Variable binding, symmetric monoidal closed theories, and bigraphs

Richard Garner, Tom Hirschowitz, Aurélien Pardon

To cite this version:

Richard Garner, Tom Hirschowitz, Aurélien Pardon. Variable binding, symmetric monoidal closed theories, and bigraphs. Bravetti, Mario; Zavattaro, Gianluigi. CONCUR 2009, 2009, Bologna, Italy. Springer, 5710, pp.321-337, 2009, Lecture Notes in Computer Science. <10.1007/978-3-642-04081-8_22>. <hal-00388100>

HAL Id: hal-00388100

https://hal.archives-ouvertes.fr/hal-00388100

Submitted on 26 May 2009

Variable binding, symmetric monoidal closed theories, and bigraphs

Richard Garner[1], Tom Hirschowitz[2], and Aurélien Pardon[3]

[1] Cambridge University
[2] CNRS, Université de Savoie
[3] ENS Lyon

Abstract. This paper investigates the use of symmetric monoidal closed (SMC) structure for representing syntax with variable binding, in particular for languages with linear aspects. In this setting, one first specifies an SMC *theory* $\mathcal{T}$, which may express binding operations, in a way reminiscent from higher-order abstract syntax (HOAS). This theory generates an SMC category $S(\mathcal{T})$ whose morphisms are, in a sense, terms in the desired syntax. We apply our approach to Jensen and Milner's (abstract binding) bigraphs, in which *processes* behave linearly, but *names* do not. This leads to an alternative category of bigraphs, which we compare to the original.

1 Introduction

How to rigorously handle variable binding? The recent amount of research on this issue attests its delicacy [10, 9, 14]. A main difficulty is perhaps to reconcile α-conversion with initial algebra semantics: α-conversion equates terms up to renaming of bound variables; initial algebra semantics requires that terms form the free, or initial model specified by a given signature.

We here investigate an approach sketched by Coccia et al. [4], based on SMC theories, which they called GS·Λ theories. In this setting, one first specifies an SMC *theory* $\mathcal{T}$, which may express binding operations, in a way reminiscent from higher-order abstract syntax [23] (HOAS). This theory freely generates an SMC category $S(\mathcal{T})$ whose morphisms are, in a sense, terms in the desired syntax.

The known presentations of $S(\mathcal{T})$ mainly fall into two classes: syntactic or graphical. Our emphasis in this paper is on a graphical presentation of $S(\mathcal{T})$ and example applications.

We start in Section 2 with an expository account of SMC theories and our construction of $S(\mathcal{T})$. This yields a monadic adjunction. The morphisms of $S(\mathcal{T})$ are a variant of proof nets, in the sense of intuitionistic multiplicative linear logic [12] (IMLL): they are equivalence classes of special graphs called *linkings*, which must satisfy a certain *correctness* condition. Linkings compose by "glueing" the graphs together, and correctness is preserved by composition.

We continue with a few examples in Section 3, to demonstrate the use of $S(\mathcal{T})$ as a representation for syntax with variable binding. In our presentation of $S(\mathcal{T})$, terms look like the usual abstract syntax trees, and are actually a generalisation of previous graphical forms of λ-calculus, e.g., Wadsworth's λ-graphs [28]. The objects of $S(\mathcal{T})$ are IMLL formulae. The subcategory of rank 0 formulae (without $\multimap$) roughly corresponds to terms, and composition models linear substitution. Rank 1 formulae express a kind of term with holes, or *context*, and composition models a kind of constrained context application, or substitution with capture. Formulae of higher ranks yield a form of higher-order contexts. This kind of substitution would show up with any closed structure, e.g., cartesian closed categories, but is not directly available in more traditional approaches [9, 14, 10]. Conversely, non-linear capture-avoiding substitution requires a bit more work in our setting (and does not appear in this paper). Along the way, we prove a decomposition result showing the flexibility of our approach, and we observe that the use of SMC structure facilitates the cohabitation of linear and non-linear aspects in a common language.

To further support this latter claim, Section 4 studies Jensen and Milner's bigraphs [17] in our setting, in which *processes* behave linearly, but *names* do not. We translate each bigraphical signature $\mathcal{K}$ into an SMC theory $\mathcal{T}_{\mathcal{K}}$, and show that bigraphs over $\mathcal{K}$ essentially embed into $S(\mathcal{T}_{\mathcal{K}})$, the free SMC category generated by $\mathcal{T}_{\mathcal{K}}$. Furthermore, although $S(\mathcal{T}_{\mathcal{K}})$ is much richer than the original, the embedding is surjective on whole programs.

2 Symmetric monoidal closed theories

In this section, we provide an overview of the construction of $S(\mathcal{T})$. A more technical presentation may be found in our work [11], which itself owes much to Trimble [27] and Hughes [16].

2.1 Signatures

Roughly, an SMC category is a category with a tensor product $\otimes$ on objects and morphisms, symmetric in the sense that $A \otimes B$ and $B \otimes A$ are isomorphic, and such that $(- \otimes A)$ has a right adjoint $(A \multimap -)$, for each object A. We do not give further details, since we are interested in describing the free such category, which happens to be easier. Knowing that there is a category SMCCat of SMC categories and strictly structure-preserving functors should be enough to grasp the following.

