simple tutorial in bigraphs; second, to treat Petri nets in the bigraphical framework, thus perhaps easing the extension of the net model to admit mobility. We shall tackle both goals by means of a simple case study. In this section we briefly describe how the study relates to existing work in Petri net behaviour, with reference to some recent papers on that topic.

Pomello, Rozenberg and Simone [10] give a comprehensive survey of behavioural equivalences for Petri nets. They cover those based on observation both of actions and of states, and range from fine equivalences respecting causality to coarser ones, for example the failures equivalence from CSP, the coarsest which respects deadlock. The study of congruence is reported as being rather incomplete at that date (1992).

Nielsen, Priese and Sassone [9] characterise some behavioural congruences on nets. Given semantic function \mathcal{B} that assigns an abstract behaviour to each net, they consider the congruence \approx it induces upon nets; this is defined by

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

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

Priese and Wimmel [11] continue this programme; they enrich the net combinators, and consider a wider range of semantic functions.

The Petri Box calculus of Best, Devillers and Hall [1], like the previous two, 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.

This brief summary does not do justice to the four papers, which represent well the progress towards a modular treatment of Petri nets. But it helps us to identify differences with bigraph theory, which suggest contributions that can be made by the latter. The first difference is that, since bigraphs and their contexts are the arrows of a category, whenever a class of agents (e.g. nets) is encoded in bigraphs the contexts and combinators are already 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.

In this brief discussion we have tried to explain the way in which bigraphs aim at a theory shared by different models of concurrency. Much work is needed to determine how far they can achieve this aim. Success can be measured in two ways: by the range of different models that can be satisfactorily treated in bigraphs, and by the depth of the theory thus shared among them. The present paper begins to evaluate these measures with particular reference to Petri nets.

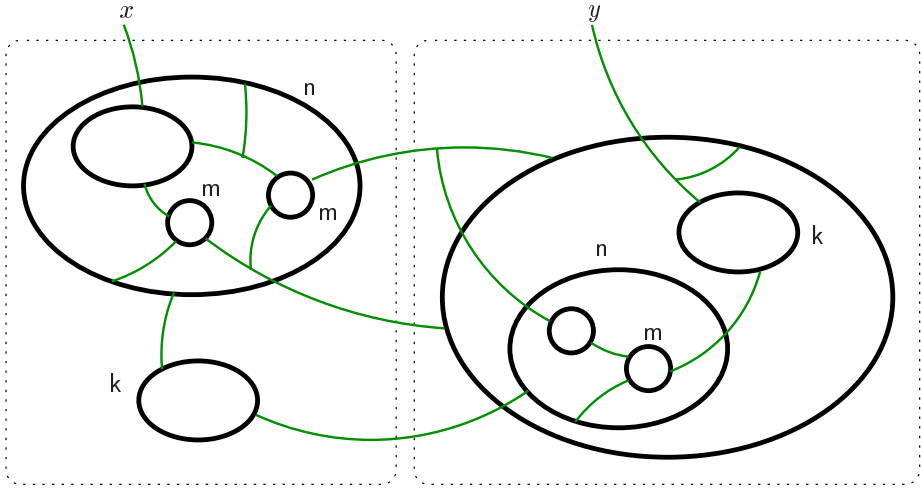


Fig. 1. A typical bigraph

3 Bigraphs and Link Graphs

Figure 1 shows a typical bigraph. The ovals and circles are *nodes*. Associated with each node is a *control* which indicates what kind of node it is. Here we show three controls, *k*, *m* and *n*; the controls of other nodes are not shown. Each control has an *arity*, a finite ordinal indicating the number of *ports* on that kind of node; here *k*, *m* and *n* have arity 2, 3 and 2 respectively.

A bigraph is so called because its nodes are structured in two ways. The first structure is *placing*; the nodes are nested inside one another, giving an ordered set of trees, i.e. a forest. In our example there are two trees; each has a *root* – not itself a node – represented by a dotted rectangle. The second structure is *linking*; the ports of the bigraphs are partitioned into *links*, shown by curved lines. A link may be *open* or *closed*; each open link has a distinct *name* (here *x* or *y*). Names allow bigraphs to be joined via their open links.

The two structures are totally independent; note here how the links cross node boundaries and even link different trees in the forest.

In other applications of bigraphs the nesting of nodes plays an important role in the way bigraphs reconfigure themselves; both placing and linking may vary dynamically. But in our present application the placing vanishes, so we shall work only with *link graphs*, i.e. the linkage structure. In following sections we shall explain only those parts of link-graph theory that we need.

Link graphs. It is common in graph theory to distinguish between *concrete* and *abstract* graphs. In the former the nodes and edges have identity, and we distinguish two graphs that differ only by a bijection between their nodes and edges; in the latter we equate them. For link graphs we are interested in both kinds; for applications we usually want the abstract ones, but the concrete ones provide us a convenient means to develop the

theory. Here we shall work mainly with the concrete link graphs; at the end we point out how the results, once derived, transfer to the abstract ones.

