Seminar notes on developments in bigraphs

Robin Milner, October 2009
Universities of Cambridge and Edinburgh

These notes support seminars on theoretical advances in the bigraph model. They are conjectural; they aim to stimulate research rather than to report completed work. They also evolve, as new challenges arise both from experimental application and from the theory itself.

The notes assume some knowledge of my recent book, *The Space and Motion of Communicating Agents* (Cambridge University Press). Links to the book can be found on my website, `www.cl.cam.ac.uk/~rm135`. Some of the seminars develop ideas suggested in Chapter 11 of the book.

On the website can also be found the slides of a course of six lectures, and explanatory notes linking the lectures to the book.

Seminar Note A: Embedding calculation

Robin Milner, 2009

The first section of this note reviews Section 11.2 in my book, which explains how non-parametric recursion can be modelled by a special class of reaction rules called atomic germination rules.

The second section generalises this to parametric germination rules, showing how they model calculation over data types. The resulting BRS may be called a data calculus. The third section then shows how a data calculus can be embedded in a general BRS. In particular, it allows ordinary reaction rules to maintain data values and to use them as preconditions for the enaction of a rule.

It can be compared with the ITU's Platographical BRSs, which combine BRSs that share (some) controls; but the emphasis here is on calculation.

1 Growth, or atomic unfolding

An atomic *germination rule* is a specialised reaction rule of the form

$$\mathsf{K}_{\vec{x}} \hookrightarrow g_{\mathsf{K}}$$

where K is an atomic control called a *seed* and the names $\vec{x}$ are distinct. We also call $\mathsf{K}_{\vec{x}}$ a seed. We find it useful to require the unfolded seed $g_{\mathsf{K}} : \langle 1, \{\vec{x}\}\rangle$ to have no idle root (it only has a single root) and no idle names. It may contain occurrences of K and other seeds. Thus the unfolding of seeds can represent recursion; for example, it models a CCS user-defined recursion

$$A(\vec{x}) \stackrel{\text{def}}{=} P_A \ .$$

This constrained form of reaction rule allows us to derive very strong behavioural properties, including a harmony between unfolding and the standard reaction relation $\longrightarrow\!\!\triangleright$.

Given a set Δ of germination rules with distinct seed controls, let $\hookrightarrow_\Delta$ stand for the union of their germination relations extended to bigraphs (not just ground bigraphs) by closure under contexts and under support equivalence $\doteq$. We omit Δ when it is understood. Note that, unlike the reaction relation $\longrightarrow\!\!\triangleright$, we close $\hookrightarrow$ under *all* contexts, not merely active ones.

We then define the *unfolding order* $\leq$ as the transitive reflexive closure of $\hookrightarrow$. We denote the symmetric closure of $\leq$ by $\equiv$; we call it *structural congruence*. An important property of unfolding is that it is preserved by every context – i.e. it is *congruential*. Second, it is *decomposable*; this means that the parts of a composition or product grow independently, i.e. if $E \circ F \leq G$, then $G = \widehat{E} \circ \widehat{F}$ for some $\widehat{E}, \widehat{F}$ such that $(E, F) \leq (\widehat{E}, \widehat{F})$. Crucially, $\leq$ enjoys two further properties:

- $\leq$ is *confluent*.
- If no seed occurs in the redex of any standard reaction rule, then $\leq$ *respects reaction*; in the following sense. Let $\longrightarrow\!\!\triangleright$ be the reaction relation generated by the standard rules; then, if $f \longrightarrow\!\!\triangleright f'$ and $f \leq g$, there exists g such that $g \longrightarrow\!\!\triangleright g'$ and $f' \leq g'$.

Recall that $\leq$ involves any number of unfoldings by $\hookrightarrow$, while $f \longrightarrow\!\!\triangleright f'$ represents a single reaction within f. The last property is proved by induction on the number of unfolding steps in $f \leq g$. The proof requires decomposability, and also the fact that unfolding cannot occur within the parametric redex R of a reaction, since R contains no seed. Note that unfolding *can* occur in the parameter d of a reaction; but then, if the reaction precedes the unfolding, a single unfolding in d can be matched by zero of more unfoldings of the parameter instance $d' = \overline{\eta}(d)$, since any factor of d may be replicated or discarded by the reaction.[1]

In the next section we generalise unfolding rules to admit parameters; in the final section we show that it retains respect for reaction.

2 Parametric unfolding

We now proceed to generalise germination, allowing a seed to germinate differently for different parameter patterns. A seed control K is no longer atomic; instead it may have any rank $k \geq 0$, i.e. $\mathsf{K}_{\vec{x}}$ is an ion with k sites. The book (on page 64) describes how to derive such higher-rank controls with the help of a sorting.

In this section we are concerned with a BRS in which each redex consists of seeds with patterns; that is, we are concerned only with unfolding, not general reaction. In the next section we consider how to embed such an unfolding BRS into a general one.

We wish to use each seed to specify a set of *patterns* for the germination by K.[2] Each pattern for K will be determined by a place graph P with outer width k, containing no seed controls. (Later, we see that P may be built from data constructors.) The patterns for a given seed K must be pairwise inconsistent, i.e. for any pair P, Q of patterns for K we must have

$$P.d \neq Q.e \ .$$

for all parameters d, e. Thus a *ground* seed, i.e. a seed supplied with parameters, takes the form of a molecule $\mathsf{K}_{\vec{x}}.P.d$; recall that the so-called *nesting* operator '$.$' is a derived form of composition in which the names of the second component are exported. Then a germination rule for a seed control K and pattern P takes the form

$$\mathsf{K}_{\vec{x}}.P \hookrightarrow G$$

where P and G have inner widths m and m', together with (as for general reaction rules) an instantiation map $\eta : m' \to m$ determining how a parameter $\vec{d}$ for the seed is instantiated. In fact, for $d = d_0 \otimes \cdots \otimes d_{m-1}$ we define

$$d' \stackrel{\text{def}}{=} d_{\eta(0)} \parallel \cdots \parallel d_{\eta(m'-1)} \ .$$

[1] In the book it is said that the proof requires the reaction rule to be *affine*, i.e. to do no replication. This constraint appears unnecessary for the present proof (this should be checked), but may be needed for later properties. However, since unfolding may occur in the parameter d of the reaction $f \longrightarrow\!\!\triangleright f'$, we are relaxing the book's constraint that the parameter of a redex should be discrete, merely requiring it to have no idle roots or names. This relaxation does not affect the book's theory of standard transitions, but may affect the theory of engaged transitions (this should be checked).

[2] Recall the notion of parameter pattern from programming languages, e.g. Standard ML.

Then the unfolding relation for this rule is generated from

$$\mathsf{K}_{\vec{x}}.P.d \hookrightarrow G.d'$$

by closure under all contexts.

For simplicity let us constrain a pattern P to be a place graph (i.e. to have no links). It may be possible to relax this constraint, but it is easier to preserve it for the present. There are plenty of examples, in particular for recursive functions on data types, where the constructors of the data type are place controls, i.e. they have arity 0.

Consider list processing; the list constructors are Cons with rank 2 and Nil with rank 0. An arbitrary finite list whose members $\{d_i : i \in n\}$ have disjoint name-sets is represented by

$$\mathsf{Cons}.(d_0 \otimes \mathsf{Cons}.(\cdots \mathsf{Cons}.(d_{n-1} \otimes \mathsf{Nil}) \cdots)) \ .$$

Notation In keeping with familiar mathematical notation, we shall often write a comma for the tensor product $\otimes$, We shall write a context such as Cons occurring in a parametric seed $\mathsf{K}_{\vec{x}}.P$ as $\mathsf{Cons}(\Box_i, \Box_j)$, where the indices i and j indicate the ordinal positions of these two sites in the inner width of P.

Note that the first site of a Cons must have the sort of the list's items while the second site has the sort of a list of such items. Note also that the list items d_i may have names, which of course can be shared or closed by imposing a suitable context.

Constructors are not seeds, but a recursive function over lists will be represented as a seed. Consider the binary function Cat for concatenating two lists; it is a seed with rank 2, and its patterns are

$$P_{\mathsf{Cons}} = \mathsf{Cons}.(\boxed{0}, \boxed{1}), \boxed{2} \quad \text{and} \quad P_{\mathsf{Nil}} = \mathsf{Nil}, \Box \ .$$

The evaluation of Cat is represented by two unfolding rules:

$$\begin{array}{rcl} \mathsf{Cat}.(\mathsf{Cons}.(\boxed{0}, \boxed{1}), \boxed{2}) & \hookrightarrow & \mathsf{Cons}.(\boxed{0}, \mathsf{Cat}.(\boxed{1}, \boxed{2})) \\ \mathsf{Cat}.(\mathsf{Nil}, \Box) & \hookrightarrow & \Box \ . \end{array}$$

Of course, other data types, such as the booleans with constants (i.e. nullary constructors) true and false, and the natural numbers with unary constructor Succ (successor) and constant Zero, can be similarly treated. With one exception (to do with copying, see below), we have shown how *calculational* systems can at least be specified as BRSs, though we would expect them to be implemented more efficiently.

What about the properties asserted in Proposition 11.6 of the book for the unfolding relation $\leq \stackrel{\text{def}}{=} \hookrightarrow^*$? They are all retained. It is *congruential* (preserved by composition and tensor product); it is *confluent*[3]; and it is *decomposable*, provided that we constrain contexts so that every data constructor (e.g. Cons) is equipped with a ground parameter.

The exception is to do with copying data. Hitherto, we have declared that the redex of a parametric germination is prime, and is moreover a molecule of the form $\mathsf{K}_{\vec{x}}.P$. This does not

[3]It should be checked that this follows as in the λ-calculus by means of the parallel moves lemma.

permit fetching copies of data, possibly even from a remote region. Thus, for example, the unfolding relation for Cat replaces the molecule $\mathsf{Cat}(d_0, d_1)$ by the concatenation of the lists d_0 and d_1, overwriting these two lists. To avoid this overwriting we want to add a polymorphic non-prime germination rule of the form

$$\mathsf{Fetch}_x \parallel \mathsf{Val}_x.\boxed{0} \hookrightarrow \boxed{0} \parallel \mathsf{Val}_x.\boxed{0}$$

which replaces the Fetch-node with a copy of the contents of the Val-node without changing the latter. This is an instance of what we shall call a *contextual* rule; see Seminar Note D, Section 2. We strongly conjecture that with such a rule (which is indeed used in Seminar Note B for encoding the λ-calculus) the unfolding relation remains confluent, but a rigorous proof is needed.

Let us call a BRS a *calculus* if its controls are either data constructors or seeds (including Fetch and Val), with an appropriate place sorting, whose only reaction rules are unfolding rules as described above. How then do we equip an arbitrary BRS with a data calculus?

Digression Before we can answer this question properly, let us define what it means to be a *sub-BRS* of a given BRS $\mathbf{A} = \mathrm{BG}(\Sigma, \mathcal{R})$. First, consider the s-category $\mathbf{A} = \mathrm{BG}(\Sigma)$,[4] where $\Sigma = (\mathcal{K}, \mathcal{D})$, a signature $\mathcal{K}$ paired with a *sorting discipline* $\mathcal{D}$. In my book the latter was called a *formation rule*; it determines a sub-s-category of the s-category of bigraphs over $\mathcal{K}$. To form a sub-s-category $\mathbf{A}' = \mathrm{BG}(\Sigma')$, where $\Sigma' = (\mathcal{K}', \mathcal{D}')$, we take a subset $\mathcal{K}' \subseteq \mathcal{K}$ and further refine the discipline $\mathcal{D}$ to $\mathcal{D}'$.

Now, to form a sub-BRS of $\mathbf{A}$, including its reaction rules $\mathcal{R}$, we must modify the latter. We pick a subset $\mathcal{R}' \subseteq \mathcal{R}$, in which the redex and reactum of each rule are both admitted by $\mathcal{D}'$. Finally, since $\mathbf{A}' = \mathrm{BG}(\Sigma', \mathcal{R}')$ must be a BRS in its own right, the parameters of each ground reaction must themselves be admitted by $\mathcal{D}'$. This completes the definition of *sub-BRS*.

3 Embedding a calculus in a BRS

We now wish to consider that the reaction rules governing the behaviour in a BRS $\mathbf{A}$ depend upon calculation with data that is present. For example, an agent may be allowed entry to a room only if her name is in a certain list, and if the room is not already full to capacity. Already, this involves lists, numbers and truth values, so we expect these to be in a calculus $\mathbf{C}$ that we shall embed in $\mathbf{A}$.

In general, let us declare a BRS $\mathbf{A}$ to be *calculational* if it has a sub-BRS $\mathbf{C} = \mathrm{BG}(\Sigma_{\mathbf{C}}, \mathcal{R}_{\mathbf{C}})$ that is a calculus, provided that certain further conditions are satisfied. We need these conditions to ensure that calculation *assists* other reactions, and never *preempts* them. Let us call a control *real* if it belongs to $\mathcal{K} \setminus \mathcal{K}_{\mathbf{C}}$, and a reaction rule *real* if it belongs to $\mathcal{R} \setminus \mathcal{R}_{\mathbf{C}}$. We wish to be sure that a real reaction is never preempted by a calculational one, i.e. an unfolding.

To see the need for further conditions to ensure this, suppose R is a real redex. If R contains a $\mathbf{C}$-node, i.e. a seed or a constructor, then an occurrence of R may be destroyed if this node can take part in an unfolding.

[4] We still call it $\mathbf{A}$ although we have dropped its reaction rules.

We now claim that if the sorting discipline of **A** implies three simple conditions, then such preemption cannot occur. The conditions are that:

- Every **C**-node contains only **C**-nodes
- No **C**-seed occurs in a real redex R
- Any **C**-constructor in a real redex R lies within a real node of R.

Notice that the last two conditions apply only to a *parametric* real redex; thus, **C**-nodes may occur in the *parameter* for a real reaction.

These conditions imply[5] that the unfolding relation in a calculational BRS respects its real reaction relation, i.e.

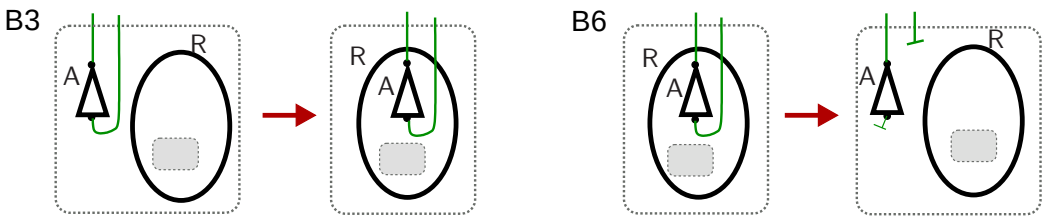
if $f \longrightarrow\!\!\triangleright\, f'$ and $f \leq g$, there exists g such that $g \longrightarrow\!\!\triangleright\, g'$ and $f' \leq g'$.

This is the key property we require of unfolding. Unfolding *facilitates* reaction, e.g. by unfolding seeds until a parametric redex is matched, but does not prevent or duplicate reactions that are already possible.