An SMC *signature* Σ consists of a set X of *sorts*, equipped with a (directed, multi-) graph whose vertices are IMLL formulae over X, as defined by:

$$A, B, \ldots \in \mathcal{F}(X) ::= x \mid I \mid A \otimes B \mid A \multimap B \qquad\qquad \text{(where } x \in X\text{)}.$$

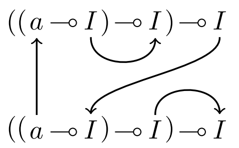
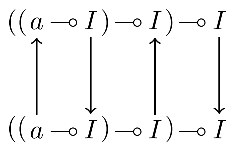
We think of each edge $A \to B$ of the graph as specifying an operation of type $A \to B$. A morphism of signatures $(X, \Sigma) \longrightarrow (Y, \Sigma')$ is a function $X \xrightarrow{f} Y$, equipped with a morphism of graphs, whose vertex component is "$\mathcal{F}(f)$", i.e., the function sending any formula $A(x_1, \ldots, x_n)$ to $A(f(x_1), \ldots, f(x_n))$. This defines a category SMCSig of signatures.

There is then a forgetful functor $\mathsf{SMCCat} \xrightarrow{U} \mathsf{SMCSig}$ sending each SMC category $\mathcal{C}$ to the graph with as vertices formulae in $\mathcal{F}(\mathrm{ob}(\mathcal{C}))$, and as edges $A \longrightarrow B$ the morphisms $[\![A]\!] \longrightarrow [\![B]\!]$ in $\mathcal{C}$, where $[\![A]\!]$ is defined inductively to send each syntactic connective to the corresponding function on $\mathrm{ob}(\mathcal{C})$.

We will now construct an SMC category $S(\Sigma)$ from any signature Σ, and extend this to a functor $\mathsf{SMCSig} \xrightarrow{S} \mathsf{SMCCat}$, left adjoint to U. How does $S(\Sigma)$ look like? Under the Curry-Howard-Lambek correspondence, an SMC signature amounts to a set of IMLL axioms, and the free SMC category $S(\Sigma)$ over a signature Σ has as morphisms IMLL proofs under the corresponding axioms, modulo cut elimination. Or, equivalently, morphisms are a variant of proof nets, which we introduce gradually in the next sections.

2.2 The free symmetric monoidal closed category over a set

In the absence of axioms, i.e., given only a set of sorts, or propositional variables, say X, Hughes [16] has devised a simple presentation of $S(X)$. Consider for a guiding example the two endomorphisms of $((a \multimap I) \multimap I) \multimap I$:


$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$

$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$

(the right-hand one being the identity).

First, the *ports* of a formula, i.e., occurrences of sorts or of I, are given polarities: a port is *positive* when it lies to the left of an even number of $\multimap$'s in the abstract syntax tree, and *negative* otherwise[4]. For example, in the above formula, a and the middle I are negative, the other occurrences of I being positive. When constructing morphisms $A \longrightarrow B$, the ports in A and B will be assigned a *global* polarity, or a polarity *in* the morphism: the ports of B have their polarity in B, while those of A have the opposite polarity. For example, in the above examples, the occurrence of a in the domain is globally positive.

A *linking* is a partial function f from negative ports to positive ports, such that for each sort a, f maps negative a ports to positive a ports, bijectively. We observe that this allows to connect I ports to ports of any type. This last bit does not appear in the above example; it does in (1) below. Clearly from the example, linkings are kind of graphs, and we call their edges *wires*.

A linking is then *correct* when it is a total function, and when it moreover satisfies the Danos-Regnier (DR) criterion [6]. The latter roughly goes as follows. An IMLL formula may be written using the connectives of *classical* linear logic, defined by the grammar:

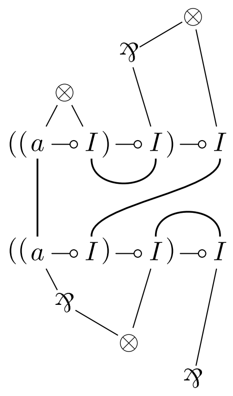
$$\begin{array}{rlcll} A, B, \ldots ::= & x & \mid & I & \mid & A \otimes B \\ \mid & x^{\perp} & \mid & \perp & \mid & A \parr B. \end{array}$$

[4] The sign of a port in A is directly apparent viewing A is a classical LL formula, see the next paragraph.

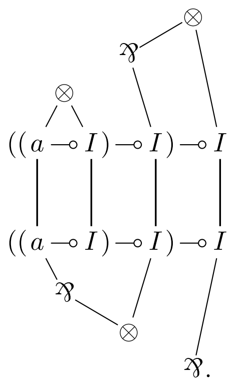
The de Morgan dual $A^\perp$ of A is defined as usual (by swapping connectives, vertically in the above grammar). We have removed $A \multimap B$, now encoded as $A^\perp \parr B$; some classical formulae are not expressible in IMLL, such as $\perp$, or $x \parr x$. The classical formulation of our above example is $((a^\perp \parr I) \otimes \perp) \parr I$.

Then, a *switching* of a classical formula is its abstract syntax tree, where exactly one argument edge of each $\parr$ has been removed. A *switching* of a linking $A \xrightarrow{f} B$ is a graph obtained by glueing (in the sense of pushouts in the category of undirected graphs) along ports the (undirected) wires of f with switchings of $A^\perp$ and B. The linking then satisfies DR iff all its switchings are acyclic and connected.

