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, 15]. 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 HOAS [27, 8, 17]. 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 construction yields a monadic adjunction, and hence provides an initial algebra semantics for variable binding. The morphisms of $S(\mathcal{T})$ look like abstract syntax, e.g., in the sense of Wadsworth's λ-graphs [33]. Technically, they are a variant of proof nets in intuitionistic multiplicative linear logic [13] (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 stable under composition. Finally, a standard issue in variable binding is induction. We propose a general induction principle derived from Girard's sequentialisation theorem [13, 6].

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 languages with variable binding, there is a standard notion of term with a hole, or *context*, and an associated operation of hole-filling, or substitution with capture. Morphisms of $S(\mathcal{T})$ and their composition are very close to contexts and hole-filling, except that hole-filling is generally total, while composition only concerns, well, composable morphisms. And moreover, contexts are here generalised to be multi-hole and higher-order (holes with holes, and so on). 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, 15, 10]. Conversely, non-linear capture-avoiding substitution requires a bit more work in our setting. We only sketch it here, and briefly discuss alternatives to the general induction principle of Section 2. 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 [20] 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 [32] and Hughes [19].

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 is 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) graph whose vertices are IMLL formulae over X, as defined by:

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

where $\otimes$ is *tensor* and $\multimap$ is *(linear) implication.*

Example 1. 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:

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

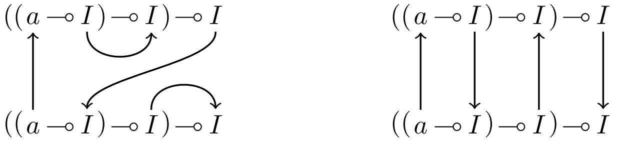
A morphism of signatures $(X, \Sigma) \longrightarrow (Y, \Sigma')$ is a function $X \xrightarrow{f} Y$, equipped with a morphism of graphs, whose vertex component is "$\mathcal{F}(f)$", i.e., the function sending any formula $A(x_1, \ldots, x_n)$ to $A(f(x_1), \ldots, f(x_n))$. This defines a category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

There is 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 [19] has devised a simple presentation of $S(X)$. Consider for a guiding example the two endomorphisms of $((a \multimap I) \multimap I) \multimap I$:



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

(domain on top for the whole paper, the right-hand morphism is 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[1]. 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 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 (i) it is a total function, and (ii) it satisfies the Danos-Regnier (DR) criterion [6]. The latter roughly goes as follows. An IMLL formula may be translated to a *classical* formula, as defined by the grammar:

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

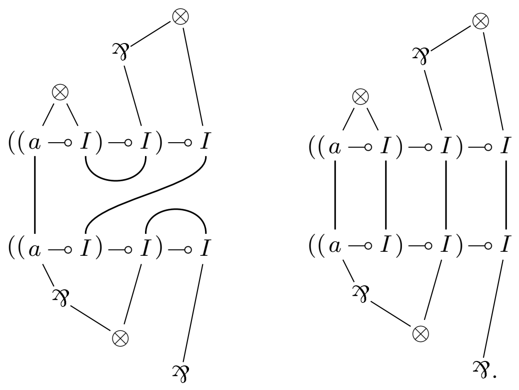

$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$
$((a \multimap I) \multimap I) \multimap I$

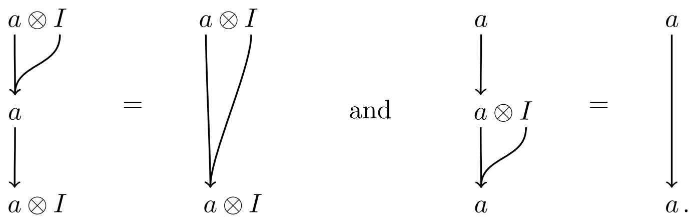
Fig. 1. Example switching

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

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, minus exactly one argument edge of each $\parr$. 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, sample switchings are depicted in Fig. 1.