If **A** is calculational, with calculus **C**, we may write it as $\mathbf{A} = (\Sigma, \mathcal{R}, \mathbf{C})$. Let us look at a simple example. Take **C** to be a calculus for the natural numbers with the constructor Succ and constant Zero; for now, give it just one atomic seed, infinity:

$$\infty \hookrightarrow \mathsf{Succ}.\infty \ .$$

Take **A** to be a refinement the built environment of Chapter 1 in the book. In Σ we include agents (Agent) and rooms (Room), with arities 2 and 0. Recall the rules B3 and B6 in the book, allowing an agent to enter and leave a room:



And in algebraic form:

$$\begin{aligned} &\text{B3}: \quad \mathsf{Agent}_{xy} \,|\, \mathsf{Room}.\Box \longrightarrow\!\!\triangleright\, \mathsf{Room}.(\mathsf{Agent}_{xy} \,|\, \Box) \\ &\text{B6}: \quad \mathsf{Room}.(\mathsf{Agent}_{xy} \,|\, \Box) \longrightarrow\!\!\triangleright\, /y\, \mathsf{Agent}_{xy} \,|\, \mathsf{Room}.\Box \ . \end{aligned}$$

This allows the room to contain an unbounded number of agents. In our refinement we wish to keep count of the agents in a room, and impose a limit on it. So let us add to Σ a non-atomic control Vacs that holds the number of vacancies in the room, decrementing it for each agent who enters and incrementing it for each agent who leaves. So entry is permitted only when the Vacs

[5] The formal argument has yet to be written down

node for the room (the sorting must ensure only one per room) contains a non-zero number – i.e. one that is a successor. . The rules become:

$$\mathrm{B3}' : \quad \mathsf{Agent}_{xy} \mid \mathsf{Room}.(\mathsf{Vacs}.\mathsf{Succ}.\boxed{0} \mid \boxed{1}) \longrightarrow\!\!\triangleright\ \mathsf{Room}.(\mathsf{Agent}_{xy} \mid \mathsf{Vacs}.\boxed{0} \mid \boxed{1})$$
$$\mathrm{B6}' : \quad \mathsf{Room}.(\mathsf{Agent}_{xy} \mid \mathsf{Vacs}.\boxed{0} \mid \boxed{1}) \longrightarrow\!\!\triangleright\ /y\, \mathsf{Agent}_{xy} \mid \mathsf{Room}(\mathsf{Vacs}.\mathsf{Succ}.\boxed{0} \mid \boxed{1})\ .$$

Notation We can write the children of a control in arbitrary order. For example, we may write the context $\mathsf{Room}(\mathsf{Vacs}.\mathsf{Succ}.\boxed{0} \mid \boxed{1})$ also as $\mathsf{Room}.(\boxed{1} \mid \mathsf{Vacs}.\mathsf{Succ}.\boxed{0})$. This is because we are indexing the sites.

It is easy to see that the original reaction rule, which allows any number of agents in a room, is equivalent to having assigned ∞ to its Vacs node, because $\infty \equiv \mathsf{Succ}.\infty$.[6] In the refinement, let us suppose that the vacancy count was originally set with no agents already in the room, and that it is altered only by the two rules we have given. Then these rules will ensure that the occupancy will never exceed the original vacancy count.

Let us refine the example further. Suppose we add a rule (which we shall not trouble to define) allowing an administrator to change the limit, by altering the vacancy count of a room. Suppose he gives the Vacs node a new value – say 5 – at a time when there may be arbitrarily many agents in the room. This will not immediately limit the occupancy to 5, because it may already exceed 5. In fact the effect will be rules merely to limit the occupancy to 5 more than the current occupancy. Consider two solutions to this problem:

1. The administrator first to assigns 0 to the Vacs node. This prevents any further entry, but at some point (depending upon the stochastic rates) all current occupants will have left. This situation can be detected by a rule whose parametric redex is only matched by a room empty of agents. (This can be done by a sorting in which a site may be assigned a sort 'noagents', forbidding any agents to occupy that site.) Using this new rule, the administrator can then assign assign 5 to the Vacs node, achieving the desired effect.

2. We change the existing rules so that instead of Vacs the room contains two nodes, Occs and MaxOccs, respectively the current number of occupants and the allowed maximum. There are then three rules that can change these values; the entry and exit rules change Occs, and the administrator can change MaxOccs. On each of these events the relevant rule can computer afresh the value of a truth-valued node IsVac, which is then used by the entry rule to decide whether or not to admit an agent.

Of course, the second solution requires the embedded calculus $\mathcal{C}$ to calculate predicates, using the data type of truth values. For our case, it must provide (as a seed) the binary predicate Exceeds over numbers. This suggests a general strategy: every rule (in our case the entry rule and the administrator's rule) that may affect the possible use of future rules should compute a predicate over the data it holds, and store the result in a truth-valued node. Then the entry rule can test this value, as a pre-condition for firing.

This does a lot, but not everything. In our scenario, it will not handle the situation in which entry to a room depends not only on data stored in the room, but also on the identity of the agent

[6] Recall that the equivalence $\equiv$ is the reflexive closure of $\leq$.

wishing to enter. This cannot be known until she attempts to enter! Of course, the room may contain a list of the identities of admissible agents, and the agent wishing to enter must provide identity (e.g. with a swipe card). Our general strategy will then work, provides that entry involves two events, i.e. the use of two rules; the first rule is for reading the swipe card and recording admissibility of entry, and the second uses this predicate value, if True, to admit the agent.

But what can ensure that the admitted agent is the one who swiped the card? One solution to this involves extending the notion of reaction rule to include what may be called *conditional* reaction rules, having the form

$$(P, R, R', \eta)$$

where the condition P is a boolean predicate expressed in the calculus, which must be satisfied (i.e. must unfold to True) when applied to the rule's parameter. (In our case, the parameter would include the identity of the agent wishing to enter.) Thus a conditional rule is a *single* rule with two inseparable parts: one to evaluate the condition and one to perform the action.

So our simple example has led us into subtleties of modelling that we might not have expected. That was the point of the example; any model must make it possible to ask and answer subtle questions of this kind.

We leave details of conditional rules for further study. But we can note two things. First, such rules seem quite compatible with the present behavioural theory; for example, they allow labelled transitions to be derived and they still admit the theorem that bisimilarity is a congruence. Second, this kind of conditional rule – imposing a calculable predicate on the *parameter* – is distinct from one in which a condition is imposed upon *context* in which the redex is matched; moreover, such conditions may be expressed either (as here) in a data calculus or in a logic, such as BiLog, associated with bigraphs.

Let us make two points, putting this work in a general context:

- This handling of data structures is just one example of combining bigraphs. Clearly the idea of bigraphical modules is important, and needs careful definition. For example, is there just one such notion, or are calculational BRSs distinct from other kinds of module?

- This treatment is surely not suitable for the end-user of bigraphs, who will need something more user-friendly akin to a programming language. But the treatment is very close to how data structures appear in the semantic definition of a programming language such as Standard ML. Similarly we should aim at a bigraphical programming-cum-specification language with an analogous semantic definition.

Research at the ITU Copenhagen has pioneered work under both these headings.

Seminar Note B: Localisation and binding

Robin Milner, 2009 *last amended 27 November 2009*

This note explains and illustrates the binding of links in bigraphs. It is based upon Section 11.3 of the book, but its formulation and expository sequence differ slightly.

Section 1 explains localisation of links, and how it constrains dynamics. Section 2 uses localisation to model the binding of names, as a sorting for any already existing BRS. Section 3 uses binding to encode the lambda calculus.

Citations [*nn*] in square brackets refer to the bibliography in my book.

1 Local bigraphs

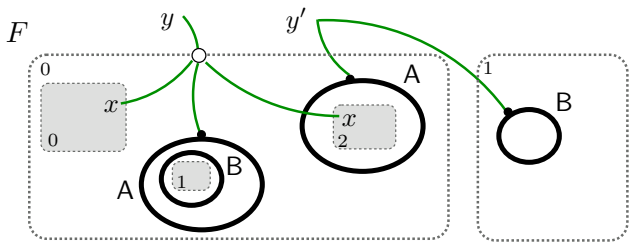
In pure bigraphs, placing and linking are completely orthogonal. This can be seen by considering some $F : \langle m, X\rangle \to \langle n, Y\rangle$, decomposed as follows:

$$F = F_k \circ \cdots \circ F_1 \quad \text{with } F_i : I_{i-1} \to I_i \ (0 < i \in k)$$
$$\text{where } I_0 = \langle m, X\rangle \text{ and } I_k = \langle n, Y\rangle \ .$$

Suppose that some name $x \in X$ is open in F, so that $F(x) = y \in Y$; in fact $x = x_0$ with $F_i(x_{i-1}) = x_i$ and $x_k = y$. Then any node in F can be linked to the outer name y; in fact, for each i $(0 < i \in k)$ any node in F_i can be linked to x_i.

It is useful, for some purposes, to constrain a bigraph by limiting the nodes and sites accessible from certain outer names. We shall call these outer names *local*. Each local name in an interface may be local to one or more of its sites; to represent this, we enrich an interface $I = \langle m, X\rangle$ to $I = \langle m, loc_I, X\rangle$, where the binary relation $loc_I \subseteq m \times X$ is called the *locality* of I. If $(i, x) \in loc_I$ then we say x is *local to* i. If x is not local to any site we call x *global*. Bigraphs with such interfaces are called *local*, provided they satisfy a constraint defined below.

Example Let us first illustrate the constraint with an example. The diagram shows $F : I \to J$ where $I = \langle 3, loc_I, X\rangle$ and $J = \langle 2, loc_J, Y\rangle$, with $x \in X$ and $y, y' \in Y$. The only local name in Y is y; that is, $loc_J = \{(0, y)\}$.



The fact that y is local to region 0 of F is represented by the little circle. F qualifies as a local bigraph, first because all the nodes linked to y are within region 0, to which y is local. Since y' is global, it may be linked (as here) to nodes in any region. But if y' were also local it would have to

be local to both regions of F. (So there would be two more little circles, placed where the y'-link enters each region.)

The second reason that F qualifies as a local bigraph is that the inner name x linked to y is local to two of the sites lying in region 0, and this linkage is represented explicitly by placing x in those sites. We shall write a site as an indexed box, annotated with its local names:

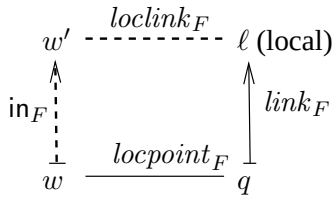
$$F = (y/x \circ (\boxed{0}_x \mid \mathsf{A}_x.\mathsf{B}.\boxed{1} \mid \mathsf{A}_x.\boxed{2}_x) \parallel \mathsf{B}.1 \ .$$

Scoping discipline for local bigraphs We now proceed to express formally our constraint on local bigraphs. Recall that the ports $ports(v)$ of a node v take the form $p = (v, i)$, for $i \in ar(v)$. For any local bigraph $F : I \to J$ with nodes V_F we define the localities of its points and links as follows:

$$\begin{array}{rcl} locport_F & \stackrel{\text{def}}{=} & \{(prnt(v), p) : v \in V_F, p \in ports(v)\} \\ locpoint_F & \stackrel{\text{def}}{=} & loc_I \uplus locport_F \\ loclink_F & \stackrel{\text{def}}{=} & loc_J \ . \end{array}$$

(The relation $loclink_F$ will be extended when binding is introduced later.) Let w, q and ℓ respectively range over places, points and links. Let in_F be the transitive reflexive closure of the parent map of F, and let $link_F$ be its link graph. Then a local bigraph F must obey the following:[7]

> SCOPE DISCIPLINE
> Whenever a link ℓ is local then all its points are local, and (see diagram)
> each location of any point of ℓ lies within a location of ℓ.



We can now see why the locality of an interface is important; by transmitting locality from the outer to the inner face of a bigraph, it ensures that the scope discipline is preserved by composition. It is also easily found to be preserved by tensor product.

What is the effect of allowing a name to be local to some, but not all, of the sites in an interface? One important effect is to exert a useful constraint upon dynamics. Consider a ground reaction rule with redex r having two regions, one of which contains a single node; the rule moves this node to the other region. What can happen if we compose the bigraph F illustrated above with a ground bigraph $g : I$, where g contains an occurrence of r? Suppose the node v to be moved is linked to x, and lies in region 0 of g. Then the rule can move v to region 2 of g, since x is local in that region, but cannot move it to region 1.

[7] This condition was wrongly stated in the book. By a procedural mistake I gave there a condition that works only in the case that a name is local to at most one region. The correct version, as here, is given in the Corrigenda to the book, which can be found on my website. It's worth noting that the two conditions are equivalent if a name can be local to at most one region!

To summarise: the locality of names in an interface I can be used to confine mobility in a bigraph whose outer face is I.

Notation Our notation $\langle m, loc, X\rangle$ for local interfaces is good for setting up their theory, but can be a little tedious in practice. One alternative is to write instead $\langle (X_0, \ldots, X_{m-1}), Y\rangle$, where X_i are the names local to site $i \in m$, and Y are the global names. Then for example

$$\langle (\{x, x'\}, \{x, x''\}), \{y, y'\}\rangle, \text{ or just } \langle (xx', xx''), yy'\rangle$$

has width 2 and name set $\{x, x', x'', y, y'\}$, with $\{x, x'\}$ local to site 0, $\{x, x''\}$ local to site 1, and $\{y, y'\}$ global. With this notation, we can also omit the global names or all the local names, if empty; so $\langle 2, yy'\rangle$ has only global names and $\langle (xx', xx'')\rangle$,or just (xx', xx''), has only local names. $\square$.

Operators Composition and tensor product (justaposition) are easily defined, and we omit details. Note only that for two disjoint interfaces $I = \langle m, loc_I, X\rangle$ and $J = \langle n, loc_J, Y\rangle$

$$I \otimes J \stackrel{\text{def}}{=} \langle m{+}n, loc_I \uplus loc'_J, X \uplus Y\rangle$$

where we define the offset locality $loc'_J \stackrel{\text{def}}{=} \{(m{+}j, y) : (j, y) \in loc_J\}$. The parallel product $I \parallel J$ of two interfaces is similar, with the disjoined union $X \uplus Y$ replaced by the union $X \cup Y$.

Nesting requires a little adjustment. In forming $G\mathbf{.}F$ we make the global names of F accessible as before, but we allow F also to have local outer names, which are local inner names of G. Thus let $F : I \to J$ and $G : J' \to K$ where, using our alternative notation for interfaces, $J = \langle (\vec{X}), X\rangle$, $J' = (\vec{X})$ and $K = \langle (\vec{Y}), Y\rangle$. We require the global names X of J to be disjoint from $\vec{Y}$, the local names of K. Then the nesting $G\mathbf{.}F : I \to \langle (\vec{Y}), X \cup Y\rangle$ is given as in pure bigraphs by

$$G\mathbf{.}F \stackrel{\text{def}}{=} (\mathsf{id}_X \parallel G) \circ F \ .$$

The proof that nesting is associative is as before.

Globalising and localising To end this section we consider the possibilities of converting local names into global ones and vice versa. For the former, there is a simple linking $\gamma : I \to J$ called a *globaliser* by Jensen in his dissertation; it is an identity, except that a name local in I may be global in J. It is obvious that this obeys the scope discipline.

In contrast, there is no bigraph that turns a global inner name into a local outer name; it would violate the discipline. But we can define a partial *operation* on bigraphs to do this. First, for any x and site i in the interface I, we define $(i, x)I$ to be the result of making the x local to i; by iterating this, we define $(\tilde{i}, x)I$ to localise x at any subset $\tilde{i}$ of the sites of I.

Now for any bigraph $F : H \to I$, let x be a global name in I not linked in F to any global name of H. How can we make x local in I, while obeying the scope discipline for F? Consider all the points of F linked to x; each such point is either a port with a unique location, or a local inner name with at least one location in H. From the scope discipline we can see that, to make x local

in I, we must localise it to at least every region of I that contains a location of a point linked to x. Denote this set of regions by $\tilde{\imath}_x$. We have thus defined the *localisation of* x *in* F :

$$(x)F : H \to (\tilde{\imath}_x, x)I \ .$$

The scope discipline is still obeyed if we also locate x at regions that contain no location of a point linked to x. This leads to a stronger form of localisation, replacing $\tilde{\imath}_x$ by the width of I:

$$((x))F : H \to (n, x)I \text{ where } n = \mathsf{width}(I) \ .$$

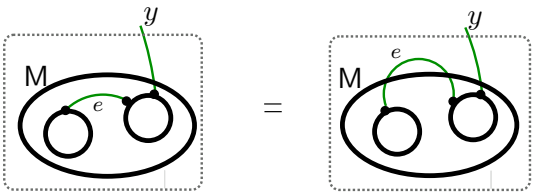
It remains to be seen whether we need both these forms.