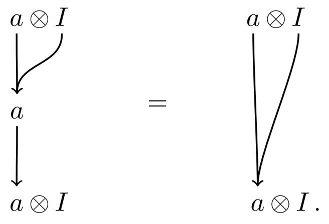
On our above examples, a sample switching yields


$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$
$\otimes$
$\parr$

and


$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$
$\otimes$
$\parr$

Linkings compose by glueing along ports in the middle formula, and correctness is preserved under composition, which yields a category $S_0(X)$. For example, composing the structural isomorphism ρ with its obvious candidate inverse yields:



(1)

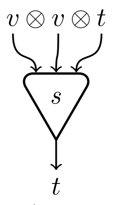
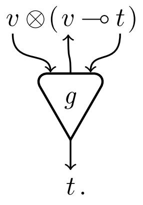
This does not yield an identity. And indeed, correct linkings do not form an SMC category. Instead, they form the free *split* SMC category over X [16]. A split SMC category is like an SMC category, where λ and ρ are only required to have left inverses, as exemplified with ρ in (1).

Here is the final step: let a *rewiring* of some correct linking f be any linking obtained by changing the target of exactly one wire from some occurrence of I in f, without breaking correctness. Typically, (1) rewires to the identity. Then $S(X)$ is the result of quotienting $S_0(X)$ by the equivalence relation generated by rewiring.

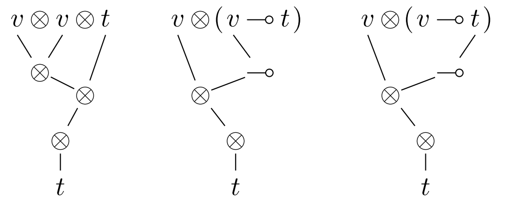
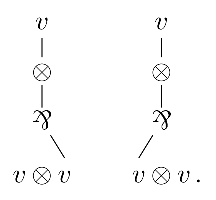
2.3 The free symmetric monoidal closed category over a signature

We now extend the construction to SMC signatures Σ: we have a set of sorts X, plus a set of operations. We now enrich linkings with, for each operation $A \xrightarrow{c} B$, a formal morphism. We picture these by *cells*, in the spirit of *interaction nets* [18]. For example, consider the π-calculus: we will see in Section 3 that the corresponding signature has one sort v for names and one sort t for processes, and among others two operations send and get of types:

$$(v \otimes v \otimes t) \xrightarrow{s} t \qquad\qquad (v \otimes (v \multimap t)) \xrightarrow{g} t.$$

This yields cells:  and 

We then extend linkings $A \to B$ to include cells in a suitable way — a glance at (3) might help. We consider linking equivalent modulo the choice of support, i.e., the choice of cells. Linkings compose as before. The question is then: what is a switching in the extended setting? The answer is that taking a switching of a cell $A \xrightarrow{c} B$ is replacing the cell with a switching of $A \otimes B^{\perp}$. For example, consider the send and get operations, and a *contraction* operation $v \xrightarrow{c} v \otimes v$. Their respective switchings are:

 and 

To understand why this is right, observe that SMC categories have a *functional completeness* property, in the sense of Lambek and Scott [19]. Roughly, this means that any morphism $C \to D$ using a cell $A \xrightarrow{c} B$ may be parameterised over it, i.e., be decomposed as

$$C \xrightarrow{\cong} I \otimes C \xrightarrow{\ulcorner c \urcorner \otimes C} (A \multimap B) \otimes C \xrightarrow{f} D, \tag{2}$$

where $\ulcorner c \urcorner$ is the currying of c. Figure 1 pictures (2) graphically. (Thick wires denote several atomic wires in parallel.) This rightly suggests that an operation $A \xrightarrow{c} B$ should have the same switchings as $A \multimap B$ in the domain, i.e., $A \otimes B^{\perp}$.

Theorem 1. *This yields a monadic adjunction*

$$\begin{tikzcd} \mathsf{SMCSig} \arrow[r, bend left, "S"] \arrow[r, phantom, "\perp"] & \mathsf{SMCCat}. \arrow[l, bend left, "U"] \end{tikzcd}$$

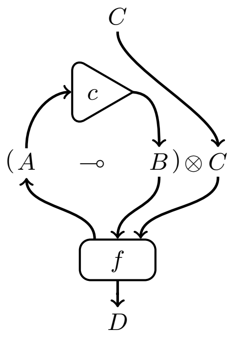


Fig. 1. Functional completeness

2.4 The free symmetric monoidal closed category over a theory

That gives the construction for signatures. We now extend it to SMC theories: define a theory $\mathcal{T}$ to be given by a signature Σ, together with a set $E_{A,B}$ of equations between morphisms in $S(\Sigma)(A, B)$, for all A, B. The free SMC category $S(\mathcal{T})$ generated by such a theory is then the quotient of $S(\Sigma)$ by the equations. Constructing $S(\mathcal{T})$ graphically is more direct than could have been feared: we first define the binary predicate $f_1 \sim f_2$ relating two morphisms $C \overset{f_1,f_2}{\rightrightarrows} D$ in $S(\Sigma)$ as soon as each f_i decomposes (remember (2) and Figure 1) as

$$C \xrightarrow{\cong} I \otimes C \xrightarrow{\ulcorner g_i \urcorner \otimes C} (A \multimap B) \otimes C \xrightarrow{f} D$$

with a common f, with $(g_1, g_2) \in E_{A,B}$, and where $\ulcorner g \urcorner$ is the currying of g. Then, we take the smallest generated equivalence relation, prove it stable under composition, and quotient $S(\Sigma)$ accordingly.