We treat concrete link graphs, then, as the morphisms of a *supported precategory*. This is like a category except that each morphism has a finite set, its *support*, and the composition of two morphisms is defined only if their supports are disjoint. The support of a composite morphism is the union of the supports of its components. Identity morphisms have empty support. Two morphisms F and F' are *support equivalent*, written $F \simeq F'$, if they differ only by a bijection between their supports.

Working with supported precategories is hardly different from working with categories; in this paper the reader can rest assured that any concept familiar from the latter means practically the same for the former. More discussion of this point can be found in the concluding section.

In the supported precategory of (concrete) link graphs, the objects are finite sets X, Y, \dots of *names*. A link graph $H : X \rightarrow Y$ has *inner face* X and *outer face* Y . An example appears in Figure 2; think of H as a *context* in which to embed a link graph G with outer face X . The *points* of a link graph are its ports and its inner names, so H has eleven points: three ports for each a-node, two for the b-node and three inner names $X = \{x_1, x_2, x_3\}$. (Note that the points do not include the *outer* names.) The *links* constitute a partition of the points, and to each *open* link (which we mentioned already) is assigned a distinct outer name; for H , these are $Y = \{y_1, y_2\}$. Note that H has two open and three closed links.

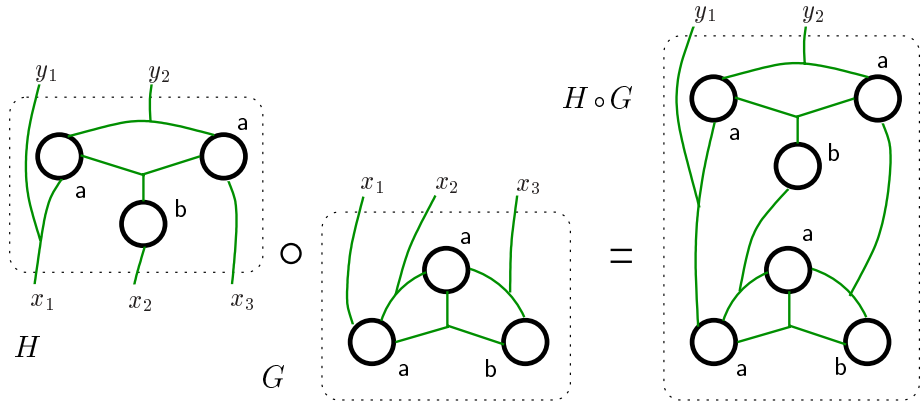


Fig. 2. The composite $H \circ G$ of link graphs $G : \emptyset \rightarrow X$ and $H : X \rightarrow Y$

The support of a link graph consists of its nodes and its closed links (the latter corresponding to the edges of a classical graph). Their identity is not shown in the diagram, but when we show a composition of two link graphs we assume disjoint supports. Figure 2 shows the composition of $G : \emptyset \rightarrow X$ and $H : X \rightarrow Y$; each open link in G is joined to the link in H that contains the corresponding inner name, and then that name is erased. The outer and inner names of a link graph need not be disjoint. The identi-

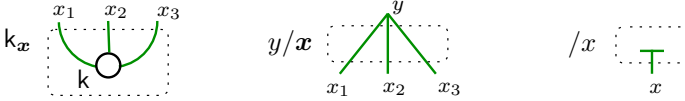
ties are those link graphs $\text{id}_X : X \rightarrow X$ with empty support in which each outer name $x \in X$ is assigned to the link whose only point is the inner name x .

A link graph with empty inner face, such as G in the diagram, is called *ground*; ground link graphs, and more generally ground bigraphs, are used to represent *agents*, such as a condition-event net with no missing pieces. We typically use lower case letters f, g, h, \dots for ground link graphs.

Algebra. Let us review briefly how complex link graphs may be built from simpler ones. As well as composition, we use *tensor product*: if F_1 and F_2 have disjoint supports, disjoint inner faces X_1 and X_2 , and also disjoint outer faces Y_1 and Y_2 , then their tensor product

$$F_1 \otimes F_2 : X_1 \cup X_2 \rightarrow Y_1 \cup Y_2$$

is formed by placing them side-by-side. The unit for \otimes is just id_\emptyset . Using composition and tensor product we can build all link graphs from the *atomic* ones (those with a single node) with the help of *wirings* (those with no nodes). If k is a control with arity n and x a sequence of n distinct names then a k -atom with ports named x_1, \dots, x_n is denoted by k_x . All wirings can be built from two elementary kinds: a *linker* y/x and a *closure* $/x$. These three elementary link graphs are as follows, when $x = x_1x_2x_3$:



For example, suppose the outer face of F is $\{xyz\}$. We may want to replace x and y by v , leaving z unaffected; or we may want to do the same but close off the link z . In each case we can form $\omega \circ F$, where the wiring ω is respectively

$$\omega = v/xy \otimes \text{id}_z \quad \text{or} \quad \omega = v/xy \otimes /z.$$

More generally, the algebra of link graphs consists of expressions built from the elements using \circ , \otimes and identities, and satisfying some simple equations. In this paper we shall use a little algebra, but rely more upon diagrams.