A special case of localisation in F is when all its inner names H are local. Thus, by taking $H = \epsilon$, we see that localisation is always possible for ground bigraphs. This will be useful later in the treatment of reaction rules.

2 Binding of names

You will have noticed that localising a link, e.g. by $(x)F$, does not merely close the link as $/x \circ F$ does. Closing x in F creates a closed link or edge e, and this makes x inaccessible to any environment into which $/x \circ F$ is inserted by composition.

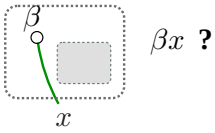
But this is not enough. The points linked by an edge e have locality, but the edge itself has not. Indeed, suppose M is non-atomic and F has a global outer name x; then we find that $\mathsf{M}.(/x \circ F) = /x \circ (M.F)$ (and this holds whether x global or local). A diagram makes this clear:



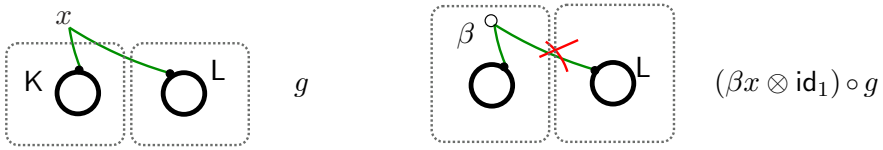
This becomes serious if, for example, a reaction rule copies the contents of an M-node. If we consider the edge e to be within the node, then each copy will have its own copy of e; if not, then all copies will be linked together by a link edge e. But the equation is undeniable; hence the copying is ill-determined.

Binding So, to determine the copying of linked structures, we need a form of closure that has locality. We want something that *binds* a name x, i.e. both closes x and confines the use of x to its own region. So we admit a new kind of control called a *binding*. If β is a binding, then we might write the binding of x by a β-node as βx; it functions both as a place and as a link.

Formally a β-node is an atom with arity 1; it has both a place and a link. We shall draw it as a little circle (thus distinguishing it from other atoms). We might attempt to bind a global name by a little pure bigraph $\beta x : \langle 1, x \rangle \to 1$, drawn thus:



But this will not work; we quickly see that it can be made to 'bind' a port outside its own region:



Therefore it is necessary to bind only a *local* name x. For this purpose, suppose I is an interface in which the name x is located at the regions $\imath_x$. Then define a small bigraph

$$\beta x : I \to /xI$$

where $/xI$ means I with name x removed; βx is the identity id_I except that the link to x is replaced by a link to a β-node, and this binding node is located at all the regions $\imath_x$. Of course this is too general for standard bigraphs, in which each node has exactly one location; so, for the present, when binding x we require it to be located only at a single region. (The general case, where a bound link may be shared among several regions, must be considered in the context of **Seminar C: Dags as place graphs**, where children having several parents are investigated.)

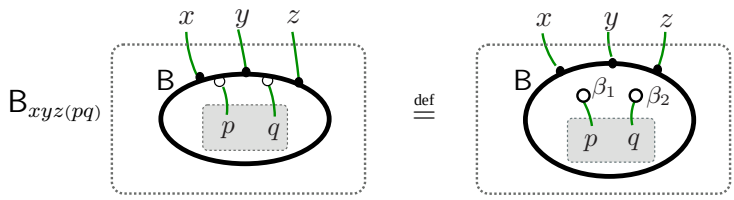
Thus we have succeeded in defining a form of link closure, called binding, which has locality. Often we shall wish to combine localisation with binding, i.e. to bind a global name by first localising it. So the following abbreviation is useful:

$$\beta(x)F \stackrel{\text{def}}{=} \beta x \circ (x)F \ .$$

It remains to refine the scope discipline of local bigraphs to account for bindings. For this purpose, for $F : I \to J$ we need only extend $loclink_F$ as follows:

$$\begin{aligned} locbind_F &\stackrel{\text{def}}{=} \{(prnt(v), p) : v \text{ a binding node with port } p\} \\ loclink_F &\stackrel{\text{def}}{=} locbind_F \uplus loc_J \ . \end{aligned}$$

Outward and inward binding We have defined the locality of a β-binding to be its parent place; we may thus call β an *outward binding* control. But we may need nodes that bind *within* themselves. By a simple sorting discipline, as in Section 6 of the book, this *inward binding* can be achieved by nesting a number of bindings inside an ordinary node. These can be ordered by using outward binding controls such as $\beta_0, \beta_1, \ldots$. If K has arity k and we equip a K-node with h binding controls, then we have turned the K-node into an inward binding control with a *dual* arity, written $h \to k$. The diagram shows the case $h = 2$, $k = 3$:

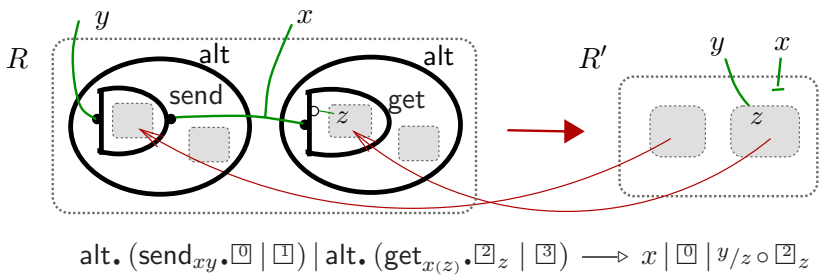


The π-calculus Let us now illustrate binding in the encoding of the finite π-calculus. The basic signature differs slightly from the one used in the book to encode CCS, since we must cater for the passage of names as data. The controls 'send' and 'get', previously both with arity 1, now have dual arities written $\mathsf{send} : 0 \to 2$ and $\mathsf{get} : 1 \to 1$. Thus 'get' becomes an inward binding control, yielding an inward-binding ion such as $\mathsf{get}_{x(z)}$.

Recall that the reaction rule in the π-calculus is written

$$(\overline{x}y.P + A) \,|\, (x(z).Q + B) \longrightarrow P \,|\, \{y/z\}Q \;.$$

The diagram below represents this in binding bigraphs:



Note how the meta-syntactic substitution of π-calculus is encoded by a substitution which is itself a bigraph; it substitutes a global name y for the local name z in site $\square_2$. The nesting operation in binding bigraphs is adjusted quite simply, to ensure that local names remain local to the 'nest', while global names are still exported from it.

Reaction rules As we see in the above rule for the π-calculus, the notion of a parametric reaction rule (R, R', η) is slightly affected by localities and binding. We consider five questions:

- *Must the parameter d of a rule be discrete, as in pure bigraphs?*
  We have to refine the notion of discrete, as in [48]. If X is a set of names, we define d to be *discrete for* X if every name in X is linked to exactly one port in d. A parameter d must not be required to be discrete for a name z bound by a node of R (as in the π-calculus rule), since z may be arbitrarily linked within such a node.

- *Can a rule have local inner names?*
  As illustrated above for the π-calculus, we want a rule to be able to bind names. So a redex R will have inner names $X \uplus Z$, with Z local and X are global; then we will typically require a parameter d for the rule to be discrete for X.

- *Can a rule have local outer names?*
  There is some freedom. Here we make the outer names of R and R' global, but we close the generated *ground* rules under localisation. Let (r, r') be a ground rule with outer face I and names X; then so is $(x)(r, r')$ for any $x \in X$. But this localisation must allow that r and r' differ in the locations of points linked to x (because for example the rule may change the region of a node). So, taking $\tilde{\imath}$ to be the sites of I within which a point of x is located

by *either* r *or* r', we define $(x)(r, r') \stackrel{\text{def}}{=} ((\tilde{\imath}, x)r, (\tilde{\imath}, x)r')$. Further research must be done to confirm that this indeed yields all the necessary ground rules.

Rules with local names are needed in Section 3, to model any redex of the λ-calculus whose free names are bound in its environment.

- *How are the parameter's names exported in the formation of a ground rule?*
  As in pure bigraphs, we define a ground redex to be $r = (R \otimes \mathsf{id}_X) \circ d$, where d is the parameter and X are its global names.

- *What about a rule that copies (part of) its parameter?*
  Here it is essential that copying a bound link creates a new bound link, rather than one shared between the original and the copy. A good example is the replication rule of the π-calculus. There it is treated as an axiom of structural congruence, $!P \equiv P \mid !P$. But we can also treat it as a reaction rule, or unfolding rule (as in Seminar Note A):

$$\mathsf{Master}.d \hookrightarrow d \mid \mathsf{Master}.d \ .$$

  For example, d may be a program script held by the Master node, from which copies may be repeatedly spun off for execution. Such a script will contain bound links, so binding within a parameter is essential. As noted earlier, name closure does not serve this purpose.

Jensen [46] adopted a different approach in his handling of the π-calculus. I believe both approaches are valid; the one I adopt here stays somewhat closer to that proposed in my book.

Sortings for localisation and binding Some simple enrichments of bigraphs were studied in my book; in particular, *place sortings* in which sorts are assigned to places, and *link sortings* in which sorts are assigned to links and points, in each case under a discipline. It was noted that in each of these cases there is a forgetful functor of s-categories from the sorted BRS to pure bigraphs.

Birkedal, Debois and Hildebrand [14] have proposed a more general notion of sorting; they define a sorting to be a functor $\mathcal{F} : \mathbf{A} \to \mathrm{B}_{\mathrm{G}}(\mathcal{K})$ of s-categories which is surjective on objects and injective when restricted to each homset.[8] Of course, we shall often want $\mathbf{A}$ to be some enrichment of bigraphs, and indeed by taking $\mathbf{A}$ to be local bigraphs we obtain a sorting in this wider sense. This also works for binding bigraphs.

The advantage of the wider canvas is that one can study the properties of sortings in an abstract way. For example, under what condition does a sorting $\mathcal{F}$ to create RPOs? That is, for an arbitrary relative bound in $\mathbf{A}$, when can one construct an RPO whose image is an RPO in $\mathrm{B}_{\mathrm{G}}(\mathcal{K})$? This was shown possible by Jensen and Milner [48] for binding bigraphs, when names have unique locality, and is easily generalised for multiple locality. By working in the proposed wider setting one can find general conditions on a sorting functor under which RPOs can be created.

[8] The second condition is called *faithfulness* in category theory.

3 Encoding λ-calculus

The π-calculus employs binding, and we have seen that the encoding of its reaction rule in binding bigraphs is a rather simple reaction rule. Reaction in the λ-calculus is more complex in two ways. First, it substitutes an arbitrary term for a bound variable, while the π-calculus simply replaces a bound name by a name. Second, reaction in the λ-calculus may occur at an arbtrarily deep level of binding, whereas in the π-calculus it can only occur at top level; this is achieved in bigraphs by making the get control passive.

The standard syntax for the λ-calculus is

$$M \quad ::= \quad x \;\big|\; \lambda x M \;\big|\; MN$$

and its reduction rule is $(\lambda x M)N \longrightarrow\!\triangleright \{N/x\}M$. This, in a single step, replaces the term N for all occurrences of x within M. The right-hand side of this rule is not a term, but denotes the term that results from performing the substitution.

Explicit substitutions We follow the Calculus of Explicit Substitutions of Abadi *et al*[9] which I shall call ABCL. It adopts a fourth term construction

$$M \quad ::= \cdots\cdots \;\big|\; M\,[x{:=}N]$$

which involves the *explicit substitution* $[x{:=}N]$. They also add reduction rules allowing this substitution to propagate itself throughout M, finding each occurrence of x and replacing it by N.

We shall adopt ABCL's syntax, and its strategy of performing the substitution separately for each occurrence of x in M. But, instead of propagating the substitution, we shall perform it by 'action at a distance', reflecting the fact that each occurrence of x may be arbitrarily deep in M. This strategy yields a calculus called Λ_{sub}, whose reduction rules are as follows:

$$\begin{array}{rcll}
(\lambda x M)N & \longrightarrow\!\triangleright & M\,[x{:=}N] & \\
(\{x/y\}M)\,[x{:=}N] & \longrightarrow\!\triangleright & (\{N/y\}M)\,[x{:=}N] & \text{where } M \text{ has a unique free occurrence of } y \\
M\,[x{:=}N] & \longrightarrow\!\triangleright & M & \text{where } M \text{ has no free occurrence of } x\,.
\end{array}$$

In the second rule, $\{x/y\}M$ distinguishes a particular occurrence of x to be replaced by N. (This occurrence may even lie within another explicit substitution!) The rules together achieve the result of the standard reduction rule. To demonstrate this requires careful analysis, which has indeed been carried out.[10] The calculus provides a natural challenge for modelling in bigraphs, where 'wide' reaction rules can be defined that do indeed represent 'action at a distance'.

Bigraphs for the λ-calculus We now define a bigraphical signature for Λ_{sub}. Its ions are shown in Figure 1. The node-shapes are merely to aid the eye. The signature is

[9] Abadi, M., Cardelli, L., Curien, P-L. and Levy, J-J. (1991), Explicit Substitutions. Journal of Functional Programming 1(4), pp375–416.

[10] Milner, R., Local bigraphs and confluence: two conjectures (extended abstract), ENTCS eo175-3, Article 4, 2007.

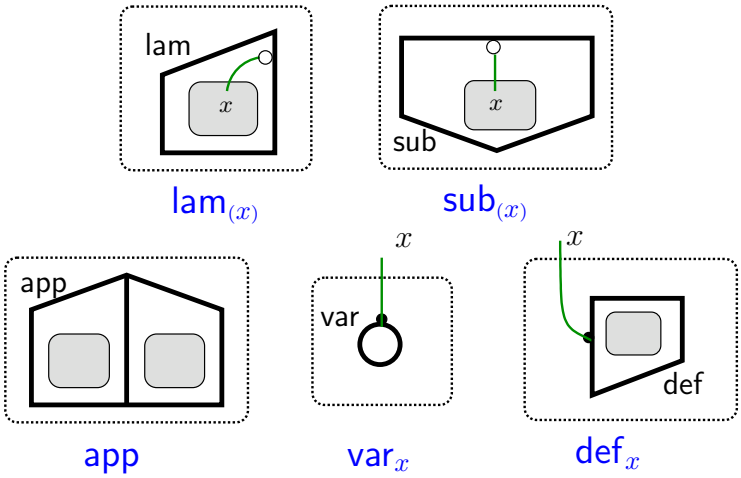


Figure 1: Bigraphical ions for λ-calculus

$$\mathcal{K}_\lambda = \{\mathsf{lam} : 1 \to 0,\ \mathsf{sub} : 1 \to 0,\ \mathsf{app} : 0,\ \mathsf{var} : 1,\ \mathsf{def} : 1\}$$

where var is atomic and the others are all active. The controls lam and sub are inward binding (hence their dual arities), and app is a two-place control.