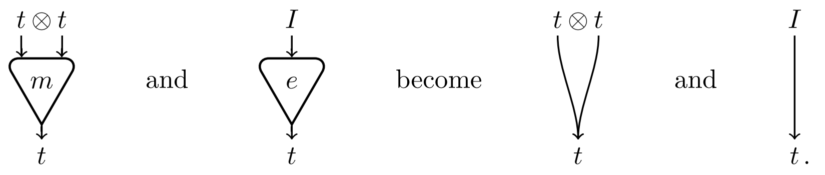
Finally, $S(\mathcal{T})$ is initial in the following sense. Let the category of *representations* of $\mathcal{T}$ be the full subcategory of the comma category $\Sigma{\downarrow}U$ whose objects are the morphisms $\Sigma \longrightarrow U(\mathcal{C})$, for which $\mathcal{C}$ is an SMC category satisfying the equations in E. Now consider the morphism $\Sigma \xrightarrow{\eta} US(\Sigma) \xrightarrow{q} US(\mathcal{T})$, where q is the quotient by the equations in E.

Theorem 2. *This morphism is initial in the category of representations of $\mathcal{T}$.*

2.5 Commutative monoid objects

We finally slightly tune the above construction to better handle the special case of commutative monoids. In a given theory $\mathcal{T} = (\Sigma, E)$, assume that a sort t is equipped with two operations $t \otimes t \xrightarrow{m} t$ and $I \xrightarrow{e} t$, with equations making it into a commutative monoid (m is associative and commutative, e is its unit). Further assume that m and e do not occur in other equations. In this case, we sketch (for lack of space) an alternative, more economic description of morphisms in $S(\mathcal{T})$.

Start from the original definition, relax the bijection condition on linkings, i.e., allow them to map negative a ports to positive a ports non-bijectively for any a, and then replace m and e as follows:



For a commutative comonoid (c, w), the dual trick does not quite work, because of problems with weakening. But still, a non-empty tree of c's may be represented by several arrows leaving its root. Observe that while m has as only switching the complete graph, c has two switchings (the formula is $v \otimes (v^{\perp} \parr v^{\perp})$).

2.6 Modularity

Melliès [20] convincingly explains the need for *modular* models of programming languages and calculi. In a slightly different sense, we argue that SMC categories provide a modular model of syntax. Namely, we obtain, for any theory $\mathcal{T}$:

Proposition 1. *For any proof net $A \xrightarrow{f} B$ in $S(\mathcal{T})$ with a set C of cells, and any partition of C into C_1 and C_2, f decomposes as*

$$A \xrightarrow{f_1} D \xrightarrow{f_2} B,$$

where each f_i contains exactly the cells in C_i.

The proof is by inductively applying the decomposition in Figure 1. The proposition intuitively says that, thinking of operations in Σ as atomic building blocks, each term may be obtained by plugging such blocks together by composition. An example is in Section 3.2.

3 First examples

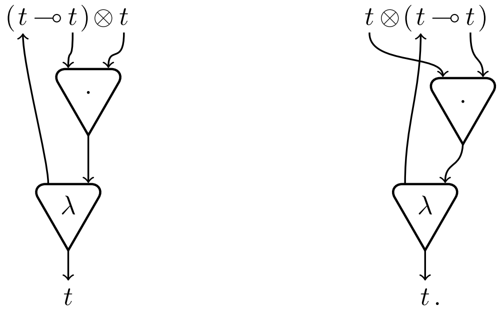
In this section, we explain how to build the λ-calculus in stages, starting from the linear λ-calculus, and passing through a kind of λ-calculus with sharing of terms. We then give an example application of Proposition 1. We end with a π-calculus example, which we will use as our main example in Section 4.

3.1 Lambda-calculus

We start with the easiest application: the untyped λ-calculus. If we naively mimick HOAS to guess a signature for the λ-calculus, we obtain one sort t and operations $t \otimes t \xrightarrow{\cdot} t$ and $(t \multimap t) \xrightarrow{\lambda} t$. However, the free SMC category on this signature is the *linear* λ-calculus, as shown by the following standard result:

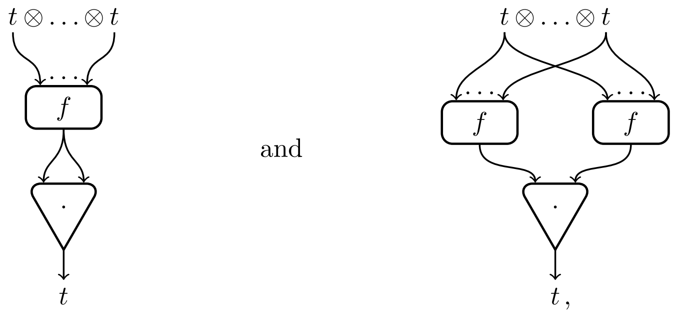
Proposition 2. *Morphisms $I \longrightarrow t$ are in bijection with closed linear λ-terms.*

Composition in our category is like *context* application in λ-calculus. A context is a term with (possibly several, numbered) holes, and context application is replacement of the hole with a term (or another context), possibly capturing some variables. The correspondence is tedious to formalise though, because contexts do not have enough information. For example, consider the context $\lambda x.(\Box_0 \cdot \Box_1)$ with two holes $\Box_0$ and $\Box_1$. Exactly one of $\Box_0$ and $\Box_1$ may use x, but this information is not contained in the context, which makes context application partial. In our setting, each possibility corresponds to one of the following morphisms:



(3)

A first attempt to recover the full λ-calculus is to add a contraction and a weakening $t \xrightarrow{c} t \otimes t$ and $t \xrightarrow{w} I$ to our signature, with the equations making (c, w) into a commutative comonoid. The free SMC category on this theory is close to Wadsworth's λ-graphs [28], which are a kind of λ-terms with a fine representation of sharing. For example, it contains two morphisms



which, because contraction is not natural, are different.