Correct linkings compose by glueing along ports in the middle formula, which yields a category $S_0(X)$. For example, pre and postcomposing the structural isomorphism ρ with its obvious candidate inverse yields:



(1)

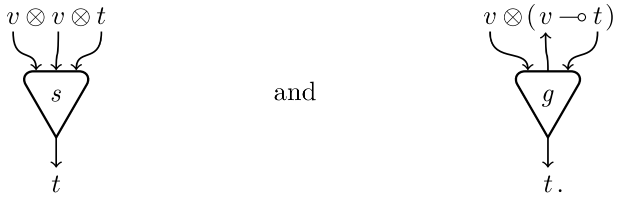
The former is not an identity. And indeed, correct linkings do not form an SMC category. Instead, they form the free *split* SMC category over X [19]. A split SMC category is like an SMC category, where λ and ρ are only required to have left inverses, as exemplified with ρ in (1).

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. A first example is that the left-hand morphism of (1) rewires to the identity; a hopefully more intuitive example is in Section 3.1. Then $S(X)$ is the result of quotienting $S_0(X)$ by the equivalence relation generated by rewiring.

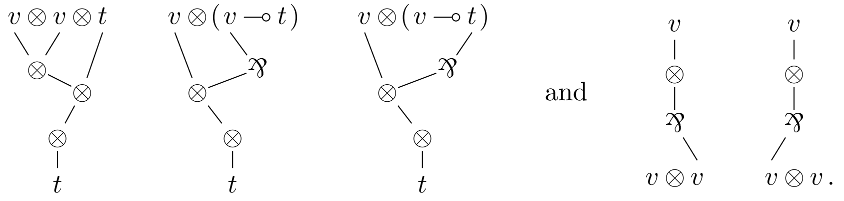
2.3 The Free Symmetric Monoidal Closed Category over a Signature

We now extend S to SMC signatures Σ: we have a set of sorts X, plus a set of operations. We enrich linkings with, for each operation $A \xrightarrow{c} B$, a formal morphism, pictured by a *cell*, in the style of *interaction nets* [21].

Example 2. The π-calculus send and get operations yield cells



We then extend linkings $A \longrightarrow B$ to include cells in a suitable way — a glance at Fig. 2 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:



To understand why this is right, observe that SMC categories have a *functional completeness* property, in the sense of Lambek and Scott [22]. Roughly, this means that any morphism $C \longrightarrow D$ 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. An example such decomposition is pictured (left) in Fig. 4 below. This rightly suggests that an operation $A \xrightarrow{c} B$ should have the same switchings as $A \multimap B$ in the domain, i.e., $A \otimes B^{\perp}$. Quotienting under rewiring as before yields the expected functor S:

Theorem 1. *The functors S and U yield a monadic adjunction*

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

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 the left-hand part of Fig. 4) 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 $(g_1, g_2) \in E_{A,B}$. The smallest equivalence relation by this relation $\sim$ is stable under composition, and we define $S(\mathcal{T})$ by quotienting $S(\Sigma)$ under it.

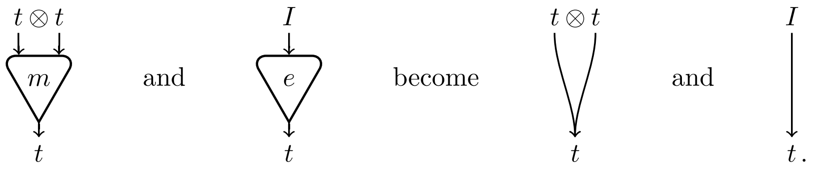
Finally, $S(\mathcal{T})$ is initial in the following sense. Let the category of *representations* of $\mathcal{T}$ be the full subcategory of the comma category $\Sigma \downarrow U$ whose objects are the morphisms $\Sigma \longrightarrow U(\mathcal{C})$, for which $\mathcal{C}$ is an SMC category satisfying 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 t ports to positive t ports non-bijectively, and then replace m and e as follows:



For a commutative comonoid (c, w), the dual trick does not work so easily, because of problems with weakening (weakening has an output I port, which cannot be left unattached, as opposed to the input I port of e). 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 and Sequentialisation as Induction

Melliès [24] 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 (representative of a) 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 $f_2 \circ f_1$, where each f_i contains exactly the cells in C_i.*

The proof is by inductively applying the decomposition (2). Intuitively, Proposition 1 says that, thinking of operations in Σ as atomic building blocks, each morphism may be obtained by plugging such blocks together by composition (see the left-hand part of Fig. 4). In a sense, this is an induction principle (a morphism only has finitely many cells). But it does not prevent wires to have a complex structure in the obtained components. We thus need a more powerful induction principle.

Let us fix a theory $\mathcal{T} = (\Sigma, E)$ for the rest of this section. A first, general induction principle on $S(\mathcal{T})$ is given by sequentialisation, in the sense of Girard [13, 6], as follows. Pick your preferred syntactic presentation of IMLL, e.g., the original sequent calculus [13] for concreteness. There is a well-known mapping from axiom-free proofs to proof nets, say $p \mapsto [p]$. Observe that by understanding the operations $A \xrightarrow{c} B$ in Σ as axioms $c\colon A \vdash B$, this mapping extends to proofs with axioms in Σ, by mapping any such c to a corresponding cell. We then have:

Theorem 3. *Any morphism $A \xrightarrow{f} B$ in $S(\mathcal{T})$ has an antecedent proof p in* IMLL *plus axioms in Σ, such that $[p] = f$.*

This provides an easy induction scheme over the morphisms of $S(\mathcal{T})$. However, this scheme has deficiencies: for example, the antecedent proofs it provides do not have to be cut-free; moreover, morphisms may not be decomposed downwards as with standard induction schemes. For an example of the latter, assume you have a λ-term starting with a λ-abstraction; the induction scheme might reveal this only after a few decompositions.

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.

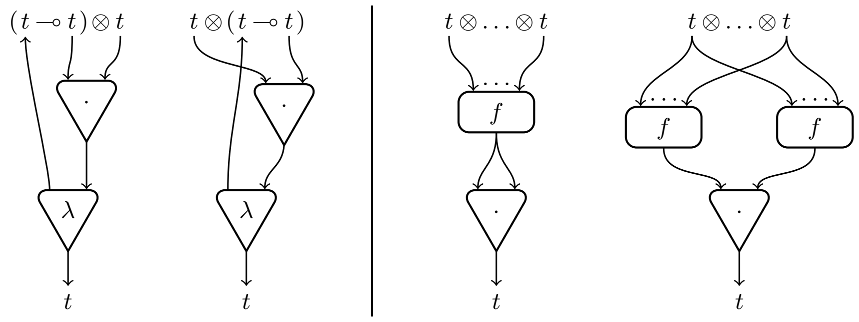


Fig. 2. Examples: linearity and sharing

We briefly discuss induction principles in this particular case. We then proceed with a π-calculus example, which we will use as our main example in Section 4. We end by a few example uses of higher order and modularity (Proposition 1), notably related to reduction and labelled transition rules.

3.1 Lambda-Calculus, Linearity, Induction

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 two morphisms on the left of Fig. 2.

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 [33], which are a kind of λ-terms with a fine representation of sharing. For example, the two morphisms on the right of Fig. 2 are different, because contraction is not natural.