We shall now translate Λ_{sub} terms into bigraphs over $\mathcal{K}_\lambda$. Consider $\lambda x\, xy$; First we form $\mathsf{app}\bullet(\mathsf{var}_x \parallel \mathsf{var}_y)$, then we localise x and nest the result in lam; this yields $\mathsf{lam}_{(x)}\bullet(x)(\mathsf{app}\bullet(\mathsf{var}_x \parallel \mathsf{var}_y))$. Thus a global name must be localised before a lambda abstraction is performed. This suggests that we translate every term of Λ_{sub} with free variables X into the homset $\epsilon \to \langle 1, X\rangle$. In fact we translate into each such homset all Λ_{sub} terms whose free names are among X. we define inductively the translation functions $[\![M]\!]_X$, for each finite set X of names, as follows:

$$\begin{array}{rcl} [\![x]\!]_{X \uplus x} & \stackrel{\text{def}}{=} & \mathsf{var}_x \mid X \\ [\![\lambda x M]\!]_X & \stackrel{\text{def}}{=} & \mathsf{lam}_{(x)}\bullet(x)[\![M]\!]_{X \uplus x} \\ [\![MN]\!]_X & \stackrel{\text{def}}{=} & \mathsf{app}\bullet([\![M]\!]_X \parallel [\![N]\!]_X) \\ [\![M[x{:=}N]]\!]_X & \stackrel{\text{def}}{=} & \mathsf{sub}_{(x)}\bullet(x)([\![M]\!]_{X \uplus x} \mid \mathsf{def}_x \bullet [\![N]\!]_X)\ . \end{array}$$

In the first clause, note that X is used to mean the ground bigraph no nodes and outer face $\langle 0, X\rangle$. In the second and fourth clauses, note that we must localise the name x before embedding a global bigraph into an ion that binds x. Of course, a notational convention can be defined to avoid repeating (x) in such terms.

Note that each term of Λ_{sub} with free names X has a translation-image in every homset $\epsilon \to \langle 1, X'\rangle$ such that $X \subseteq X'$. This multiplicity is harmless. Also, it's likely that $\mathcal{K}_\lambda$ can be submitted to simple sorting discipline Σ_λ, yield a bigraphical category $\mathrm{BG}(\Sigma_\lambda)$ that excludes all 'junk' from such homsets, i.e. the translation is surjective on each such homset. We leave this conjecture open; you may find it interesting to explore. One property of the translation is worth noting (the proof is routine): alpha-convertible Λ_{sub}-terms have equal images.

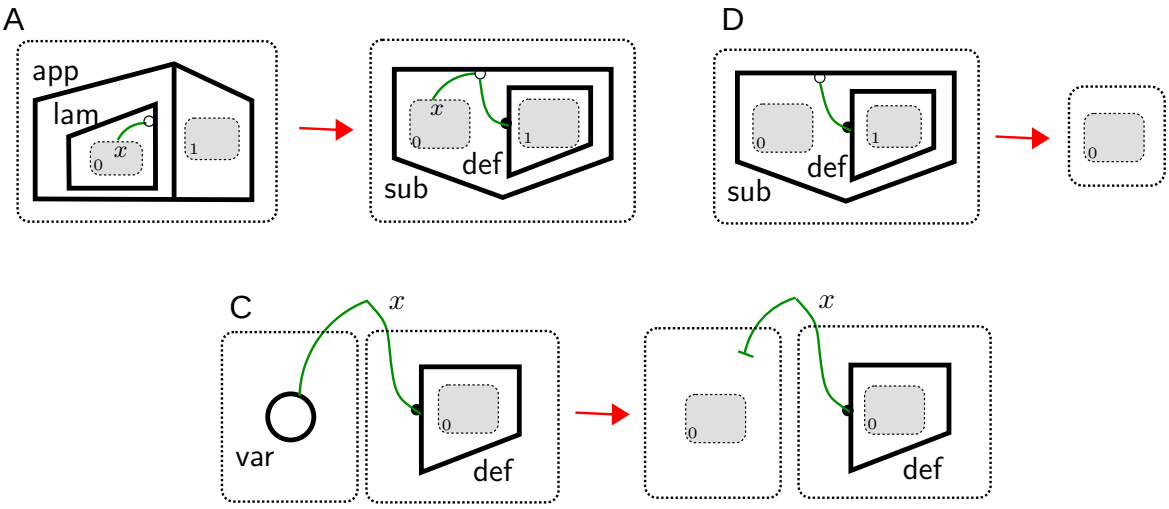


Figure 2: Parametric reaction rules $\mathcal{R}_\lambda$ for BG_λ

Bigraphical reaction rules for λ-calculus We now add three parametric reaction rules $\mathcal{R}_\lambda$ to $\mathrm{BG}(\Sigma_\lambda)$, forming a BRS BG_λ. These rules model those of Λ_{sub}, and are shown in Figure 2. The rules are named A, C and D for *apply*, *copy* and *destroy*. Note that the index of each site in the reactum indicates the parameter factor that should occupy it. Here are the rules in algebraic form:

$$\begin{array}{lrcl} \mathsf{A}: & \mathsf{app}.(\mathsf{lam}_{(x)}.\boxed{0}_x \parallel \boxed{1}) & \longrightarrow & \mathsf{sub}_{(x)}.(\boxed{0}_x \mid \mathsf{def}_x.\boxed{1}) \\ \mathsf{C}: & \mathsf{var}_x \parallel \mathsf{def}_x.\boxed{0} & \longrightarrow & \boxed{0} \parallel \mathsf{def}_x.\boxed{0} \\ \mathsf{D}: & \mathsf{sub}_{(x)}.(\boxed{0} \mid \mathsf{def}_x.\boxed{1}) & \longrightarrow & \boxed{0} \end{array}$$

Rule A can create an explicit substitution whenever (in Λ_{sub}terms) the operator M of an application MN is a λ-abstraction. Rule C is 'action at a distance'; note that it will always be applied within a sub node, with x bound, so it is essential that ground reaction rules are closed under localisation. Rule D handles the case when there is no further occurrence of the variable to be substituted; indeed, the scope discipline directly requires that, for a sub-molecule to be a D redex, the occupant of its first site may contain no occurrence of the variable to be substituted.

Certain λ-calculus reduction strategies can be easily modelled. For call-by-value, one simply creates two rules from A, in which site 1 is replaced respectively by a lam-ion or a var-node. For the lazy λ-calculus, one merely makes the control lam passive.

We leave it for further research to check that BG_λ faithfully models Λ_{sub}. It is also possible that BG_λ is a special instance of a BRS whose reaction relation is confluent. Some work in this direction has already been done (*op. cit.*) but for a different embedding of the λ-calculus in which all names were taken to be local. That treatment needed to introduce new operations on bigraphs. Here, in addition to the notions of localisation and bindings, we have only needed to introduce the localisation operations $(\tilde{\imath}, x)$.

Seminar Note C: Dags as place graphs

Robin Milner, 2009

The first aim of this note is to settle on the algebraic laws for dag place graphs, and also to settle on a working algebraic syntax which users can be comfortable with. I've discussed it with Michele Sevegnani, Søren Debois and Marcelo Fiore.

Section 1 presents the algebra for dag place graphs as an extension of what we now have for forest place graphs. Section 2 proposes conventions for notation and diagrams, advocating *stratified* diagrams in favour of the familiar *nesting* diagrams. Section 3 presents an application being studied by Michele Sevegnani. Finally, Section 4 discusses issues raised by dag place graphs, especially for reaction rules.

1 Algebraic laws

We now propose equational laws for dag place graphs.[11] First we assume the usual laws for the symmetries $\gamma_{I,J}$ as in my paper *Axioms for bigraphical structure*.[12] In particular $\gamma_{1,1}$, also denoted by $swap$, just interchanges two sites (or roots). The other laws in that paper are the monoid laws for $1 : 0 \to 1$ and $merge : 2 \to 1$, as follows:

$$\textbf{monoid} \ : \ \begin{cases} merge \circ (1 \otimes \mathsf{id}_1) & = & \mathsf{id}_1 & \text{(unit)} \\ merge \circ (merge \otimes \mathsf{id}_1) & = & merge \circ (\mathsf{id}_1 \otimes merge) & \text{(associative)} \\ merge \circ \gamma_{1,1} & = & merge & \text{(commutative)} \end{cases}$$

We generalise $merge$ to $merge_m : m \to 1$ by induction on m:

$$\begin{array}{rcl} merge_0 & \stackrel{\text{def}}{=} & 1 \\ merge_{m+1} & \stackrel{\text{def}}{=} & merge \circ (\mathsf{id}_1 \otimes merge_m) \end{array}$$

It is readily seen from this definition that $merge_2 = merge$.

Now $split : 1 \to 2$ is the dual of $merge$, and is the only extra operation we need for dags. We add the co-monoid laws for $0 : 1 \to 0$ and $split$:

$$\textbf{co-monoid} \ : \ \begin{cases} (0 \otimes \mathsf{id}_1) \circ split & = & \mathsf{id}_1 & \text{(unit)} \\ (split \otimes \mathsf{id}_1) \circ split & = & (\mathsf{id}_1 \otimes split) \circ split & \text{(associative)} \\ \gamma_{1,1} \circ split & = & split & \text{(commutative)} \end{cases}$$

and we define $split_m : 1 \to m$ by induction on m:

$$\begin{array}{rcl} split_0 & \stackrel{\text{def}}{=} & 0 \\ split_{m+1} & \stackrel{\text{def}}{=} & (\mathsf{id}_1 \otimes split_m) \circ split \end{array}$$

[11] I am grateful to Marcelo Fiore for pointing out what we need in addition to those for forests. The proof of their completeness needs to be confirmed, but we expect no difficulty in that.

[12] MSCS 15, 2005, pp1005–1032. By an oversight the paper omitted one law in Table 1; the law is $\gamma_{I \otimes J,K} = (\gamma_{I,K} \otimes \mathsf{id}_J) \circ (\mathsf{id}_I \otimes \gamma_{J,K})$.

where of course (dually) we find that $split_2 = split$.

The question then is: how do we relate $merge$ and $split$? Marcelo drew our attention to two further axioms for $merge$ and $split$, using $swap$ for $\gamma_{1,1}$:

$$\begin{array}{rrcl} \textbf{degeneracy}: & merge \circ split & = & \mathsf{id}_1 \\ \textbf{bialgebra}: & split \circ merge & = & (merge \otimes merge) \circ (\mathsf{id}_1 \otimes swap \otimes \mathsf{id}_1) \circ (split \otimes split)\ . \end{array}$$

There are also three axioms involving 0 and 1:

$$0 \circ 1 = \mathsf{id}_0 \qquad 0 \circ merge = 0 \otimes 0 \qquad split \circ 1 = 1 \otimes 1\ .$$

Fiore points out that the free symmetric monoidal category generated by these together with the monoidal and co-monoidal laws is known to be the category of relations between finite ordinals.

We call a node-free place graph a *placing*. For current pure bigraphs, a placing $\phi : m \to n$ is just a *function* from $m = \{0, 1, \ldots, m-1\}$ to $n = \{0, 1, \ldots, n-1\}$; this is the free interpretation of the laws for the symmetries and $merge$. What is the analogous free interpretation of the extra laws we have added? Fiore points out that it is known to consist of binary *relations* $\phi \subseteq m \times n$.

What happens when nodes are added? More precisely, we add to our algebra a generator in the form of a single node $v : 1 \to 1$. So what then is the free interpretation of the axioms? We strongly conjecture[13] that a place graph $P : m \to n$ with nodes V consists of a binary *relation* $P \subseteq (m \uplus V) \times (V \uplus n)$ that is acyclic, i.e. if $v \in V$ and $(v, v) \in \phi^k$ then $k = 0$. This perfectly matches the definition of forest place graphs, except that the relations are functions in that case. The composition of dag place graphs is in obvious analogy with composition on forest place graphs.

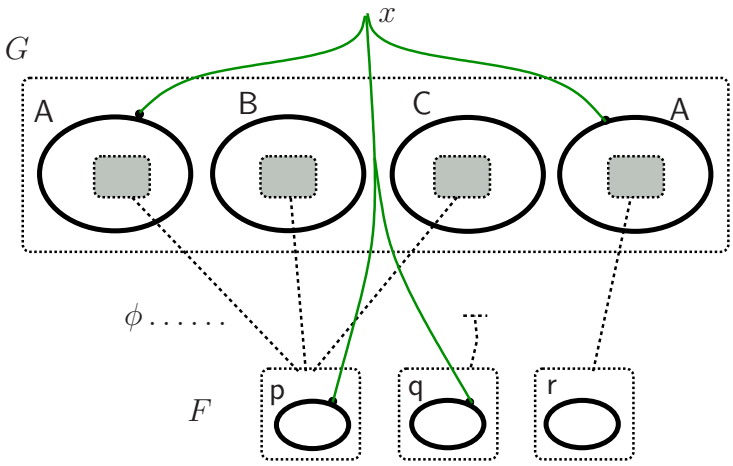
For a place graph $P : m \to n$ with nodes V, let $w \in m \uplus V$ and $w' \in V \uplus n$. We define w to be *child* of w' if $(w, w') \in P$. Childhood is not (necessarily) transitive; if w'' in turn is a child of w' it may or may not be a child of w. Define the *descendant* relation P^* to be the transitive reflexive closure of P. A sorting may impose discipline on both childhood and descendancy, e.g. that two distinct regions of a certain sort may not share a child, or may not share a descendant. For example, two rooms in a building may not share a descendant, but the range of a mobile sensor may share a child with a room or with the range of another sensor. As usual, we require that reaction rules preserve the sorting discipline.

2 Diagrams and notation

The general case in which several sites of a bigraph G may 'share' a component of F is when we have $F : I \to J'$ and $G : J \to K$, with $J' = \langle n', X \rangle$ and $J = \langle n, X \rangle$. For $n' \neq n$ we can't form $G \circ F$, but we can form $G \circ \phi \circ F$ allowing parts of F to be shared by G.

In general J' will be non-prime, and we may have the different regions of F split among different numbers of the sites of G. In that case, we must specify the placing to be used. As an example, consider the case $J' = \langle 3, x \rangle$ and $J = \langle 4, x \rangle$; for specific bigraphs F and G we can represent $G \circ \langle \phi, \mathsf{id}_x \rangle \circ F$ by a *stratified* diagram that makes the placing $\phi : 3 \to 4$ explicit, as follows:

[13] We believe the proof is routine



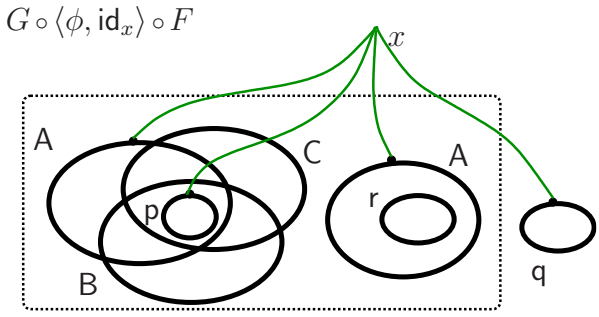
We can conveniently write ϕ as a vector of three subsets of $4 = \{0, 1, 2, 3\}$:

$$\phi : 3 \to 4 = \{\{0, 1, 2\},\ \emptyset,\ \{3\}\}\ .$$

The region 0 of F is shared by the sites $0, 1$ and 2 of G, while the region 2 of F is inserted only at site 3 of G. The region 1 of F is 'closed', and has *no* place! But its nodes could to be linked via x, and some of its inner nodes could even be shared by the other regions of F. In general, we might write $G \circ \langle \phi, \mathsf{id}_X \rangle \circ F$ in programming style as

$$\text{share } F \text{ by } \phi \text{ in } G\ .$$

If course, we may choose not to make ϕ explicit, and be content with a nesting diagram which represents sharing by overlapping of nodes (or of sites or regions). This recalls Venn diagrams, and for the above example it is not bad:



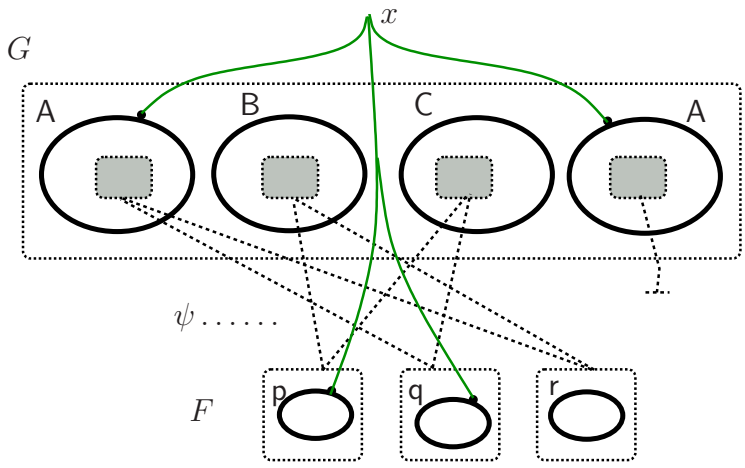
But there are disadvantages. First, note that certain 'regions' in the diagram, e.g. the region shared by the B- and C-nodes but not by an A-node, are empty. Such a fictitious region means nothing. It would mean something if it contains a site; then, via composition, it may come to contain nodes. But in a stratified diagram such 'empty' regions do not arise naturally. Second, a nesting diagram does not naturally represent how a node may share a child with one of its descendants.

place closure The closure of places – as for the middle region of F above – may seem strange to those familiar with bigraphs. But it resembles the closure of links in link graphs; just as the latter makes a name inaccessible to the outside, so the former makes a root or region inaccessible to the outside. Far from causing problems, I believe place closure will add useful expressive power.