One abbreviation will come in handy. If F has outer face $\{xyz\}$ and G has inner face $\{xy\}$, then we may write $G \circ F$ instead of $(G \otimes \text{id}_z) \circ F$. In other words, we sometimes omit identities in composition when no confusion arises.

Sorting. For many purposes, it is useful to enrich link graphs by imposing a *sorting*, i.e. a discipline of *sorts* (or *types*). We set up a sorting in three stages:

1. Specify a set $S = \{\alpha, \beta, \dots\}$ of sorts.
2. Declare for each control with arity n an ordered list of n sorts. This determines a sort for every port in a link graph.
3. Enrich interfaces X, Y, \dots by assigning a sort to each name.

We may then define a *well-sorted* link graph to be one that satisfies certain constraints. Here we are interested especially in *many-one* sorting, in which there are just two sorts α and β . Each link may have any number of α -points, but β -points are constrained follows:

- A closed link has exactly one β -point;
- An open link with a β -name has exactly one β -point;
- An open link with an α -name has no β -points.

As an example, for link graphs with controls a and b we may declare that every port of an a -node has sort α , and every port of an b -node has sort β . It can be checked for Figure 2 that G , H and $H \circ G$ are well-sorted if X has the sorting $\{x_1: \alpha, x_2: \alpha, x_3: \beta\}$ and Y has the sorting $\{y_1: \alpha, y_2: \alpha\}$. The reader may like to look ahead and see the rôle of sorting in representing condition-event nets; it ensures that each port on an event node will be connected to at most one pre- or post-condition node.

Dynamics. To equip link graphs with behaviour, we first specify a subclass of the interfaces called *agent interfaces*. If X is such an interface we call any $f: \emptyset \rightarrow X$ an *agent*; these are the link graphs whose behaviour we want to define. For this purpose we specify a set of *reaction rules*, each being a pair (r, r') of ground link graphs with the same outer face. In each rule we call r the *redex* and r' the *reactum*. Then we specify the *reaction relation* \longrightarrow over agents to be the smallest such that

$$D \circ r \longrightarrow D \circ r'$$

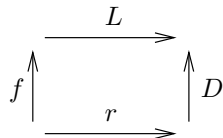
for every reaction rule (r, r') and every context D for which the compositions are defined and are agents. We also require both the rule-set and the reaction relation to be closed under support-equivalence. In the next section we shall set up the firing rules of condition-event nets as reaction rules.

What we have defined so far is called a *reactive system*. In process calculi it has become usual to refine this to a (*labelled*) *transition system* (TS), with transitions of the form $f \xrightarrow{\ell} f'$, where the *labels* ℓ are specific to each calculus. Intuitively, ℓ represents the contribution that f may make to a reaction; typically this contribution is incomplete, so the transition makes precise the idea that both an agent and its environment may contribute to a reaction. In terms of these TSs, one may define bisimilarity and other equivalences and preorders over agents; a test of a good TS is that these behavioural relations are *congruential*, i.e. preserved by insertion into any context.

In bigraph theory we adopted a proposal by Leifer and Milner [5] to derive TSs uniformly over all bigraphical reactive systems, in a way that guarantees congruential behavioural relations. For link graphs, it works as follows. We consider a label L to be a (link graph) context into which an agent may be inserted in order to enable a reaction to occur; that is, we define the transition

$$f \xrightarrow{L} f'$$

to mean that the equation $L \circ f = D \circ r$ holds for some context D and reaction rule (r, r') , and moreover that $f' \simeq D \circ r'$. Think of this as inserting f into a (small) context L so that an instance of the redex r occurs in the composite $L \circ f$, and then replace this occurrence by r' .

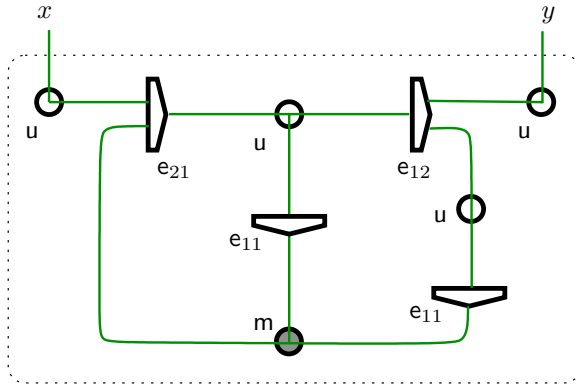


But if we were to allow *all* such contexts as labels L , there would be an unwieldy multitude of labels. Indeed, a moment's reflection reveals that if L is a transition label, then so would be *any larger context* $C \circ L$! To avoid this, the theory limits labels L to be the (in some sense) *smallest* for which the equation $L \circ f = D \circ r$ holds for some D , given f and r . It turns out that strong bisimilarity is congruential for any TS so defined, and we believe that this extends to other behavioural relations. By *smallest*, we mean that the above diagram should not only commute but should also be an *idem pushout* (IPO), a weaker version of the more familiar *pushout*.