To recover the standard λ-calculus without sharing, a solution is to consider two sorts: a sort t for terms, and a sort v for variables, an idea that has been explored in the context of HOAS [8]. The theory then contains:

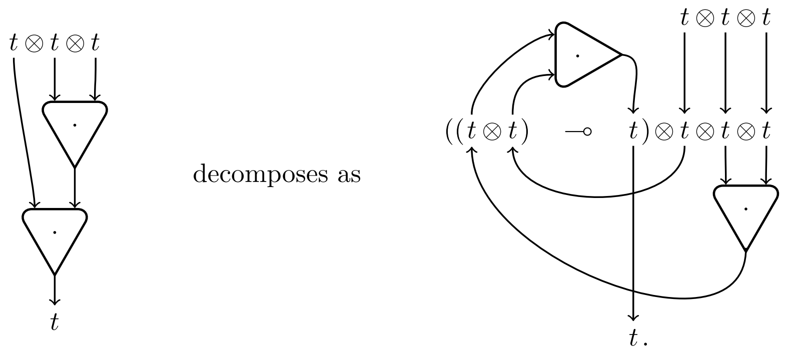
$$t \otimes t \xrightarrow{\cdot} t \qquad (v \multimap t) \xrightarrow{\lambda} t \qquad v \xrightarrow{c} v \otimes v \qquad v \xrightarrow{w} I \qquad v \xrightarrow{d} t,$$

where the latter is instantiation of a variable as a term, and (c, w) is a commutative comonoid. We obtain:

Proposition 3. *Morphisms $I \longrightarrow t$ are in bijection with closed λ-terms. Among them, those not using c nor w are in bijection with closed linear λ-terms.*

3.2 Higher order and modularity

We now give an example decomposition as in Proposition 1. Consider the context with numbered holes $(\square_0 \cdot \square_1) \cdot \square_3$, and consider the decomposition of Proposition 1 with C_1 containing exactly the outermost application. Pictorially,



Observe that this makes use of a higher-order formula, namely $(t \otimes t) \multimap t$. Also observe in advance that Jensen and Milner's category of bigraphs $M(\mathcal{K})$ does not feature such a decomposition.

3.3 Pi-calculus example

A reasonable theory $\mathcal{T}$ for the π-calculus could have at least the operations s and g specified above, plus commutative comonoid structure (c, w) on v, plus commutative monoid structure $(|, \mathbf{0})$ on t. Consider furthermore a name restriction operation $I \xrightarrow{\nu} v$, with the equation $w \circ \nu = \mathrm{id}_I$. We do not claim that this theory $\mathcal{T}$ is the right one for the π-calculus, but it is relevant for bigraphs. (An alternative type for ν is $(v \multimap t) \longrightarrow t$.)

Consider the π-calculus term with ordered holes

$$(a(x).(\square_0 \mid \bar{x}\langle x \rangle)) \mid \nu b.(\bar{a}\langle b \rangle.\square_1).$$

This term may have many different interpretations as a morphism in $S(\mathcal{T})$. A first possibility is depicted in Figure 2. Recall that several arrows leaving a port mean a tree of contractions (the port has to have type v), while several arrows entering a port mean a tree of parallel compositions (the port has type t). Finally, a positive t port with no entering arrow means a 0.

The free variable a of the term is represented by the occurrence of v in the codomain. It is used three times: twice following the term, and once more for transmitting it to $\square_0$ and $\square_1$.

But the language of SMC categories allows additional flexibility w.r.t. syntax. For example, we could choose to impose that $\square_0$ and $\square_1$ may not use a. That would mean changing the domain for $(v \multimap t) \otimes (v \multimap t)$, and removing the leftmost wire. Or, we could, e.g., only allow $\square_0$ to use a, and not $\square_1$. That would only mean change the domain to $((v \otimes v) \multimap t) \otimes (v \multimap t)$ (the leaves do not change, so the wires may remain the same).

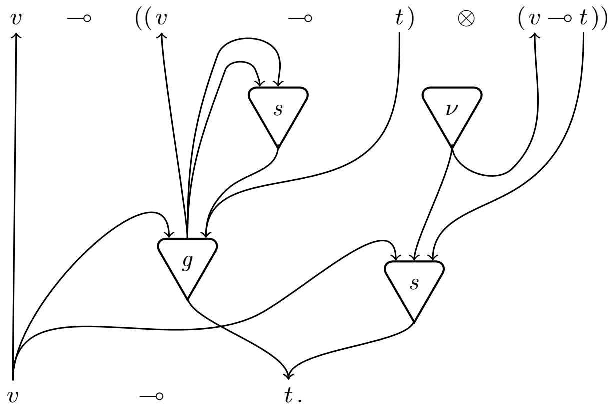


Fig. 2. A π-calculus example

4 Binding bigraphs

In this section, we consider Jensen and Milner's [17] (abstract binding) bigraphs. They are a general framework for reasoning about distributed and concurrent programming languages, designed to encompass both the π-calculus [22] and the Ambient calculus [3]. We are here only concerned with bigraphical syntax: given what we call a *bigraphical signature* $\mathcal{K}$, Milner constructs a *pre-category*, and then a category $M(\mathcal{K})$, whose objects are *bigraphical interfaces*, and whose morphisms are bigraphs.

Its main features are (1) the presence of *relative pushouts* (RPOs) in the pre-category, which makes it well-behaved w.r.t. bisimulations, and that (2) in both the pre-category and the category, the so-called *structural* equations become equalities. Examples of the latter are, e.g., in π and Ambients, renaming of bound variables, associativity and commutativity of parallel composition, or scope extrusion for restricted names. Also, bigraphs follow a scoping discipline ensuring that, roughly, bound variables are only used below their binder.