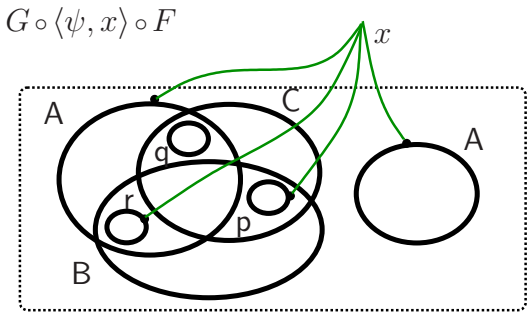
placings that both merge and split Given the same F and G and in the previous example, we could form a shared composition via a different placing

$$\psi : 3 \to 4 = \{\, \{1, 2\},\ \{2, 0\},\ \{0, 1\} \,\} \,.$$

Since these subsets are not disjoint, ψ involved merging as well as splitting. Here is a stratified diagram analogous to the first diagram for the preceding example:



Again, as an alternative, there is a tolerable nesting diagram:



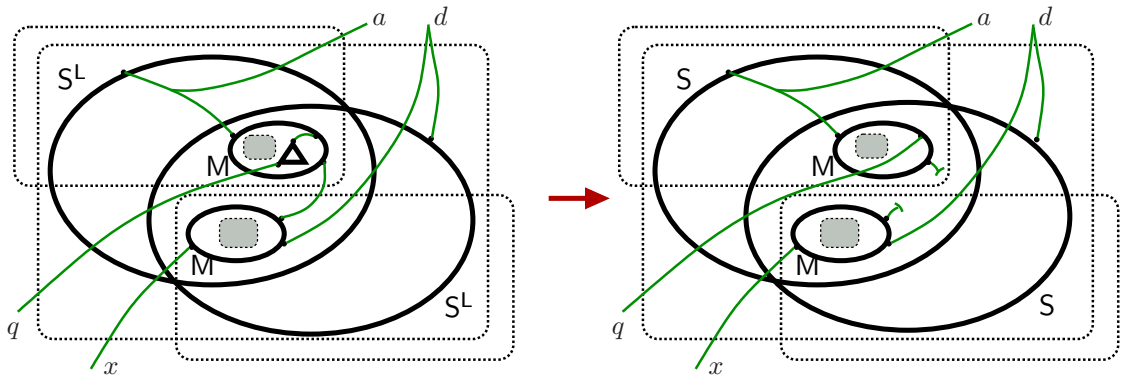
3 Application

As an experiment in the use of dags as place graphs, Michele Sevegnani has modelled the 802.11 CSMA/CA protocol. We do not report the whole model here,[14] but confine ourselves to reproducing (with his permission) the diagrams for the reaction rule that represents the last of the five phases

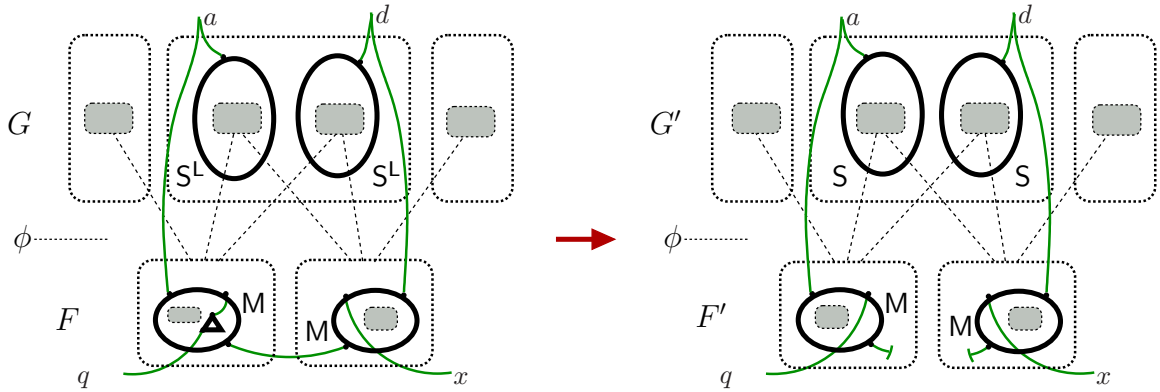
[14]Michele Sevegnani can be contacted at michele@dcs.gla.ac.uk .

of the protocol. The relevant controls are M for a wireless station and S for the range of a station's signal. The latter changes its state to S^{L} when a station's signal is locked onto another station. The five phases are (1) a sender senses a channel free and becomes ready to transmit an RTS (request to receive) packet; (2) it transmits this packet, in the absence of collisions; (3) the receiver replies with a CTS (clear to send) packet; (4) the data packet is sent; (5) the receiver acknowledges with an ACK packet.

In the fifth and last phase the sender and receiver stop transmitting and the locks on the channel are dropped. The following nesting diagram depicts this phase. the sender named a is the upper station (M), the receiver named d is the lower, and the data packet is the triangular node. The sender discards the data packet, while the receiver has further transmitted the received copy to where it will be processed. The dropping of locks is represented both by the loss of a link between the two stations (M) and by the change of state of the signal range from locked (S^{L}) to unlocked (S).



The stratified version of the diagram is somewhat simpler:



This may be written as

$$\text{share } F \text{ by } \phi \text{ in } G \longrightarrow \text{share } F' \text{ by } \phi \text{ in } G' \,.$$

In modelling the protocol Sevegnani gives stochastic rates (see Note E) to the five rules. This opens the way to running simulations of the protocol. The rigorous algebraic presentation also

opens the way to formal analysis of the protocol, either by model-checking or by other mathematical means. To be tractable these methods must be mediated by software tools. Projects to design and implement such tools are under way.

4 Discussion

link graphs with aliases We have left link graphs unchanged. But having made place graphs self-dual we may wish to do the same for link graphs. This only means introducing *aliases*; that is, a link can have any number of outer names. (Already, closure allows it to have *no* outer names.)

theoretical strategy We have already conjectured what the axioms of dag bigraphs should be, and this should be not too hard to confirm. When I considered dag place graphs – many years ago – there were four reasons for not then adopting them:

1. I was concerned that diagramming them was not so straightforward.
2. They appeared unnecessary for encoding process calculi.
3. Relative pushouts (RPOs) do not always exist in the presence of dag place graphs, and they seem essential to handling behavioural equivalence in bigraphs.
4. To discover that something is necessary, see how far you can go without it!

The third reason is not strong: RPOs do exist for any pair $\vec{A}$ of place-sharing bigraphs in which no two roots share a descendant and no root is idle. Thus we may find that behavioural equivalence is tractable for some place-sharing BRSs.

The fourth reason was strategic. As a result, we are now confident that some applications *demand* dag place graphs, and therefore we are right to include them provided that they do not create serious difficulty.

binding bigraphs We still have to analyse the impact of dag place graphs on the theory of *binding* bigraphs, where some links (those that represent name-binding) are confined to points that lie in a certain region.

reaction rules Recall that a parametric reaction rule with redex R, generates a ground reaction rule $r = (R \otimes \mathsf{id}X) \circ d$ for each parameter d. A reaction rule may discard or replicate factors of d; we have to be clear how this affects factors that share nodes.

conclusion As can be seen from above, analysis is still needed for aspects of place-sharing in bigraphs. For this, experimental applications are valuable; they may influence the general definition of dynamics.

Seminar Note D: Linked data structures

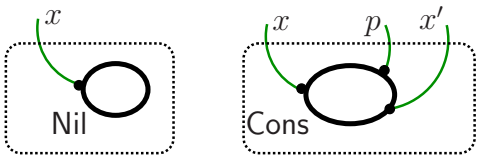
Robin Milner, 2009

In this note we represent data by *linking*, rather than by *nesting* as in Note A. We begin by using *binding* as in Note B. We then show how data structures that share, e.g. two lists sharing a terminal sublist, can be simply encoded if bigraphs are enriched with *aliases* for outer names. As an application, we consider the indexing of arrays (of arbitrary data) by natural numbers; in can be done simply and efficiently with the numbers as linked structures.

1 Disjoint data structures

In Note A, a data constructor was a control with arity zero, and zero or more sites. For example, for list structures we had $\mathsf{Nil} : 0 \to 1$ and $\mathsf{Cons} : 2 \to 1$ Here instead we define a data constructor to be an atom with arity ≥ 1; it has a principal port, the name of the constructed datum, and zero or more subsidiary ports for linking to components.

In this section we stay close to Note A in one sense: different data structures are *disjoint*, i.e. they have no nodes in common.[15] Let us illustrate with the case of list structures. First, we define Nil and Cons as atomic controls with arities 1 and 3 respectively.



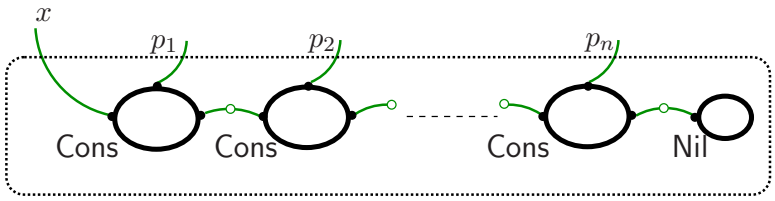
Instead of nesting constructors, we shall connect them with bound links. For this purpose we define an operator to create bound links:[16]

$$F \widehat{xy\cdots}\, G \stackrel{\text{def}}{=} \beta(x)\beta(y)\cdots(F \mid G)\ .$$

Thus the spine of a list of n elements named $p_1, \ldots, p_n$ will be

$$\mathsf{List}_{x\vec{p}} \stackrel{\text{def}}{=} \mathsf{Cons}_{xp_1x_1} \widehat{x_1}\, \mathsf{Cons}_{x_1p_2x_2} \widehat{x_2} \cdots \widehat{x_{n-1}}\, \mathsf{Cons}_{x_{n-1}p_nx_n} \widehat{x_n}\, \mathsf{Nil}_{x_n}\ ,$$

which is pictured thus, using a small circle to represent a bound link:



[15] In Section 2 we shall show how to model data structures that share elements.

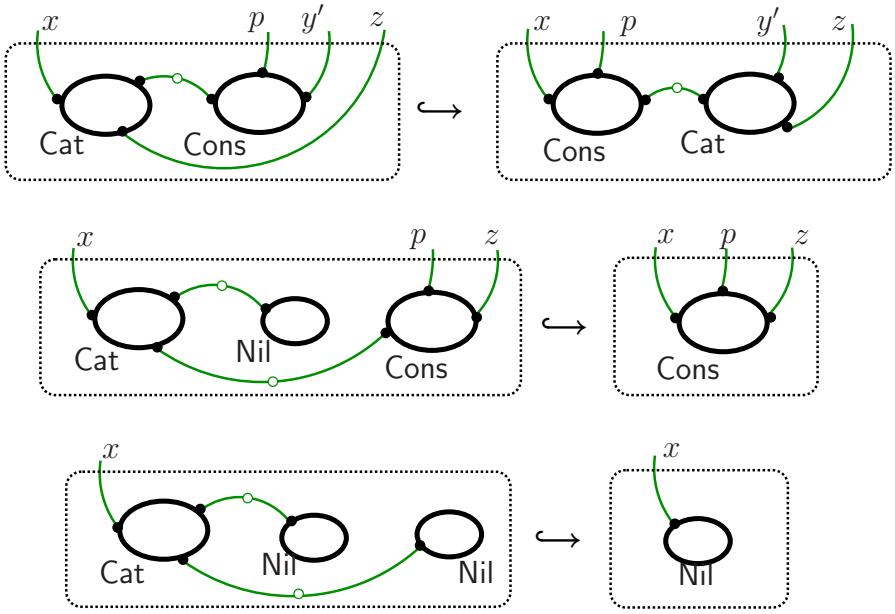
[16] Binding requires that F and G have no global inner names. The usage here will obey this condition.

The closure of names $x_1, \ldots, x_n$ prevents two lists sharing a terminal sublist. This effect is analogous to the effect of nesting the constructors, as in Note A.[17]

We are now ready for seeds. In this treatment of data a parametric does not nest its parameters, but has bound links to them. So the seed for concatenation is Cat_{xyz}, with principal port x, where y and z link to the two arguments. There are three germination rules:

$$\begin{aligned} \mathsf{Cat}_{xyz} \mathbin{\widehat{\ y\ }} \mathsf{Cons}_{ypy'} &\hookrightarrow \mathsf{Cons}_{xpx'} \mathbin{\widehat{\ x'\ }} \mathsf{Cat}_{x'y'z} \\ \mathsf{Cat}_{xyz} \mathbin{\widehat{\ yz\ }} (\mathsf{Nil}_y \mid \mathsf{Cons}_{zpz'}) &\hookrightarrow \mathsf{Cons}_{xpz'} \\ \mathsf{Cat}_{xyz} \mathbin{\widehat{\ yz\ }} (\mathsf{Nil}_y \mid \mathsf{Nil}_z) &\hookrightarrow \mathsf{Nil}_x \ . \end{aligned}$$

They are pictured as follows:



Just as in the germination rules of Note A for nested data constructors, operations defined in this way destroy their arguments. We therefore need the ability for a calculation to make copies of data values, perhaps from remote locations, in order to operate upon them. So we imagine that these copiable values are stored within an inward-binding control. Recall the controls Fetch and Val with rank 1 and arity 0, at the end of Section 2 in Note A. Here we give Fetch rank 0 and arity 2, and we give Val rank 1 and arity $1 \to 1$ (see Note B for binding arities); then we define a germination rule:

$$\mathsf{Fetch}_{xy} \parallel \mathsf{Val}_{x(z)}.\boxed{0}_z \hookrightarrow (y/z \circ \boxed{0}_z) \parallel \mathsf{Val}_{x(z)}.\boxed{0}_z \ .$$

This is close to the reaction rule in Note A. However, it differs in that the Val-node and its stored value are still present after the germination. The rule differs from previous germination rules because its 'seed' is non-prime.

Of course, similar but simpler rules work for the natural numbers, with constructors Zero and Succ with arities 1 and 2. We strongly conjecture that such germination rules generate an unfolding

[17] In a later section we shall show how *aliasing* of outer names permits shared substructures.

relation with the essential property that it is confluent and respects reaction. But a rigorous proof has not been given. Note that in the case of the Fetch–Val rule the proof must rest strongly on the fact that the stored value is preserved by reaction.

2 Contextual rules

In the final germination rule of the previous section, involving Fetch, the redex and reactum share a Val component. We would like to write it as

$$\mathsf{Val}_{x(z)}.\Box_z \;\|\!\vdash \mathsf{Fetch}_{xy} \hookrightarrow (y/z \circ \Box_z) \;.$$

This is a special case of a *contextual* reaction rule (in this case germination rule), as proposed at the end of Section 11.1 of my book. It adds no power to rules, since such a rule can also be written out with the context on both sides. But it distinguishes the part of a redex that changes from the part which is just a required context for the change. In this particular case we have chosen the symbol '$\|\!\vdash$', since the context is a parallel product; when it is a prime product we shall use '$\vdash$'. Contextual germination rules will occur often in the following sections.