We need not explain IPOs here, because our precategory of condition-event nets actually has pushouts where we need them. We shall not show how to construct pushouts for link graphs; we shall just exhibit the resulting TS and then work with it. The construction can be found in the Technical Report by Jensen and Milner [4]. We should note that pushouts – even IPOs – exist only for *concrete* bigraphs, not for abstract ones. Intuitively, support provides a means of defining exactly which nodes and edges are shared between two link graphs.

4 Condition-Event Nets as Link Graphs

We are now ready to set up condition-event nets as link graphs¹. There are many ways to do it; we choose one that appears to give a smooth treatment. We shall use the example from the introduction as an illustration:



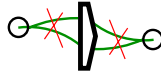
We choose three kinds of control: m ('marked') and u ('unmarked') for holding and non-holding conditions, and e_{hk} for events with h pre- and k post-conditions. The shape and colour of each node will save us from writing controls in diagrams. Conditions have arity 1; we site the single port of a condition node in its centre. An e_{hk} -node has $h + k$ ports; h pre-ports for pre-conditions, k post-ports for post-conditions. You may like to check that the above net has two open and three closed links.

¹ Terminology can become confused when discussing two different formalisms. In particular, Petri nets and bigraphs differ in their use of the terms 'transition' and 'place'. Fortunately, in this paper we are concerned only with *condition-event* nets, not *place-transition* nets, so we are able to avoid confusion.

We adopt the many-one sorting described above. Specifically, there are two sorts, γ for condition ports and η for event ports. An interface assigns one of these sorts to each of its names. When a net satisfies the many-one sorting constraints from the previous section (with η and γ for α and β) we call it *well-sorted*. Thus, in a well-sorted net, each condition has a single link to all its pre- and post-events, and each event port is linked to at most one condition. Let us denote the precategory of well-sorted condition-event nets by 'CE ; the accent means that we are dealing with *concrete* link graphs.

In general an interface may contain both γ -names and η -names. But in the example you will notice that both x and y are γ -names, because each names a link containing a condition. In fact we shall confine our attention to the subprecategory 'CE_γ whose interfaces contain only γ -names, and whose nets are well-sorted. We call these γ -nets, for short. The ground γ -nets are our agents; note in particular that an agent contains all the pre- and post-conditions of its events.

The reader should note that our encoding of condition-event nets into γ -nets is not surjective, even up to support equivalence. The reason is that, in an encoded condition-event net, each pre-condition of a single event is linked to exactly one of its pre-ports (and similarly for post-conditions). This constraint is illustrated thus:



There are γ -nets that violate this constraint; it is not imposed by many-one sorting. This situation arises because in 'CE_γ we have equipped events with several ports, for technical reasons. But these spurious γ -nets need not disturb us, for it can be shown that a spurious one never arises from a genuine one as the result of a transition.

Let us now add dynamics to 'CE_γ , making it a reactive system. To do this, we introduce the usual Petri-net firing rules as reaction rules (r, r') , one for each e_{hk} . Figure 3 shows the rule for $h = 1, k = 2$. Note that r and r' are indeed agents. Note also that all the links of r are open; this means for every occurrence of r in an agent f there is a context D such that $f = D \circ r$. You may be concerned that we have given *particular* names to the interface of our reaction rules; this is no constraint, because by using wirings we can rename – or close – these names at will.

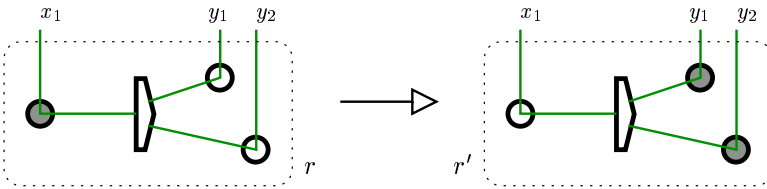


Fig. 3. A link-graph reaction rule for condition-event nets

We are now ready to examine the behaviour of γ -agents. Recall from the introduction that we already have a TS for them, namely \mathcal{L}_p , defined without any help from link graph theory; the labels ℓ in its transitions $f \xrightarrow{\ell} f'$ take one of the forms $+x, -x$ or τ .

We shall assume each transition relation $\xrightarrow{\ell}$ to be closed under support equivalence. Denote by \sim_p the strong bisimilarity induced by \mathcal{L}_p .