We now proceed to recall what bigraphs are, and sketch our interpretation in terms of SMC theories.

4.1 Bigraphs

We work with a slightly twisted definition of bigraphs, in two respects. First, we restrict Jensen and Milner's *scope* rule by adding a *binding* rule to be respected by bigraphs. This rule rectifies a deficiency of the scope rule, which prevented bigraphs to be stable under composition in the original paper [17]. It was added in later work [21]. Our second twist is to take names in an infinite and totally ordered set of names fixed in advance, say $\mathcal{X}$. This helps comparing bigraphs with our SMC category.

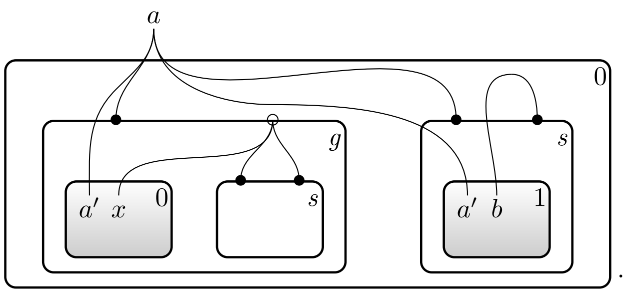
A *bigraphical signature* is a set of operations, or *controls* $k \in \mathcal{K}$, with arity given by a pair of natural numbers $a_k = (B_k, F_k) = (n, m)$. The number $B_k = n$

is the number of *binding* ports of k, $F_k = m$ being its number of *free* ports. Additionally, a signature specifies a set $\mathcal{A} \subseteq \mathcal{K}$ of *atomic* controls, whose binding arity has to be 0.

Typically, get and send have arities: $a_s = (0, 2)$ and $a_g = (1, 1)$. They are not atomic (send would be atomic in the asynchronous π-calculus). The other operations of the π-calculus are all kind of built into bigraphical structure, as we will see shortly.

Bigraphs form a category, whose objects are *interfaces*. An interface is a triple $U = (n, X, \ell)$, where n is a natural number, $X \subseteq \mathcal{X}$ is a finite set of names, and $X \xrightarrow{\ell} n + \{\bot\}$ is a *locality* map (n is identified with the set $\{0, \ldots, n-1\}$, i.e., the ordinal n). Names x with $\ell(x) = i \in n$ are *located* at i; others are *global*.

Introducing the morphisms, i.e., bigraphs, themselves seems easier by example. We thus continue with an example bigraph, which will correspond to the proof net in Figure 2:



. (4)

The codomain of this bigraph, which is graphically its outer face, is $W = (1, \{a\}, \{a \mapsto \bot\})$: the element $0 \in 1$ represent the (only) outer box, which we accordingly marked 0. The global name a is the common end of the group of four wires reaching the exterior of the box.

The domain of our example bigraph, which is graphically its inner face when the grey parts are thought of as holes, is $U = (2, \{a', x, b\}, \{x \mapsto 0, b \mapsto 1, a' \mapsto \bot\})$. Comparing this to the domain of our morphism in Figure 2, we observe that the elements 0 and 1 of 2 correspond to $\Box_0$ and $\Box_1$. Furthermore, the name a' being global corresponds to the domain $v \multimap ((v \multimap t) \otimes (v \multimap t))$ of Figure 2 having both t's *under the scope* of the first v (i.e., there is a $\multimap$ with t on its right, v on its left, and no other implication on the paths from it to them). Finally, the locality map sending x to 0 corresponds to the second v having only the first t under its scope, and similarly for b being sent to 1.

The morphism itself is a compound of two graphical structures. The first structure, the *place* graph, is a forest (here a tree), whose leaves are the inner 0 and 1, the *sites*, and whose root is the outer 0. Following Milner and Jensen, we represent nodes by regions in the plane, the parent of a region being the immediately enclosing region. The second structure, the *link* graph, is a bit more complicated to formalise. First, each internal (i.e., non leaf, non root) node v is labelled with an operation $k_v \in \mathcal{K}$. We then compute the set of *ports* P: it is the set of pairs (v, i), where v is a node, and $i \in B_{k_v} + F_{k_v}$ is in either component of the arity of v. The link graph is then a function $P + X \xrightarrow{link} E + Y$, where $X = \{a', x, b\}$ is the set of *inner* names, $Y = \{a\}$ is the set of *outer* names, and E is the set of *edges*. In our morphism, a' and both occurrences of a are mapped to the outer name a by the *link* map. Furthermore, E is a two-element set, say $\{x', b'\}$. The edge x' links the name x received by the get node g to its three occurrences. Formally, the three involved ports and the name x are all sent to x' by the *link* map. The edge b' represents the νb in the term; formally, both b and the involved port of the right-hand s node are sent to b' by the *link* map.

Until now, there is not much difference between the edge representing the bound name x received on a and the bound name b created by νb. The difference comes in when we check the *scope* and *binding* rules. The binding rule requires that each binding port (such as the one marked with a circle in (4)) be sent to an edge, as opposed to a name in the codomain. The scope rule further requires that its *peers*, i.e., the ports and names connected to the same edge, lie strictly below it in the place graph. For ports, this should be clear. For inner names, this means that they should be located at some site below it. In our example, the inner 0 node indeed lies below the get node, for instance. This all ensures that bound names are only used below their binder.

Remark 1. An edge is connected to at most one binding port, by acyclicity of the place graph. An edge connected to one binding port is called *bound*.