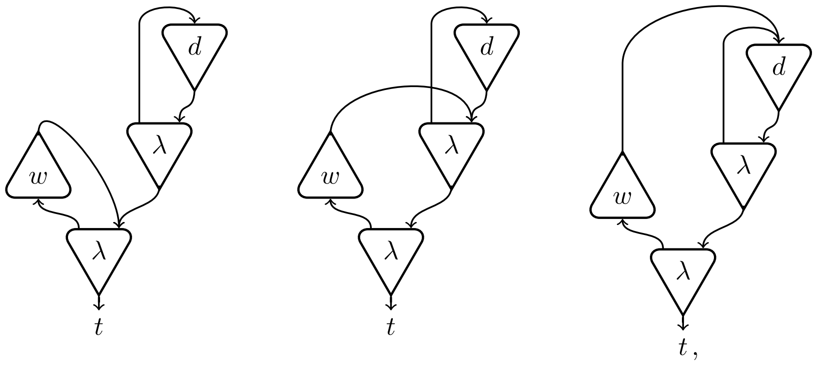
To obtain the standard λ-calculus without sharing, we now consider two sorts: a sort t for terms, and a sort v for variables, an idea that has been explored independently in *weak* HOAS and tile logic [8, 17, 18, 2]. The theory then contains:

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

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

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

The commutative comonoid structure on v, and Trimble rewiring are crucial to this result. Intuitively, the latter allows a weakened variable to be indifferently linked anywhere under its scope. For example, the term $\lambda x.\lambda y.y$ has linkings



which all equivalent under Trimble rewiring.

Beyond Proposition 3, we also may recover open terms as follows. Mimicking the standard construction of a monad from an operad using coends [23], for any set X, let $T(X)$ contain triples of a natural number n, a function from the ordinal n to X, and a morphism $t^{\otimes n} \longrightarrow t$, where $t^{\otimes 0} = I$ and $t^{\otimes n+1} = t^{\otimes n} \otimes t$. Two such triples (n, u, f) and (n, v, g) are considered equivalent when there is a permutation $\sigma\colon n \longrightarrow n$ such that $g = f \circ \sigma$ and $v = u \circ \sigma$ (finite ordinals and permutations form a subcategory of $S(\mathcal{T})$ through the embedding $n \mapsto t^{\otimes n}$).

Theorem 4. *The function T extends to a monad on* Set, *isomorphic to the monad sending each set X to the set of λ-terms with variables in X modulo α-conversion.*

Multiplication for this monad, i.e., substitution, works as follows: for any triple of a number n, a morphism $t^{\otimes n} \xrightarrow{f} t$, and a function $n \xrightarrow{u} T(X)$, let each $u_i = (n_i, v_i, f_i)$, and $m = \Sigma n_i$. Multiplication maps (n, u, f) to m, the morphism

$$t^{\otimes \Sigma n_i} \xrightarrow{\bigotimes f_i} t^{\otimes n} \xrightarrow{f} t$$

and the coproduct function $\Sigma n_i \xrightarrow{[v_0, \ldots, v_{n-1}]} X$.

We may derive from Theorem 4 an analogue of the standard induction scheme on λ-terms. We also expect to derive it directly, but for now defer a full treatment for further work. Also, it seems worth investigating sufficient conditions on the signature for such a general induction scheme to be derived. For instance, it is not at all obvious which induction scheme should be derived from the signature we choose in the next section for the π-calculus.

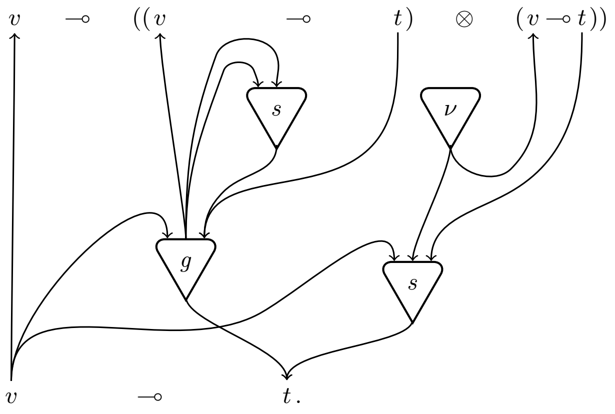


Fig. 3. A π-calculus example

3.2 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 $(v \multimap t) \longrightarrow t$ [2, 18].)