To compare this with the strong bisimilarity \sim induced by link graph theory, let us now define the latter equivalence accurately. First recall that in the TS \mathcal{L}_g , each L -transition $f \xrightarrow{L} f'$ is such that, for some D and reaction rule (r, r') , the pair (L, D) is a pushout for (f, r) , and $f' \simeq D \circ r'$. This ensures that \mathcal{L}_g also is closed under support equivalence. Then, recalling the introduction, the equivalence \sim is the largest symmetric relation such that

whenever $f \sim g$ and $f \xrightarrow{L} f'$, with $L \circ g$ defined,
there exists g' such that $g \xrightarrow{L} g'$ and $f' \sim g'$.

(The condition that $L \circ g$ be defined is needed because we are working in a precategory.) Unlike \sim_p , the bisimilarity \sim is guaranteed by link graph theory to be a congruence, i.e. preserved by insertion into any context.

Our first task is to characterise the labels of \mathcal{L}_g . We omit the detailed analysis. It turns out that (up to isomorphism in CE_γ) each label is either an identity, or an open γ -net with exactly one e-node, linked to zero or more m-nodes as preconditions and u-nodes as post-conditions. An identity label just signifies that the agent makes a transition with no assistance from its environment. In fact $f \xrightarrow{\text{id}} f'$ iff $f \longrightarrow f'$; this justifies our use of the same arrow for both reactions and transitions.

Figure 4 shows a non-identity label. It is not quite a redex; it requires its client agent

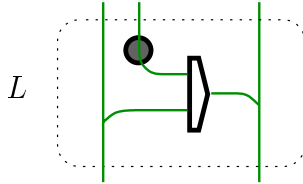


Fig. 4. A typical label in \mathcal{L}_g

to provide one marked precondition and one unmarked postcondition. Figure 5 shows the anatomy of a transition $f \xrightarrow{L} f'$ with this label. Note that f' takes the form $\overline{L} \circ \overline{f}$; we call \overline{L} and \overline{f} the *residuals* of L and f respectively. We see that a single transition may change the marking of several named conditions of f , however far apart they may lie in f . Any other agent g with the same interface as f will have a similar transition, provided only that it has the same initial marking of its named conditions.

The two TSs \mathcal{L}_p and \mathcal{L}_g are significantly different, so it is not immediately clear that they will induce the same bisimilarity. We prove that they do so in the next section.

5 Coincidence of Bisimilarities

In CE_γ we have two TSs on condition-event nets: \mathcal{L}_p defined directly with labels ℓ of the form $+x, -x$ or τ , and \mathcal{L}_g derived in link graph theory, with labels L consisting of

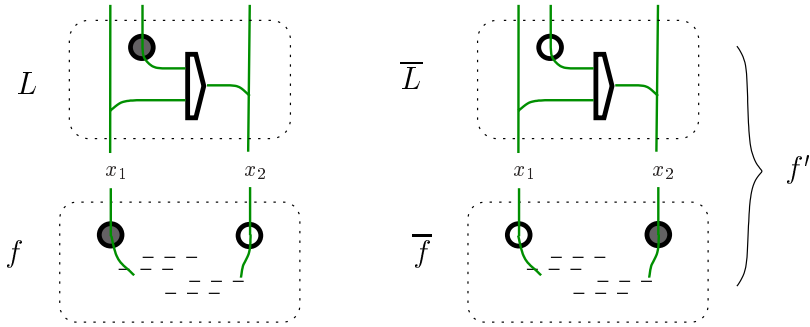


Fig. 5. Anatomy of a transition $f \xrightarrow{L} f'$ in \mathcal{L}_g

link graph contexts having at most a single event node. The bisimilarities for the two TSs are \sim_p and \sim respectively.

We shall first show that $\sim \subseteq \sim_p$. This asserts that if we can distinguish two γ -nets f and g by using ‘experiments’ ℓ like $+x$ and $-x$, then we can also do so using ‘experiments’ L that are elementary link graph contexts. So, among the labels L generated by our theory (see Figure 4), we need to find those that can do the job of the experiments $+x$ and $-x$.