It is important theoretically to make a rule's context explicit, because the proof that unfolding is confluent and respects reaction will depend crucially on the fact that a context persists unchanged through a reaction.

3 Shared structures

McCarthy's LISP was a ground-breaking advance in non-numerical programming. Those who have used LISP will not be satisfied with the non-shared lists of the previous section. We proposed non-shared structures partly because they are closely analogous to the nested ones of Note A, and partly because sharing – at least in an elegant form – seems to demand *aliasing* of names, which allows a link to have any finite number of outer names.

By the way, aliasing in link graphs is the analogue of dags as place graphs; see Note C. Just as a link with outer name x may also have outer name x' as an *alias*, so a node in a place w may also have another place w' as an *alibi*.

We shall not develop the theory of aliases here,[18] but will state their obvious properties and constructions as we need them. To begin with, the quintessential alias is written $x{=}y$, and all other aliasing – such as an A-node with two outer names, can be derived by parallel or prime products, e.g. $\mathsf{A}_x \,|\, x{=}y$:

[18] Some of it was done in an early tech report on bigraphs. Later, aliases were excluded because they were were not needed for encoding many process calculi, and they falsified the theorem that RPOs (which *were* needed) always exist in bigraphs. But from a more modern perspective they may turn out to be valuable. In any case, bigraphs without alibis and/or without aliases can be represented by a special sorting, and in general we expect the theories of sorted BRSs to refine, or strengthen, the unsorted theory.

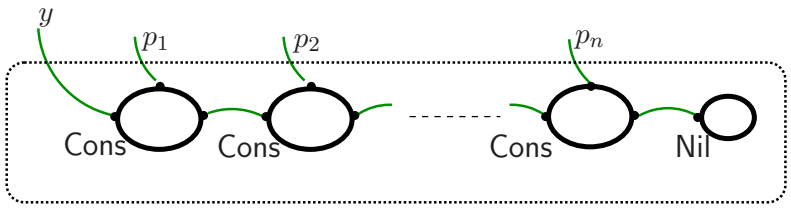
Now let us reconsider linked lists. We take the same constructors Nil and Cons, but we connect them differently. The reason is that, even if two Cons-nodes are connected by a link that is not open, the link may later be opened in order that a user can gain access to it and thereby share a terminal sublist. It is therefore pointless for the link to be *bound*; instead, we shall just *close* it. Indeed, with aliasing it is accurate to think of a closure $/x$ not as closing a link, but as removing the name x from it; if x has aliases then the link is still open, but with one name fewer.

To represent a structure (e.g. a list) with closed but not bound links, we define an operator analogous to $F \widehat{\ xy\cdots\ } G$:

$$F \overline{xy\cdots} G \stackrel{\text{def}}{=} /x/y\cdots(F \mid G)\ .$$

Thus the spine of a list of n elements named $p_1, \ldots, p_n$ now becomes

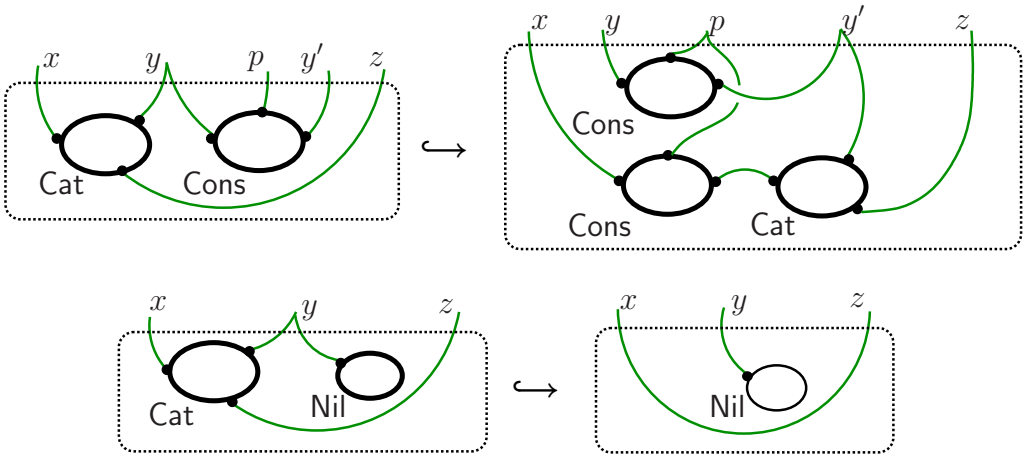
$$\mathsf{List}_{y\vec{p}} \stackrel{\text{def}}{=} \mathsf{Cons}_{yp_1y_1}\ \overline{y_1}\ \mathsf{Cons}_{y_1p_2y_2}\ \overline{y_2}\ \cdots\ \overline{y_{n-1}}\ \mathsf{Cons}_{y_{n-1}p_ny_n}\ \overline{y_n}\ \mathsf{Nil}_{y_n}\ .$$



Note that there is no little circle on the closed links. When a list is first created, probably all its links except the first will be closed, as here. But sharing can arise as the result of operations.

We now give germination rules for concatenation, in which the first list is copied but the second is shared by the result. We give the rules first in non-contextual form.

$$\begin{aligned} \mathsf{Cat}_{xyz} \mid \mathsf{Cons}_{ypy'} &\hookrightarrow \mathsf{Cons}_{ypy'} \mid \mathsf{Cons}_{xpx'}\ \overline{x'}\ \mathsf{Cat}_{x'y'z} \\ \mathsf{Cat}_{xyz} \mid \mathsf{Nil}_{y} &\hookrightarrow z{=}x \mid \mathsf{Nil}_{y}\ . \end{aligned}$$

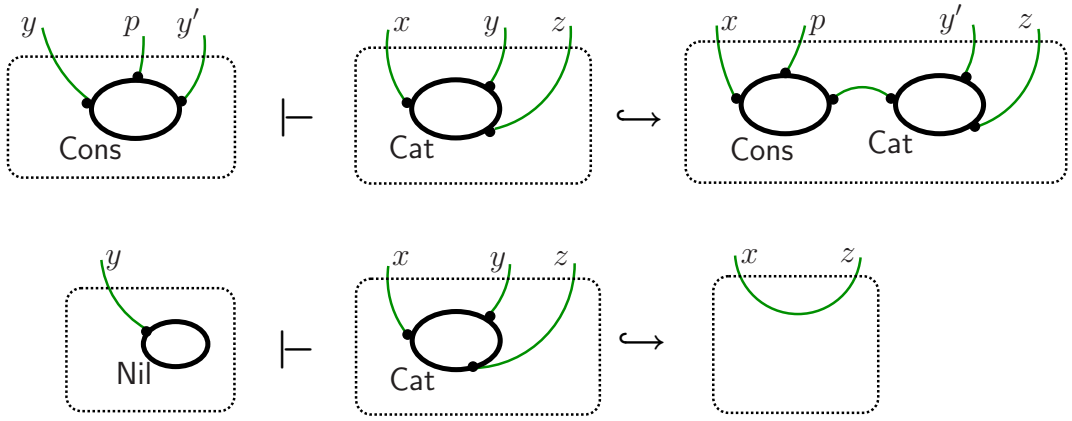


This definition raises many interesting points:

- In both redexes, the link y from Cat to an element of the first list is an open link, allowing for the possibility that the list – or its terminal sublists – may be shared.
- The first rule copies the first list, element by element, and the new copy has closed links (which may become shared after further use). On the other hand, the second rule merely creates a link from the end of the first list to the start of the second. The aliasing $z{=}x$ is useful for this purpose, though it can be avoided by copying the second list, as we did for lists with bound links.
- So far, we have not excluded multiple links from a Cons element to different ensuing sublists. But this would make the concatenation non-deterministic. To ensure that these links are unique, we impose a simple sorting to ensure that each such link contains at most one target port, where a *target* is the left-hand port of a Cons or Nil.
- From the point of view of LISP all we have done is to capture what goes on in *pure* LISP, where there are no rplaca and rplacd operations for assigning a new head or tail to a list cell. But for example rplaca is simple; it is only necessary to re-connect a cell's p-link.

It is revealing to see the text and diagrams for the above rules in contextual form:

$$\begin{aligned} \mathsf{Cons}_{ypy'} \quad &\vdash \quad \mathsf{Cat}_{xyz} \;\hookrightarrow\; \mathsf{Cons}_{xpx'}\, \overline{{}^{x'}}\, \mathsf{Cat}_{x'y'z} \\ \mathsf{Nil}_{y} \quad &\vdash \quad \mathsf{Cat}_{xyz} \;\hookrightarrow\; z{=}x\ . \end{aligned}$$



Finally, there are alternative contextual rules using '$\|\!-$' instead of '$\vdash$'. Such 'wide' rules can yield list structures that are straddled across many regions, but that seems to create no difficulty.

4 Indexing

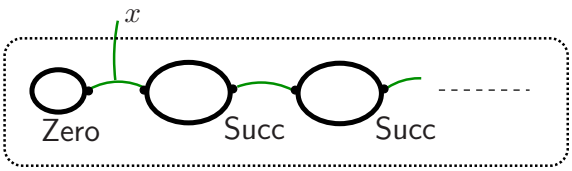
A simple shareable linked structure is the natural numbers. We wish to consider a calculational BRS in which there is a *unique* such structure, and in which natural numbers, especially those used

for indexing, can be represented simply by links into this unique structure. We leave aside the question of whether such a BRS can be defined as a sorting; but we shall deal with unfolding rules that preserve this unicity as an invariant.

So, analogous to the definition of Zero and Succ from Note A, let us redefine them as constructors with arities 1 and 2, and ∞ as a seed with arity 1. Then define

$$\infty_x \hookrightarrow \mathsf{Succ}_{xy}\,\overline{y}\,\infty_y \ , \ \mathsf{Nat}_x \stackrel{\text{def}}{=} \mathsf{Zero}_x \,|\, \infty_x \ .$$

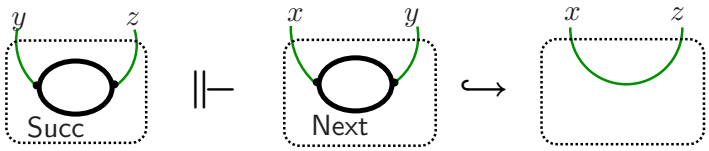
Nat_x looks like this, with all its links closed except one:



Our unfolding rules will work with a unique such structure. Moreover, the rules will preserve as an invariant that the only instances of Zero and Succ are within this unique structure. However, they will allow aliases to be created for closed links, thus opening them. For example, this can be done by the seed (not constructor) Next_{xy}, with the germination rule

$$\mathsf{Succ}_{yz} \ \|\!\!- \ \mathsf{Next}_{xy} \hookrightarrow x{=}z \ .$$

which may be pictured thus:



This does what would be written in a programming language as the assignment "$x := x + 1$".

It is convenient to introduce a special infinite class of names, the *numerals* 0,1,2,.... We then insist that we work with a unique structure Nat_0 for the natural numbers, whose links have been opened by unfolding $\mathsf{Next}_{1,0} \,\|\, \mathsf{Succ}_{0,z}$, then $\mathsf{Next}_{2,1} \,\|\, \mathsf{Succ}_{1,z}$ and so on, as far as needed. The numerals must be subject to a special discipline, to ensure that their links into the structure Nat_0 remain fixed. We omit details of this discipline; it will be assisted by a sorting (as already proposed for linked lists) but it remains to be seen whether the entire discipline can be expressed as a sorting.

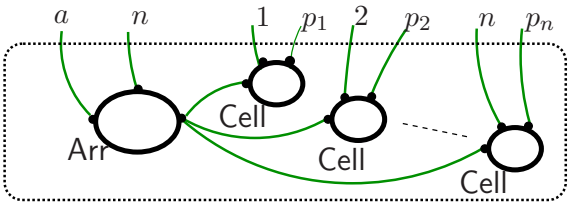
The reader may enjoy defining simple functions of natural numbers as seeds. For example $\mathsf{Plus}_{x,2,3}$, linked into Nat_0 by the numerals 2 and 3, should create the alias $x{=}5$. The germination rules for such functions are quite simple, especially for those who know primitive recursive arithmetic, but care is needed to treat the case in which two names are linked to the same Succ-node.

We now turn to arrays. The cells of an array will be indexed by the numerals. We introduce atomic controls $\mathsf{Arr} : 3$ and $\mathsf{Cell} : 3$. An array of size n, with name a and elements of arbitrary sort s, will be represented by Arr_{any}, where a is its name, n its size and y is linked to n cells

$\mathsf{Cell}_{y1p_1}, \ldots, \mathsf{Cell}_{ynp_n}$, where each p_i has sort s and names an array element. Thus the array structure is

$$\mathsf{Arr}_{any}\,\overline{y}\,(\mathsf{Cell}_{y1p_1} \mid \cdots \mid (\mathsf{Cell}_{ynp_n})$$

which can be pictured thus:



What reaction rules are needed for arrays? First, one must be able to assign a new element to any cell. This is a state-changing operation, and therefore it cannot be treated as an unfolding. It is easy enough to define a reaction rule for a control $\mathsf{Assign} : 3$, so that the atom Assign_{amq} assigns the name q of a new datum to the m^{th} cell of an array named a. We shall ignore this rule, and focus on germination rules that do not change the state of data, and are therefore likely to yield an unfolding relation that is confluent and respects reaction.

The first such rule is for fetching an array element. For this purpose we introduce the atomic seed $\mathsf{Fetch} : 3$, with germination rule

$$\mathsf{Arr}_{any} \mid \mathsf{Cell}_{ymp} \;\|\!\!-\; \mathsf{Fetch}_{amq} \hookrightarrow q{=}p\ ;$$

thus Fetch_{amq} makes the name q an alias for the m^{th} element in the array named a.

More ambitiously, suppose that we wish to iterate a binary operator over the elements of an array. We represent it as a seed $\mathsf{Op} : 3$, such that Op_{rpq} applies the operator to values (named) p and q and names the result r.[19] We suppose that the array elements – of whatever sort – are built by constructors, and that there are germination rules for Op over these constructors. We need not detail these rules; we shall merely define how to iterate Op over the array structure, unfolding it into a structure of Op-nodes, which will cause further unfolding when the array structure is linked to its elements.

We introduce two seeds $\mathsf{Iterop} : 3$ and $\mathsf{Appop} : 4$ that respectively iterate Op over an array and apply it to each individual element:[20]

- Given an array named a, the seed Iterop_{arq} initiates the iteration with a starting value q and delivers the final result at r.
- Given an array of size n linked at y to its elements, the seed Appop_{nyrq} applies Op to each element p and accumulated value q, delivering the result at r. It uses n to terminate.

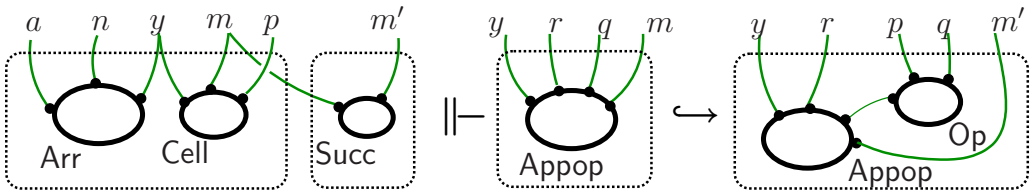
Here are the germination rules:

$$\begin{aligned}
\mathsf{Arr}_{any} \quad &\|\!\!-\quad \mathsf{Iterop}_{arq} \hookrightarrow \mathsf{Appop}_{yrq0} \\
\mathsf{Arr}_{any} \quad &\|\!\!-\quad \mathsf{Appop}_{yrqn} \hookrightarrow r{=}q \\
(\mathsf{Arr}_{any} \mid \mathsf{Cell}_{ymp}) \parallel \mathsf{Succ}_{mm'} \quad &\|\!\!-\quad \mathsf{Appop}_{yrqm} \hookrightarrow \mathsf{Appop}_{yrq'm'}\,\overline{q'}\,\mathsf{Op}_{q'pq}\ .
\end{aligned}$$

[19] We assume the operator to be associative and commutative, allowing the order of iteration to be freely chosen.