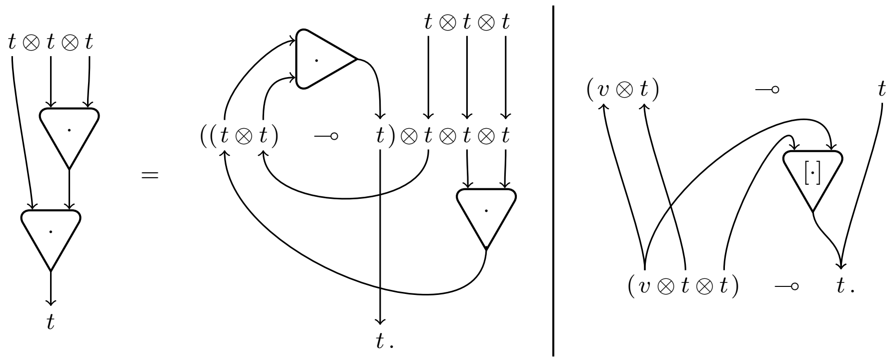
Consider the π-calculus term with holes $(a(x).(\square_0 \mid \bar{x}\langle x \rangle)) \mid \nu b.(\bar{a}\langle b \rangle.\square_1)$. This term may have many different interpretations as a morphism in $S(\mathcal{T})$. A first possibility is depicted in Fig. 3. Recall: several arrows leaving a v port mean a tree of contractions; several arrows entering a t port mean a tree of parallel compositions; a positive t port with no input arrow means a 0.

The holes $\square_0$ and $\square_1$ are represented by the occurrences of t in the domain formula, in order. 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).

3.3 Higher Order and Modularity

We now give an example decomposition as in Proposition 1, in the λ-calculus. Consider the context with numbered holes $(\square_0 \cdot \square_1) \cdot \square_2$. The decomposition obtained by Proposition 1 for C_1 containing exactly the outermost application is depicted left in Fig. 4. Such a decomposition is not possible in bigraphs, mainly because it makes use of a higher-order formula, namely $(t \otimes t) \multimap t$.



$t \otimes t \otimes t$
t
$=$
$t \otimes t \otimes t$
$((t \otimes t) \multimap t) \otimes t \otimes t \otimes t$
$t\,.$
$(v \otimes t) \multimap t$
$[\cdot]$
$(v \otimes t \otimes t) \multimap t\,.$

Fig. 4. Examples: modularity and higher order

A possible use of such decompositions and higher order is in specifying reduction rules parametrically, as opposed to ground reduction rules. For example, the π-calculus rule $a(x).\,\square_0\,|\bar{a}\langle x\rangle.\,\square_1 \longrightarrow \square_0\,|\square_1$ may be represented as a rule between morphisms looking like Fig. 3, which we omit for lack of space. Another possible use is in specifying transitions using second-order contexts, in the style of [28, 12]. An example, using Cardelli and Gordon's Mobile Ambients [3], is the transition rule $\mathtt{in}\,a.P \xrightarrow{\lambda Q.(a[Q] \mid \square)} a[P|Q]$, whereby the process $\mathtt{in}\,a.P$, in the presence of a process of the shape $a[Q]$, migrates inside the location a, to yield $a[P|Q]$. A possible representation using SMC theories would take as states of the labelled transition system morphisms $I \xrightarrow{f} A$ in the free SMC category generated by the obvious SMC theory for Mobile Ambients, with as transitions $f \xrightarrow{\ell} g$ certain morphisms $A \xrightarrow{\ell} B$ such that $\ell \circ f = g$. In the above example, $\mathtt{in}\,a.P$ is represented a morphism $I \longrightarrow (v \otimes t) \multimap t$ obtained by currying the operation $v \otimes t \xrightarrow{\mathtt{in}} t$ from the signature, and the label is the morphism depicted right in Fig. 4.