It turns out that such labels need only involve events with one pre- and one post-condition; we call them *input* and *output probes* respectively. They are denoted by in_{xz} and out_{xz} , and are shown in the first column of Figure 6. The second column shows the *spent* probes, the residuals of the probes. The third column shows the spent probes with their conditions closed; they are defined by $\overline{\text{in}}_x \stackrel{\text{def}}{=} /z \circ \overline{\text{in}}_{xz}$ and $\overline{\text{out}}_x \stackrel{\text{def}}{=} /z \circ \overline{\text{out}}_{xz}$. They may be called *twigs* because, up to the equivalence \sim , they can be broken off. The

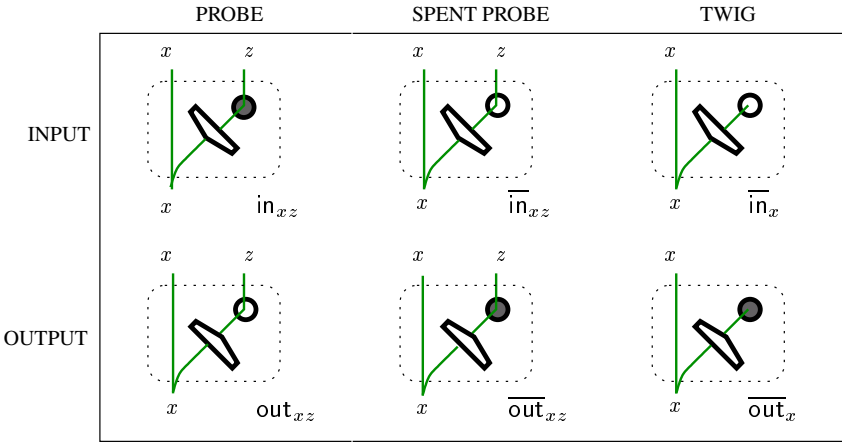


Fig. 6. Probes: labels in \mathcal{L}_g for observing conditions in a γ -net

intuition is simply that a twig occurring anywhere in a net can never fire. In fact we have a lemma, proved easily in link graph theory:

Lemma 1. *For any γ -agent f having x in its outer face, $\overline{\text{in}}_x \circ f \sim \overline{\text{out}}_x \circ f \sim f$.*

Now to prove that $\sim \subseteq \sim_p$ it is enough to show that \sim is an \mathcal{L}_p -bisimulation. For this, suppose that $f \sim g$, and let $f \xrightarrow{\ell} f'$ in \mathcal{L}_p . We must find g' such that $g \xrightarrow{\ell} g'$ and $f' \sim g'$. If $\ell = \tau$ this is easy, because then our assumption implies that $f \xrightarrow{\tau} f'$, and hence $f \xrightarrow{\text{id}} f'$ in \mathcal{L}_g ; but then by bisimilarity in \mathcal{L}_g we have $g \xrightarrow{\text{id}} g' \sim f'$, and by reversing the reasoning for f we get that $g \xrightarrow{\tau} g'$ and we are done.

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

$$f \xrightarrow{\text{in}_{xz}} f'' \stackrel{\text{def}}{=} \overline{\text{in}}_{xz} \circ f'.$$

Hence by bisimilarity in \mathcal{L}_g we have

$$g \xrightarrow{\text{in}_{xz}} g'' = \overline{\text{in}}_{xz} \circ g'$$

where $f'' \sim g''$ and g' is the residual of g'' under the transition. This residual g' differs from g only in having a marked condition named x that was unmarked in g , and hence we also have $g \xrightarrow{+x} g'$ in \mathcal{L}_p . It remains only to show that $f' \sim g'$. We deduce this using the congruence of \sim and Lemma 1:

$$\begin{aligned} f' \sim \overline{\text{in}}_x \circ f' &= /z \circ \overline{\text{in}}_{xz} \circ f' = /z \circ f'' \\ &\sim /z \circ g'' = /z \circ \overline{\text{in}}_{xz} \circ g' = \overline{\text{in}}_x \circ g' \\ &\sim g', \end{aligned}$$

and so we have proved

Lemma 2. $\sim \subseteq \sim_p$.

To complete our theorem we must prove the converse, $\sim_p \subseteq \sim$. It would be enough to prove that \sim_p is an \mathcal{L}_g -bisimulation; but this is false. Instead we have to consider the closure of \sim_p under all contexts, namely

$$\mathcal{S} \stackrel{\text{def}}{=} \{ (C \circ f, C \circ g) \mid f \sim_p g \}.$$

In fact it will be enough to prove that \mathcal{S}^\simeq , the closure of \mathcal{S} under support equivalence, is a bisimulation. We get the required result by considering the case $C = \text{id}$.

So let us assume that $f \sim_p g$, and that $C \circ f \xrightarrow{M} f''$ in \mathcal{L}_g . Then there is a reaction rule r and context D such that (M, D) forms a pushout for $(C \circ f, r)$, as shown in the left-hand diagram of Figure 7, and $f'' \simeq D \circ r'$. We now take the pushout (L, F) for (f, r) , and properties of pushouts yield the right-hand diagram, in which the upper square is also a pushout. So there is a transition $f \xrightarrow{L} f'$, where $f' \simeq F \circ r'$; note also that $f'' \simeq C' \circ f'$.