Composition $g \circ f$ in the category of bigraphs $M(\mathcal{K})$ is by plugging the outer boxes of f into the inner boxes of g, in order, and connecting names straightforwardly. This only works if we quotient out bigraphs by the natural notion of isomorphism, i.e., modulo choice of nodes and edges. We actually consider a further quotient: removing an edge from E which was outside the image of *link*. The whole is called *lean support* equivalence by Jensen and Milner.

4.2 Bigraphs as smc theories

We now describe our SMC theory for bigraphs, starting with the translation of signatures. Consider any signature $(\mathcal{K}, B, F, \mathcal{A})$. We translate it into the following SMC signature $\mathcal{T}_{\mathcal{K}}$, which has two sorts $\{t, v\}$, standing for terms and variables (or names), and whose operations consist of *structural* operations and equations, plus *logical* operations. The *structural* part, accounting for the built-in structure of bigraphs, is as in Section 3.3, i.e., it consists of

- a commutative monoid structure $(|, \mathbf{0})$ on t,
- a commutative comonoid structure (c, w) on v, and
- a name restriction $I \xrightarrow{\nu} v$, such that $w \circ \nu = \mathrm{id}_I$.

The *logical* part consists, for each $k \in \mathcal{K}$ with $a_k = (n, m)$, of an operation $v^{\otimes m} \xrightarrow{k} t$ if k is atomic (in which case $n = 0$), and $(v^{\otimes n} \multimap t) \otimes v^{\otimes m} \xrightarrow{k} t$ otherwise.

For example, recall send and get, defined above to have arities $(1, 1)$ and $(0, 2)$, this gives exactly the operations $(v \otimes v \otimes t) \xrightarrow{s} t$ and $(v \otimes (v \multimap t)) \xrightarrow{g} t$ from

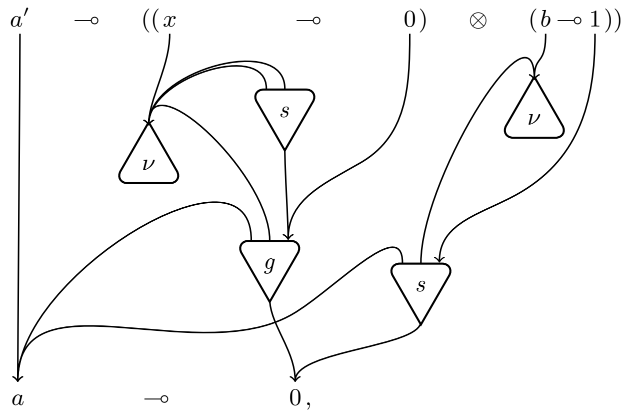


Fig. 3. A hybrid picture between bigraphs and proof nets

Section 2.3. An atomic get operation, in the style of the asynchronous π-calculus, would have the same bigraphical arity, translated into $v \otimes v \xrightarrow{g'} t$.

Now, on objects, we define our functor T by:

$$\mathsf{T}(n, X, \ell) = v^{\otimes n_g} \multimap \bigotimes_{i \in n} (v^{\otimes n_i} \multimap t), \tag{5}$$

where $n_g = |\ell^{-1}(\bot)|$ and for all $i \in n$, $n_i = |\ell^{-1}(i)|$. The ordering on $\mathcal{X}$ induces a bijection between X and v leaves in the formula, which the translation of morphisms exploits. On our main example, this indeed maps the domain and codomain of (4) to those of Figure 2.

We will here only describe the translation of morphisms on (4), for readability. The full translation is available in a companion preprint [15]. Starting from (4), a first step is to represent the place graph more traditionally, i.e., as usual with trees. But in order to avoid confusion between the place and link graphs, we represent each node as a cell, and adopt the convention that edges from the place graph relate a principal port to a rightmost auxiliary port. Wires from the link graph thus leave from other auxiliary ports.

Finally, edges in E in the bigraph are pointed to by ports and inner names. We now represent them as (nullary) ν cells with pointers to their principal port. We obtain the hybrid picture in Figure 3, where we have drawn the connectives to emphasise the relationship with Figure 2. And indeed we have almost obtained the desired proof net. A first small problem is the direction of wires in the linking graph which, intuitively, go from occurrences of names to their creator (be it a ν or an outer name). So we start by reversing the flow.

But that does not completely correct the mismatch, because in the case of bound edges like x' in our example bigraph, the ν cell is absent in proof nets. But by Remark 1, the name in question has a unique binding occurrence, and the ν cell may be understood as an indirection between this binding occurrence and the others. Contracting this indirection (and fixing the orientation accordingly) yields exactly the desired proof net in Figure 2.

The procedure sketched on our example generalises, up to some subtleties with unused names, and we have

Theorem 3. *This yields a functor $M(\mathcal{K}) \xrightarrow{\mathsf{T}} S(\mathcal{T}_{\mathcal{K}})$, which is faithful, essentially injective on objects, and neither full nor surjective on objects.*

The functor is not strictly injective on objects, because two isomorphic interfaces differing only by the choice of their set of names have the same image under T.

The functor T being non-full means that even between bigraphical interfaces, $S(\mathcal{T}_{\mathcal{K}})$ contains morphisms which would be ill-scoped according to Milner's scope rule. So it seems useful to verify that the overall scoping discipline is maintained. This is indeed the case, in the sense that T is full on whole programs, i.e., bigraphs with no sites nor open names. Formally:

Theorem 4. *The functor T induces an isomorphism on closed terms, i.e., an isomorphism of hom-sets $S(\mathcal{T}_{\mathcal{K}})(I, t) \cong M(\mathcal{K})((\emptyset, 0, \emptyset), (\emptyset, 1, \emptyset))$.*

So, $S(\mathcal{T}_{\mathcal{K}})$ has as many whole programs as $M(\mathcal{K})$, but more program fragments.