[20] Using binding, one can define generic seeds Iter and App that are parametric in Op; we ignore the details.

It is worth drawing a picture of third rule, which does all the work:



Note two things especially. First, the unfolding of Appop is unambiguous, since the its two rules cannot both apply to a given occurrence; for if $m = n$ then there can be no cell of the given array that is linked to m. Second, the rules work properly even when the data named by $p, q, \ldots$ are themselves natural numbers, i.e. $p, q, \ldots$ are links into Nat_0. In this case, Op could be Plus for example, and we are summing an array of natural numbers.

5 Conclusion

This seminar, representing data by linking, is more conjectural than Note A – especially in Section 4 where the data structures can be shared. This is partly because aliasing of outer names is used, though this was indeed studied in an early technical report by me, and omitted in later work because RPOs do not always exist in the presence of aliasing.[21]

Another reason is that it is unclear whether the discipline for shared data structures can be fully expressed as a sorting; it should be not hard to settle this question. Finally, work is needed to establish that the unfolding relation induced by the more adventurous germination rules of this seminar is indeed confluent and respects reaction. I strongly recommend this as a challenging project, perhaps for PhD study; it will greatly clarify the concept of a calculational BRS.

[21] *Bigraphical reactive systems: basic theory*, Technical Report 523, University of Cambridge Computer Laboratory, 2001. There are sufficient conditions for RPOs to exist, and these are satisfied by the encodings of process calculi, so there is no strong reason to preclude aliases; only that the theory is a little harder in their presence.

Seminar Note E: Measuring space and time

Robin Milner, 2009

In this note we look first, in Section 1, at how superpose cartesian space on our regions, which are hitherto only structured by containment and contiguity.

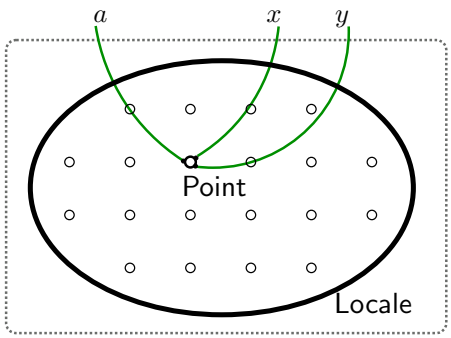
Next, in Section 2 we introduce stochastic behaviour, in which a non-deterministic BRS is refined into one whose reactions have exponential rates, thus admitting probabilistic analysis and simulation.

Finally, in Section 3 we look at time and clocks. We also suggest how to approximate differential equations, whose independent variable is time, by difference equations that step time by a (small) non-zero amount, and how such equations can control reaction.

1 Measured regions

Consider situations in which we need to measure physical locality. Even when a bigraphical region models a part of physical space, it may be concrete or abstract. It is concrete if it is a room, or a biological cell. It is abstract if it is an imaginary division of a physical space; for example, it may be the work space or the attention space of an individual working in a room, or it may be the range of a wireless station (as in Note C), or it may be a terrestrial region defined only by the nature of the terrain – e.g. forest, mountains, desert or pasture.[22] In either case some behaviour may depend on details of cartesian space, other behaviour only on containment or contiguity, and some on both together.

So let us see how to superpose cartesian space on a bigraph, or more exactly on a given region – a Locale. Let $\mathsf{Point} : 3$ be a binary constructor of numerals, so that Point_{axy} gives the name a to the point (x, y) in discrete 2-D cartesian space. We suppose that x and y point into the shared structure Nat_0 of natural numbers from Note D. Supposing that the Locale contains a set of neighbouring points, we draw it like this:



where we have detailed just one point. The points are arranged as a grid, to suggest that they are indeed neighbours in cartesian space. So this is just an ordinary bigraph.

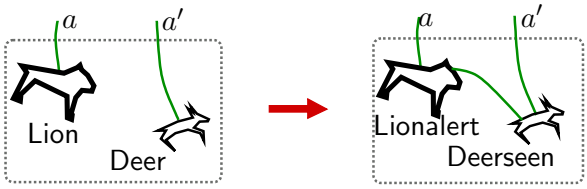
[22] The latter idea comes from the paper by Steve Benford *et al*, sl Life on the Edge: Supporting Collaboration in Location-based Experiences, CHI 2005, Portland, USA, pp721–730.

As an example, suppose we want a reaction rule for an event that requires two agents to be not only in the same locale, but close enough to each other. I modelled Benford's Savannah game (see the cited paper), in which children pretend to be lions. Lions roam the savannah, considered to be divided into locales. The lions are hunting for prey. A lion may sight a deer that is not only in the same locale but also near enough together, say within a distance of 50 metres.

So we need a reaction rule that is subject to a calculable condition on its redex. This seems to demand a refined form of reaction rule. Let us formulate what is needed in this special case, to get a better idea of what kind of refinement is needed in general.

Begin with a crude reaction rule that allows a lion to sight any deer in the same locale:

$$\mathsf{Lion}_a \,|\, \mathsf{Deer}_{a'} \longrightarrow\!\!\triangleright\ \mathsf{Lionalert}_{ab}\, \overline{{}^{b}}\, \mathsf{Deerseen}_{a'b}\ .$$



The new closed link represents the sighting. But we want to refine this so that it can only happen if the points a and a' are near. We therefore define a contextual germination rule

$$\mathsf{Point}_{axy} \,|\, \mathsf{Point}_{a'x'y'} \ \ \|\!\!-\ \ \mathsf{Near}_{aa'} \hookrightarrow \mathsf{Less}_{z,50}\, \overline{{}^{z}}\, \mathsf{Dist}_{zxyx'y'}\ .$$

together with germination rules for Dist and Less. we omit them, but they are easily defined; one calculates the distance between two points, and the other compares two arbitrary numbers. Then we refine the crude reaction rule into a conditional reaction rule, as defined at the end of Note A, by adding as the condition P the predicate expression

$$\mathsf{Point}_{axy} \,|\, \mathsf{Point}_{a'x'y'} \ \,|\, \mathsf{Near}_{aa'}$$

which must unfold to True to enable the reaction.

We could have defined two separate reaction rules, one to evaluate the predicate and the other to enact the sighting. But then there would be no guarantee that other reactions – perhaps the deer moving out of range – would not intervene between the enaction of these two rules.

Let us briefly consider abstract notions such as the attention space of a person, or the range of a radio transmitter or sensor. Such a region can naturally be modelled as a set of cartesian points, or as a node that contains such a set. Then if we adopt shared regions – i.e. dags as place graphs – as in Note C, we can expect to model the overlapping attention spaces of two people (allowing them to interact), or the overlapping ranges of sensors by the side of a motorway. We can also attempt to model the movement of a person's attention space as she moves through a crowd of people, or the movement of a car from one sensor range to another as it travels on the motorway. It is challenging to model this kind of mobility in a BRS equipped with a calculational sub-BRS.

2 Stochastic behaviour

Jane Hillston[23] pioneered the attribution of stochastic rates to the transitions of a process calculus; this led to the PEPA toolkit for performance analysis and simulation. Later Corrado Priami[24] extended this work to the π-calculus. Inspired by these advances, and by the challenge to apply bigraphs to biological behaviour, Jean Krivine, Angelo Troina and I have developed a generic stochastic treatment for bigraphs.[25]

Our approach differs in one sense. Behaviour in CCS and π-calculus is expressed directly in the form of labelled transitions, while basic behaviour of bigraphs consists of reaction rules, which are essentially *unlabelled* transitions. We therefore *define* stochastic rates only for reaction rules; then the rates of labelled transitions can by *derived*, as we shall see.

We begin by assigning to every reaction rule $\mathsf{R} = (R, R', \eta)$ a *rate* $\rho > 0$, yielding the rule

$$\mathsf{R} = (R, R', \eta, \rho)$$

where the significance of ρ is that, if applicable, the rule will be delayed by at least time t with probability $e^{-\rho t}$; a Poisson distribution. Thus rules with high rate are more likely to occur earlier. The infinite rate $\rho = \infty$ means immediate occurrence, but if two applicable rules have infinite rate then it is undetermined which will occur first (thus possibly precluding the other).

The Poisson distribution of delay is that it is 'memoryless'; that is, if a rule with rate ρ is delayed by any time, but remains applicable, then its delay thereafter will also have a Poisson distribution with rate ρ. This leads to tractable simulation and analysis using Gillespie's method.[26]

Given the rate ρ_{R} for a rule R, we derive a rate

$$rate_{\mathsf{R}}[g, g'] \stackrel{\text{def}}{=} \rho_{\mathsf{R}} \cdot count_{\mathsf{R}}[g, g']$$

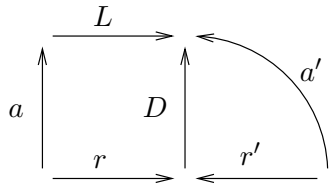
for the reaction $g \longrightarrow\!\!\triangleright\, g'$, where $count_{\mathsf{R}}[g, g']$ is the number of distinct occurrences of the redex R in g that give rise to the reaction $g \longrightarrow\!\!\triangleright\, g'$ via the rule R. This is not entirely trivial. To get the right count we first fix a concretion of g, i.e. an assignment of support elements to its nodes and edges; then we count only the occurrences of R in g whose supports are different, limiting our count to those that indeed deliver the result g'. Then, given a set $\mathcal{R}$ of rules, the rate of $g \longrightarrow\!\!\triangleright\, g'$ is defined as

$$rate_{\mathcal{R}}[g, g'] \stackrel{\text{def}}{=} \Sigma_{\mathsf{R} \in \mathcal{R}} rate_{\mathsf{R}}[g, g']$$

and, denoting this rate by ρ, we write $g \longrightarrow\!\!\triangleright_{\rho}\, g'$.

We now turn to labelled transitions. In bigraphs these take the form $a \xrightarrow{L}\!\!\triangleright\, a'$, where there is an underlying reaction rule $\mathsf{R} = (R, R', \eta, \rho_{\mathsf{R}})$ which generates a ground reaction rule $r \longrightarrow\!\!\triangleright\, r'$ satisfying the commuting diagram

[23] *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.
[24] *The stochastic π-calculus*, the Computer Journal 38(6), 1995, pp578–589.
[25] *Stochastic bigraphs*, Electronic Notes in Computer Science, 2008, pp73–96.
[26] We need assume no familiarity with this method.



where some concretion of the square is an idem-pushout (IPO). Then we derive a rate

$$rate_{\mathsf{R}}[a, L, a'] \stackrel{\text{def}}{=} \rho_{\mathsf{R}} \cdot count_{\mathsf{R}}[a, L, a']$$

where, as for a reaction $g \longrightarrow\!\!\triangleright\, g'$, we fix a concretion of $L \circ a$ and we count its distinct occurrences of the parametric redex R, limiting the count to those that indeed deliver the result $a' = D \circ r'$. Then, given a set $\mathcal{R}$ of rules, the rate of $a \longrightarrow\!\!\triangleright\, a'$ is defined as

$$rate_{\mathcal{R}}[a, L, a'] \stackrel{\text{def}}{=} \Sigma_{\mathsf{R} \in \mathcal{R}} rate_{\mathsf{R}}[a, L, a']$$

and then, denoting this rate by ρ, we write $a \stackrel{L}{\longrightarrow\!\!\triangleright}_{\rho} a'$.

This treatment of rates is reported in the paper (cited above) with Krivine and Troina. In particular the paper shows, by a simple sanity check, how the rates derived for labelled transitions are consistent with those assigned to reaction rules. Another simple check is that a transition $a \stackrel{\mathsf{id}}{\longrightarrow\!\!\triangleright}\, a'$ with identity label, which corresponds to the reaction $a \longrightarrow\!\!\triangleright\, a'$, does indeed receive the same rate as the reaction.

There appears to be a disparity with Hillston's original work in which the rates of transitions labelled (say) x may be freely varied by the user from instance to instance. For example, in CCS one can assign a rate ρ to an x-transition by writing the alternation

$$(x, \rho).P + \cdots\ ,$$

and in another alternation an x-labelled transition can be assigned a different rate ρ'. Here, by contrast, it may appear that the rate is fully determined by the underlying reaction rule.

However, variation is possible. Recall that one factor in the rate of a reaction or transition is the number of distinct redex occurrences that yield it. So, to increase the rate of the above x transition by a factor of 100, one need only include 100 copies of it in the alternation; and to save writing a long expression one can write

$$(x, 100).P + \cdots\ ,$$

where 100 is no longer a rate, but a multiplier for the rate of the underlying reaction rule. One may object that this multiplier can only be a natural number. But it is relative rates that matter, not absolute rates, so a process description using rational numbers as rating multipliers can be exactly mimicked by using natural numbers instead.

Are there interesting stochastic variants of the behavioural congruences, such as bisimilarity? First, without considering stochastics, one can limit the behaviour of a bigraphical agent a to the transitions based upon any subset $\mathcal{R}$ of the reaction rules. In this generalisation, a process is essentially a pair which we may write $a \restriction \mathcal{R}$. There is an obvious definition of bisimilarity between

such constrained processes, and it is easily shown to be a congruence – in the sense that $a\restriction\mathcal{R} \sim b\restriction\mathcal{S}$ implies $(C \circ a)\restriction\mathcal{R} \sim (C \circ b)\restriction\mathcal{S}$ for any context C.

It is then an easy step to refine this congruence by stochastics. For example, the rules $\mathcal{S}$ may have on the whole higher rates than those in $\mathcal{R}$, and this may then yield a precongruence $\prec$, where $a\restriction\mathcal{R} \prec b\restriction\mathcal{S}$ means that every $\mathcal{R}$-based transition of a is matched by a faster $\mathcal{S}$-based transition of b, and every transition in the latter class is matched by a slower one in the former. So stochastics may lead to a richer generic theory of bigraphical behaviour.

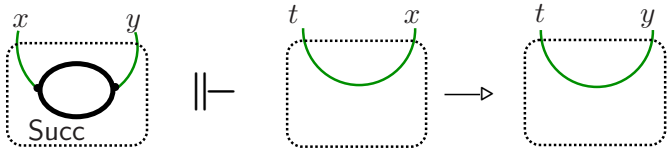
3 Measured time

Many applications will need either a global clock, or clocks local to particular regions, or both. Here, we briefly explore how stochastic rates yield a possible way to handle global time.

Let us adopt the shareable natural numbers of Section 3 of Note D. (This assumed aliasing, for convenience, but there are alternative treatments that avoid aliasing). Assume the discipline that there is a distinguished global name t for time, pointing into the unique structure Nat_0. Introduce the contextual reaction rule

$$\mathsf{Succ}_{xy} \parallel\!\!- \; t{=}x \longrightarrow\!\!\triangleright \; t{=}y$$

which we can call *ticking*. It may be drawn as follows:



This looks like the germination rule for Next in Section 3. But it is *not* a germination rule; indeed, it has no seed! It is a change of state – an advance of the clock.

We need some way to relate the use of other rules to the clock. For example, we would like ticking to trigger certain rules, ensuring that they are applied immediately, before the next clock advance. A simple way to achieve this is to use stochastic rates. Let us give ticking a finite non-zero rate, say 1. Rules that should be triggered by ticking must be given infinite rate; so they will be applied on every tick provided that they can be matched.