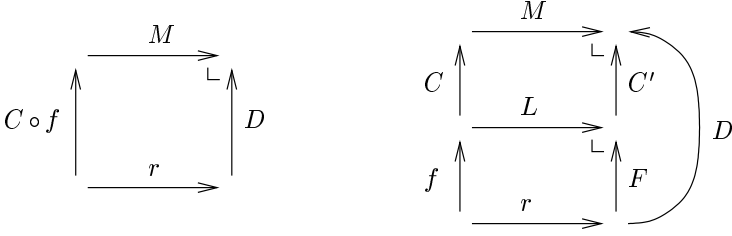


Fig. 7. Pushouts underlying transitions of $C \circ f$ and f

Now consider the anatomy of this transition, exemplified in Figure 5. We know that the residual \bar{f} differs from f only in the changed marking of zero or more named conditions. It follows therefore that in \mathcal{L}_p there is a sequence of transitions

$$f \xrightarrow{\ell_1} f_1 \dots \xrightarrow{\ell_n} f_n = \bar{f} \quad (n \geq 0)$$

where $\ell_i \in \{+x_i, -x_i\}$; each transition marks or unmarks a single named condition. Moreover $f' = \bar{L} \circ \bar{f}$. Since $f \sim_p g$ there exists a similar sequence

$$g \xrightarrow{\ell_1} g_1 \dots \xrightarrow{\ell_n} g_n = \bar{g}$$

with $\bar{f} \sim_p \bar{g}$. This implies that g has the same initial marking as f for the named conditions involved in the transitions. But we know that $L \circ g$ is defined (since we assumed $M \circ C \circ g = C' \circ L \circ g$ to be defined), so in \mathcal{L}_g there is a transition $g \xrightarrow{L} g' \stackrel{\text{def}}{=} \bar{L} \circ \bar{g}$. Its underlying pushout is shown in the left-hand diagram of Figure 8. Also it has an underlying reaction rule (s, s') , with $g' \simeq G \circ s'$.

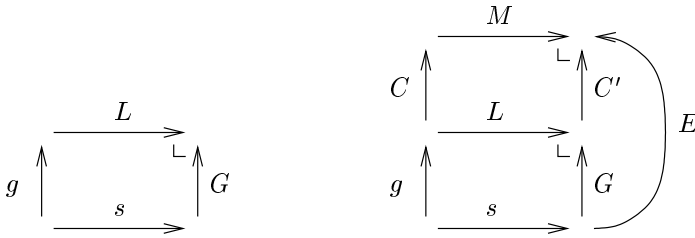


Fig. 8. Pushouts underlying transitions of g and $C \circ g$

Now we form the right-hand diagram of Figure 8 by replacing this pushout for the lower square in right-hand diagram of Figure 7. Since both small squares are pushouts, so is the large square; therefore it underlies an \mathcal{L}_g -transition

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

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

$$g'' = E \circ s' = C' \circ G \circ s' \simeq C' \circ g' = C' \circ \overline{L} \circ \overline{g},$$

and hence $(f'', g'') \in \mathcal{S}^\approx$ since $\overline{f} \sim_p \overline{g}$. It follows that $\sim_p \subseteq \sim$.

So we have proved:

Theorem 1. (coincidence of bisimilarities) $\sim = \sim_p$ in 'CE_γ .

The reader will remember that we have worked in *concrete* link graphs 'CE_γ in order to ensure the existence of IPOs (in fact pushouts); these were needed to define a transition system in a way that ensures congruence of bisimilarity. Having done this, we can now transfer both the transitions and the bisimilarity to the corresponding category CE_γ of *abstract* link graphs, which has no IPOs. Note that CE_γ is indeed a category, not just a precategory, because support no longer places a constraint upon composition.

If G is a concrete link graph, let $[G]$ denote the corresponding abstract one – essentially the support-equivalence class of G . Then we define the transition system $[\mathcal{L}_g]$ in CE_γ to be the smallest set such that

$$\text{if } g \xrightarrow{L} g' \text{ in } \mathcal{L}_g \text{ then } [g] \xrightarrow{[L]} [g'] \text{ in } [\mathcal{L}_g].$$

Similarly we define $[\mathcal{L}_p]$ in CE_γ ; this is even simpler because its labels ℓ are not subject to support equivalence.

These two abstract transition systems induce corresponding bisimilarities in CE_γ ; we shall again denote them by \sim and \sim_p . They are simply related to those in 'CE_γ . We conclude by stating this relationship, omitting the proof; it also has the consequence that the assertion of Theorem 1 for concrete link graphs is matched for abstract link graphs.

Corollary 1. (coincidence of bisimilarities in abstract condition-event nets)

1. $f \sim g$ in 'CE_γ iff $[f] \sim [g]$ in CE_γ .
2. $f \sim_p g$ in 'CE_γ iff $[f] \sim_p [g]$ in CE_γ .
3. In CE_γ bisimilarity \sim is a congruence and coincides with \sim_p .

6 Discussion

This exercise has shown that a particular class of Petri nets, condition-event nets, can be modelled and analysed in link graphs. It has not shown that this modelling is canonical, nor that it extends to other net disciplines. I hope that the relatively simplicity of the present analysis may provoke interest in these questions.