5 Conclusions

Related work Various flavours of closed categories have long been known to be closely related to particular calculi with variable binding [19, 1]. As mentioned in the introduction, our approach may be considered as an update and further investigation of Coccia et al. [4]. We should also mention Tanaka's work on variable binding in a linear setting [26], whose relation to the present work remains unclear to us.

A number of papers have been devoted to better understanding bigraphs, be it as sortings [7], as cospans over graphs [25], as a compact closed category [13], or as a language with variable binding [5]. We appear to be the first to reconcile a full treatment of scope (Theorem 4) with algebraic tools, i.e., seeing bigraphs as satisfying a universal property.

Future work We should try to push our approach further, e.g., by trying to use it in an actual implementation. Also, we here only handle abstract bigraphs, which do not have so-called *relative pushouts* (RPO). We thus should generalise our approach to deal with *concrete* bigraphs, be it in the form of Milner's original pre-category or of Sassone and Sobociński's *G-categories* [24], and then try to construct the needed (G)RPOs.

Another natural research direction from this paper concerns the dynamics of bigraphs. Our hope is that Bruni et al.'s [2] very modular approach to dynamics may be revived, and work better with SMC structure than with cartesian closed structure. Specifically, with SMC structure, there is no duplication at the static level, which might simplify matters.

References

[1] A. Barber, P. Gardner, M. Hasegawa, G. Plotkin. From action calculi to linear logic. In *Annual Conference of the European Association for Computer Science Logic (CSL'97), Aarhus, August 1997, Selected Papers*, vol. 1414 of *Lecture Notes in Computer Science*. Springer, 1998.
[2] R. Bruni, U. Montanari. Cartesian closed double categories, their lambda-notation, and the pi-calculus. In *LICS '99: Proc. 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1999.
[3] L. Cardelli, A. Gordon. Mobile ambients. In M. Nivat, ed., *Foundations of Software Science and Computational Structures*, vol. 1378 of *Lecture Notes in Computer Science*. Springer, 1998.
[4] M. Coccia, F. Gadducci, U. Montanari. GS·Λ theories: A syntax for higher-order graphs. In *CTCS'02, Category Theory and Computer Science*, vol. 69 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.
[5] T. Damgaard, L. Birkedal. Axiomatizing binding bigraphs. *Nordic Journal of Computing*, 13(1–2), 2006.
[6] V. Danos, L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28, 1989.
[7] S. Debois. *Sortings & bigraphs*. PhD thesis, IT University of Copenhagen, 2008.
[8] J. Despeyroux, A. Felty, A. Hirschowitz. Higher-order abstract syntax in Coq. In *TLCA '95*, vol. 902 of *Lecture Notes in Computer Science*. Springer, 1995.
[9] M. Fiore, G. Plotkin, D. Turi. Abstract syntax and variable binding. In *LICS '99: Proc. 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1999.
[10] M. J. Gabbay, A. M. Pitts. A new approach to abstract syntax involving binders. In *14th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, Washington, 1999.
[11] R. Garner, T. Hirschowitz, A. Pardon. Graphical presentations of symmetric monoidal closed theories.
[12] J.-Y. Girard. Linear logic. *Theoretical Comput. Sci.*, 50, 1987.
[13] D. Grohmann, M. Miculan. Directed bigraphs. *Electr. Notes Theor. Comput. Sci.*, 173, 2007.
[14] A. Hirschowitz, M. Maggesi. Modules over monads and linearity. In *Logic, Language, Information and Computation, 14th International Workshop, WoLLIC 2007, Proceedings*, vol. 4576 of *Lecture Notes in Computer Science*. Springer, 2007.
[15] T. Hirschowitz, A. Pardon. Binding bigraphs as symmetric monoidal closed theories.
[16] D. J. D. Hughes. Simple free star-autonomous categories and full coherence. ArXiv Mathematics e-prints, math/0506521, 2005.
[17] O. H. Jensen, R. Milner. Bigraphs and mobile processes (revised). Technical Report TR580, University of Cambridge, 2004.
[18] Y. Lafont. Interaction nets. In *POPL*, 1990.
[19] J. Lambek, P. Scott. *Introduction to Higher-Order Categorical Logic*. Number 7 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1986.
[20] P.-A. Melliès. Double categories: a modular model of multiplicative linear logic. *Mathematical Structures in Computer Science*, 12, 2002.
[21] R. Milner. Bigraphs whose names have multiple locality. Technical Report TR603, University of Cambridge, 2004.
[22] R. Milner, J. Parrow, D. Walker. A calculus of mobile processes. *Information and Computation*, 100(1), 1992.
[23] F. Pfenning, C. Elliott. Higher-order abstract syntax. In *ACM SIGPLAN '88 Symposium on Language Design and Implementation*. ACM, 1988.
[24] V. Sassone, P. Sobociński. Deriving bisimulation congruences using 2-categories. *Nordic Journal of Computing*, 10(2), 2003.
[25] V. Sassone, P. Sobociński. Reactive systems over cospans. In *Logic in Computer Science, LiCS '05*. IEEE Press, 2005.
[26] M. Tanaka. Abstract syntax and variable binding for linear binders. In M. Nielsen, B. Rovan, eds., *MFCS*, vol. 1893 of *Lecture Notes in Computer Science*. Springer, 2000.
[27] T. Trimble. *Linear logic, bimodules, and full coherence for autonomous categories*. PhD thesis, Rutgers University, 1994.
[28] C. Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. PhD thesis, University of Oxford, 1971.