In this way, we can approximate ODEs (ordinary differential equations) whose independent variable is the time t. One way is to represent the dependent variables, and each of their derivatives occurring in the equations, by a family of controls indexed by the real numbers. The rules that model the ODEs will step each such control by a small increment, where it is assumed that each tick means that time is stepped by such an increment ϵ. By giving these ODE rules infinite rates, we ensure that each variable and derivative is stepped on every tick.

Certain values of the variables and their derivatives can then be used to trigger events, represented by rules whose rates may or may not be infinite. For example if such a value is a boundary condition, then it may cause a new set of ODEs to replace the old ones.

We will take this no further here. Clearly it is an important research project to evolve a standard way to represent ODEs in bigraphs, and thereby ease the representation hybrid systems. In doing this we must exploit the wide research literature that exists for hybrid systems.

Seminar Note F: Categories and motion

Robin Milner, 2009

In this seminar we begin by outlining how bigraphical theory is organised by a variety of categories, and especially by means of the functors between them. My book explains this organisation in detail, but it is helpful to summarise it here.

These categories only organise the static theory: the bigraphs without their reactions. So we go on to recall how reaction rules are added, to yield bigraphical reactive systems (BRSs). The book explains this, but stops short of asking how the static functor between two BRSs may be enriched to relate their dynamics. There is more than one answer; each answer yields a category $\mathcal{BRS}$ whose objects are BRSs, and whose arrows are dynamically-enriched functors between them.

We end by posing a question for research: How do we formulate the modularity of BRSs, explaining how to assemble a BRS from sub-BRSs whose signatures may be partly shared? Does the categorical structure help here?

1 S-categories

Let us recall some of the basic categorical notions of bigraphs. These details need not be fully understood for what follows, but they justify some of the design decisions taken for bigraphs.

First, a *symmetric partial monoidal* (*spm*) category is a standard kind of category except for one thing: it is only *partially* monoidal. This partiality derives from the choice to draw the names in a bigraph from an infinite alphabet $\mathcal{X}$, rather than to represent them more abstractly by finite ordinals. The choice is a pragmatic one; it enables a user-friendly treatment of linkage in bigraphs, including a direct derivation of the parallel composition combinators that are familiar from process calculi.

In turn, an *s-category* is just an spm category, except for one thing: given arrows $F : I \to J$ and $G : J \to K$, the composition $G \circ F : I \to K$ exists only if F and G have disjoint *support sets*; this disjointness is also required for a monoidal product $F \otimes G$ to be defined. A support set is a finite set associated with each arrow $F : I \to J$. A familiar example of support is in ordinary graph theory, distinguishing a concrete from an abstract graph; there, a concrete graph has the identities $\{v_1, \ldots, v_n\}$ of nodes as its support, and to combine two graphs we typically require their node-sets to be disjoint. An abstract graph is an equivalence class of bigraphs that differ only by a bijection of supports.

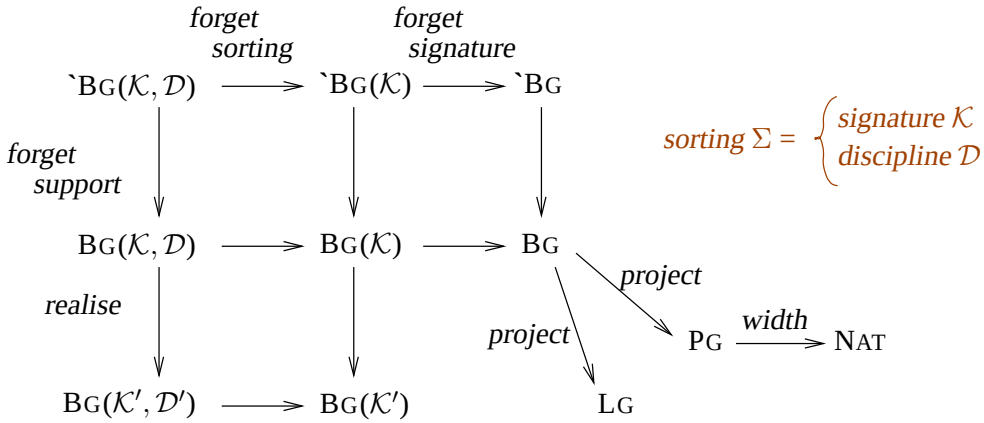
Similarly a concrete bigraph has support consisting of its nodes and edges, while an abstract bigraph is a support-equivalence class of concrete ones. Thus concrete bigraphs form an s-category, while abstract ones – being support-equivalence classes of concrete ones – form an spm category, which is a just an s-category whose support sets are empty. Intuitively, support plays for bigraphs the role that *labelling* plays in the λ-calculus; it is useful for distinguishing or counting the *occurrences* of an entity.

2 Wide s-categories of bigraphs

Each kind of bigraphs forms an s-category; some kinds have empty support, so are spm categories. This section is concerned only with how s-categories organise the static structure of bigraphs, not their dynamics. At the end of the section we see what a 'wide' s-category is, and the central role that it plays.

The kinds can be arranged in a diagram showing the functors between them; we shall explain the role played by each functor.

WIDE S-CATEGORIES:



Let us begin with $\mathrm{BG}(\mathcal{K})$, in the middle of the diagram. This is the spm category of abstract bigraphs over a signature, which is a set of controls, each with its arity. Each node is assigned a control K, whose arity determines the number and order of its ports. There is an obvious forgetful functor that forgets the signature; its target BG is just bigraphs whose nodes lack controls. Much theory is done first in BG; it is then less cluttered, and is preserved by retrofitting the controls.

Since a bigraph is a combination of a place graph and a link graph, there are two obvious functors projecting each bigraph onto these two constituents. This allows some theory of place graphs and link graphs to be done separately, then combined. Later we shall discuss the width functor to NAT.

Moving to the left, a *sorting* Σ consists of a signature $\mathcal{K}$ together with a *discipline* $\mathcal{D}$ that constrains the bigraphs over $\mathcal{K}$ that are admissible. This enriched and constrained category is denoted by $\mathrm{BG}(\Sigma)$, where $\Sigma = (\mathcal{K}, \mathcal{D})$. Hitherto, almost every application of bigraphs has involved a sorting. For example, in the simple case of CCS, the sorting Σ_{ccs} requires that the nesting of nodes should interleave send- and get-nodes with alt-nodes. In a built environment, say a building, one would naturally require no room to contain another room or a building.

It is not fully settled what the general definition of a sorting $\Sigma = (\mathcal{K}, \mathcal{D})$ should be, beyond requiring that there be a forgetful functor from $\mathrm{BG}(\Sigma)$ to $\mathrm{BG}(\mathcal{K})$ for some $\mathcal{K}$. But Birkedal, Debois and Hildebrandt[27] have proposed that it consist of any functor that is both surjective on objects and faithful (i.e. injective on each homset). This definition has the merit that it allows *binding bigraphs*

[27] *Sortings for reactive systems*, in Proc. CONCUR'06, Lecture Notes in Computer Science 4137, 2006, pp248–262.

(see Note B) to be expressed as a sorting. The spirit of the definition is that, while it determines only the target of this functor to be a bigraphical category, the source category is less determined – it may be any wide spm category.

Consider now the bottom row of the diagram. Bigraphs may be used to model a system at different levels of detail. This recalls homomorphisms of algebras, in which an algebra $\mathcal{A}$ is refined to an algebra $\mathcal{A}'$ by realising each single operator of $\mathcal{A}$ by a compound operation built from the operators of $\mathcal{A}'$. Here, then, we would represent each K-ion of $\mathrm{BG}(\Sigma)$ by a compound bigraph in $\mathrm{BG}(\Sigma')$.[28]

We now come to the top row. Hitherto we have dealt with abstract bigraphs, wehere we choose not to identify different 'occurrences' of the same control K. However, there are situations in which we may wish to do so. The first such situation was in deriving labelled transitions for an agent. This derivation depends critically on identifying nodes of a redex R that occur in an agent a, so that the derived label L of a transition $a \xrightarrow{L}\!\!\triangleright\, a'$ may be constructed from the other nodes of R. Another need for node identity arises when we wish to ask "how *many* ways does a redex R occur in a given bigraph?". This has been crucial in defining the stochastic rate of a reaction, as explained in Note C.

There is one final feature of our diagram to be explained: the role of NAT, the category of finite ordinals and maps between them. Recall that each finite ordinal m is the set of its predecessors, i.e. $m = \{0, 1, \ldots, m-1\}$; also recall that the objects in PG are finite ordinals. The functor $\mathsf{width} : \mathrm{PG} \to \mathrm{NAT}$ is defined as the identity on objects, while for any place graph $P : m \to n$ we take $\mathsf{width}(P)$ to be the map $\mathsf{root} : m \to n$, where $\mathsf{root}(i) \in n$ is the unique root (region) of P that contains the site $i \in m$.

Such a functor, then, is possessed by all bigraphical categories in our diagram; this is why we call them *wide* s-categories. Indeed an s-category is *defined* to be wide just when it is equipped with a functor to NAT. This functor is vital to the enrichment of s-categories with reaction rules, as we see in the next section.

3 Wide reactive systems (WRSs)

We now embark on an overview of the dynamics of bigraphs. It has turned out that much of bigraphical theory, static and dynamic, is best done at the more general level of s-categories. Having outlined how some of the static theory can be organised in this way, we now look at organising the dynamic theory.

In Definition 7.1 of the book, a *reactive system* (RS) is defined as an s-category equipped with reaction rules of the form (r, r'), *redex* and *reactum*.[29] Such a set of rules generates a reaction relation $\longrightarrow\!\!\triangleright$ between agents; $a \longrightarrow\!\!\triangleright\, a'$ means that some occurrence of a redex r in a is replaced by the corresponding reactum r' to yield a'.

Then in Definition 7.2 a *wide* reactive system (WRS) is defined as an RS equipped with a so-called *width* functor to NAT, and a way to use this functor to determine *where* within an agent a a

[28] Ions typically have rank 1, i.e. a single site, but a sorting can give them any rank.

[29] Some detail is suppressed here, especially how these rules are closed under support equivalence.

reaction rule is applied to yield a reaction $a \longrightarrow\!\triangleright\, a'$. The first importance of this has been to ensure that, at the general level of WRSs, bisimilarity and other behavioural equivalences are congruential. The short proof of this theorem[30] is one of the deeper results in bigraphical theory.

Here we focus upon a different aspect of dynamics: How may we enrich the static functor between two WRSs into a morphism that imposes a condition requiring their reaction relations to be compatible? This would yield a category of WRSs whose arrows are these morphisms.

Let $(\mathcal{A}, \mathcal{R})$ be a WRS having consisting of a wide s-category $\mathcal{A}$ equipped with reaction rules $\mathcal{R}$). Let $(\mathcal{B}, \mathcal{S})$ be another such, with a functor $\mathcal{F} : \mathcal{A} \to \mathcal{B}$. Each WRS then has a reaction relation $\longrightarrow\!\triangleright$ over its agents. A natural compatibility condition, to enrich $\mathcal{F}$ into a morphism of WRSs, is to require

$$\text{CONDITION 1 :} \qquad a \longrightarrow\!\triangleright\, a' \text{ implies } \mathcal{F}a \longrightarrow\!\triangleright\, \mathcal{F}a' \ .$$

This will clearly be satisfied if $\mathcal{S} = \{(\mathcal{F}r, \mathcal{F}r') \mid (r, r') \in \mathcal{R}\}$. Condition 1 is preserved by composition of functors, so it could be taken as the definition of morphisms in a category of WRSs. But it is too weak to be of much use. For example we might adopt it for the functors that forget sorting discipline or signature, or for the two projection functors; but then the converse of Condition 1 is far from holding, so we are not likely to gain insight into the dynamics of the source WRS in each case.

We could indeed strengthen our condition to require the converse:

$$\text{CONDITION 2 :} \quad \begin{cases} a \longrightarrow\!\triangleright\, a' \text{ implies } \mathcal{F}a \longrightarrow\!\triangleright\, \mathcal{F}a' \ ; \\ \mathcal{F}a \longrightarrow\!\triangleright\, b' \text{ implies that, for some } a',\ a \longrightarrow\!\triangleright\, a' \text{ and } b' = \mathcal{F}a' \ . \end{cases}$$

Interestingly enough, this is satisfied by the forgetful functor $\mathcal{U} : \grave{\mathrm{B}}\mathrm{G}(\Sigma) \to \mathrm{B}\mathrm{G}(\Sigma)$, provided that the rules $\mathcal{R}'$ in the concrete BRS $\grave{\mathrm{B}}\mathrm{G}(\Sigma, \mathcal{R}')$ are taken to be all concretions of the rules $\mathcal{R}$ in the abstract BRS $\mathrm{B}\mathrm{G}(\Sigma, \mathcal{R})$. This plays an important role in the proof that bisimilarity is a congruence for abstract BRSs.

Now consider realisations of one BRS by another. Let $\mathcal{A} = \mathrm{B}\mathrm{G}(\Sigma_a, \mathcal{R})$ and $\mathcal{B} = \mathrm{B}\mathrm{G}(\Sigma_b, \mathcal{S})$ be BRSs for which there is a realisation functor $\mathcal{F} : \mathrm{B}\mathrm{G}(\Sigma_{\mathcal{A}}) \to \mathrm{B}\mathrm{G}(\Sigma_{\mathcal{B}})$. What condition would we like to impose in the two reaction relations, for the realisation to be useful in practice? We suppose that it takes many reactions in $\mathcal{B}$ to realise a single reaction in $\mathcal{A}$; but for any agent a in $\mathcal{A}$ we also want every reaction of $\mathcal{F}a$ in $\mathcal{B}$ to be a step towards realising a reaction of a in $\mathcal{A}$. We may therefore adopt

$$\text{CONDITION 3 :} \quad \begin{cases} a \longrightarrow\!\triangleright\, a' \text{ implies } \mathcal{F}a \longrightarrow\!\triangleright^{*} \mathcal{F}a' \ ; \\ \mathcal{F}a \longrightarrow\!\triangleright\, b' \text{ implies that, for some } a',\ a \longrightarrow\!\triangleright\, a' \text{ and } b' \longrightarrow\!\triangleright^{*} \mathcal{F}a' \ . \end{cases}$$

It is easy to check (and well-known in other contexts) that this condition is preserved by the composition of morphisms.

Since condition 2 is stronger than condition 3, we may now consider the category $\mathcal{WRS}$ of both concrete and abstract WRSs, with realising morphisms satisfying condition 3. As a sub-category, it has a candidate for our proposed category $\mathcal{BRS}$

[30] worked out with Jamey Leifer

But is Condition 3 strong enough? Intuitively, it says that any first reaction of the agent $\mathcal{F}a$ in $\mathcal{B}$ *can* be extended to a sequence of reactions that realises some single reaction $a \longrightarrow\!\!\triangleright\, a'$ in $\mathcal{A}$. But it still allows that subsequent reactions in $\mathcal{B}$, after the first, *can diverge* from that sequence, preventing its completion.

We leave open the problem of strengthening Condition 3. A similar problem may have been solved in another context, and indeed there appears to be more than one solution. A solution may already exist in TeReSe, the definitive text on rewriting systems.[31] A good solution will be important for modelling complex systems at different levels of detail, each level realised by the one below.

4 Modularity of BRSs

Realisations are one way to relate BRSs to each other; assembling BRSs from sub-BRSs is a very different concern. In Notes A and D we have discussed embedding calculation (itself a BRS) in another BRS, and research at ITU Denmark has explored other forms of assembly. I do not yet have more to say on this, but I regard it as an essential research topic if we are going to modularise large applications – and a large application will be unintelligible unless it is indeed assembled from modules, in a way that allows analysis of the whole to be built on analysis of the parts.

So it is to be hoped that some forms of assembly can be modelled within the category $\mathcal{BRS}$ that we have discussed here.

[31] TeReSe (JW Klop *et al.*), *Term Rewriting Systems*, Cambridge Tracts in Theoretical Computer Science 55, Cambridge University Press, 2003.