This would not only determine how far the present theory of bigraphs goes for Petri nets; it may also suggest improvements and variations of bigraph theory. Indeed it was by trying to analyse other concurrency models – especially the π -calculus of Milner, Parrow and Walker [7] and the mobile ambients of Cardelli and Gordon [2] – that bigraphs evolved from their predecessor action calculi, Milner [6]. A large concern in

defining bigraphs has been to admit theoretical analysis (such as we have illustrated) which could not be provided so well for action calculi. Furthermore, Jensen and Milner [3] have recently shown that the behavioural theory of a version of the π -calculus can be exactly mirrored in bigraphs.

Even within the present exercise, interesting points emerge. In the present encoding of Petri nets, we stratify the event controls e_{hk} by their arities; this limits the number of pre- and post-conditions that can be connected to a given event in any context. In contrast, Nielsen et al [9] use a *recursion* combinator that connects a given condition to a given event. Thus the algebraic combinators provided uniformly by link graphs may not coincide with those designed for a particular application, and the comparison of the two requires further examination.

Another interesting outcome is the mismatch between the transition system \mathcal{L}_g generated by link-graph theory and the simple specific transition system \mathcal{L}_p , despite the coincidence of the bisimilarities they induce. It is clear that the labels in \mathcal{L}_g are redundant, in the sense that the same phenomenon may be detected by more than one experiment. This is not surprising, because the labels are generated from each reaction rule *separately*; no attempt has yet been made to discover to what extent the labels from a *family* of reaction rules duplicate each other's discriminating power. The exercise suggests that in further development the theory of BRSs we should try to identify general properties of rule-sets that lead to such redundancies; this could yield more economical transition systems.

Our formulation of bigraphs uses precategories for two reasons. First, they provide concrete bigraphs with RPOs, which categories do not. Second, more generally, they provide a very direct way to distinguish different *occurrences* within the same bigraph. Although manipulation with precategories is not troublesome, they are not standard in category theory. Sassone and Sobocinski [13] have provided a valuable link with a more standard categorical concept, 2-categories; using these they have been able to recover exactly the RPO theory and congruence theory. Whether 2-categories will ease the further development of bigraphical theory, as presented in Jensen and Milner [4], is a topic for further research. But there is already advantage in an alternative formulation.

Finally, although no proposal is made here about how to enrich Petri nets with mobility, the present exercise offers a microcosm in which to test such proposals.

In summary, Petri nets and bigraphs may be able to enrich one another.

Acknowledgement

I am grateful to Mogens Nielsen, Vladimiro Sassone and Thiagarajan for helpful comments on the ideas in this paper.

References

1. Best, E., Devillers, R. and Hall, J.G., The box algebra: a model of nets and process expressions. 20th International Conference on Application and Theory of Petri Nets, LNCS 1639 (1999) 344–363.
2. Cardelli, L. and Gordon, A.D., Mobile ambients. Foundations of System Specification and Computational Structures, LNCS 1378 (2000) 140–155.

3. Jensen, O.-H. and Milner, R., Bigraphs and transitions. In Proc. 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (2003).
4. Jensen, O.-H. and Milner, R., Bigraphs and mobile processes. Technical Report UCAM-CL-TR-570, University of Cambridge Computer Laboratory (2003). Also available at <http://www.cl.cam.ac.uk/users/rm135>, together with an index and slides.
5. Leifer, J.J. and Milner, R., Deriving bisimulation congruences for reactive systems. Proc. CONCUR 2000, 11th International Conference on Concurrency Theory (2000) 243–258.
6. Milner, R., Calculi for interaction. *Acta Informatica* 33 (1996) 707–737.
7. Milner, R., Parrow, J. and Walker D., A calculus of mobile processes, Parts I and II. *Journal of Information and Computation* 100 (1992) 1–77.
8. Park, D., Concurrency and automata on infinite sequences. In LNCS 104, Springer Verlag (1980).
9. Nielsen, M., Priese, L. and Sassone, V., Characterizing behavioural congruences for Petri nets. Proc. CONCUR'95, LNCS 962 (1995) 175–189.
10. Pomello, L., Rozenberg, G. and Simone, C., A survey of equivalence notions for net-based systems. *Advances in Petri Nets '92*, LNCS 609 (1992) 410–472.
11. Priese, L. and Wimmel, H., A uniform approach to true-concurrency and interleaving semantics for Petri nets. *Theoretical Computer Science* 206 (1998) 219–206.
12. Reisig, W., *Petri Nets: an Introduction*. Springer Verlag, Berlin (1985).
13. Sassone, V. and Sobocinski, P., Deriving bisimulation congruences: 2-categories vs. precategories. In Proc. FOSSACS '03, LNCS 2620 (2003) 409–424.