4 Binding Bigraphs

In this section, we consider (abstract binding) bigraphs [20]. They are a framework for reasoning about distributed and concurrent programming languages, designed to encompass both the π-calculus [26] and the Mobile Ambients calculus [3]. We are here only concerned with bigraphical syntax: any so-called *bigraphical signature* $\mathcal{K}$ generates a *pre-category*, and then a category $M(\mathcal{K})$, whose objects are *bigraphical interfaces*, and whose morphisms are bigraphs.

Its main features are (i) the presence of *relative pushouts* (RPOs) in the pre-category, which makes it well-behaved w.r.t. bisimulations, and that (ii) in both the pre-category and the category, the so-called *structural* equations become equalities. Also, bigraphs follow a scoping discipline ensuring that, roughly, bound variables are only used below their binder.

We now recall bigraphs and sketch our interpretation in terms of SMC theories, which we compare to the original (see our preprint [16] for a more technical account).

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 [20][2]. It was added in later work [25]. Our second twist is to take names in a fixed, infinite, and totally ordered set, say $\mathcal{X}$. This helps relating our approach with the original.

A *bigraphical signature* is a set of operations, or *controls* $k \in \mathcal{K}$, with arity given by a pair of natural numbers $a_k = (B_k, F_k) = (n, m)$, where $B_k = n$ 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, send and get 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 $X \xrightarrow{\ell} n + \{\perp\}$ 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 in Fig. 5, which will correspond to the proof net in Fig. 3. The codomain of this bigraph, which is graphically its upper, outer face, is $W = (1, \{a\}, \{a \mapsto \perp\})$: the element $0 \in 1$ represents the (only) outer box, which we accordingly marked 0. The global name a is the common end of the group of three wires reaching the top side of the box.

The domain of our example bigraph, which is graphically its inner face when the grey parts are thought of as holes (plus a'), is $U = (2, \{a', x, b\}, \{x \mapsto 0, b \mapsto 1, a' \mapsto \perp\})$. Comparing this to the domain of our morphism in Fig. 3, we observe that the elements 0 and 1 of 2 correspond to $\square_0$ and $\square_1$. Furthermore, the name a' being global corresponds to the domain $v \multimap ((v \multimap t) \otimes (v \multimap t))$ of Fig. 3 having both t's *under the scope* of the first v (i.e., there is an implication with the t's 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

[2] Being peers only involves inner names or ports by definition, not outer names. Thus, binding ports may be linked to them. It is then easy to show that the scope rule is not stable under composition. The binding rule [25] is the straightforward fix.

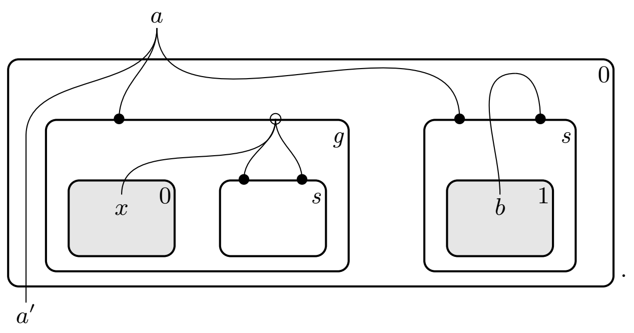


Fig. 5. An example bigraph

0 and 1, the *sites*, and whose root is the outer 0. (Atomic controls would have to be leaves.) 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 (i.e., the black dots connected to a in the picture) are mapped to the outer name a by the *link* map. Furthermore, E is a two-element set, say $\{x', b'\}$. The edge x' acts as a link from the name x received by the get node g to its three occurrences (the three dots connected to it in the picture). 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 Fig. 5) 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 Symmetric Monoidal Closed Theories

We now describe our intepretation for bigraphs, starting with 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 3.2, 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 control $k \in \mathcal{K}$ with $a_k = (n, m)$, of an operation $v^{\otimes m} \xrightarrow{k} t$ if k is atomic (and $n = 0$), 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 (up to isomorphism) the operations $(v \otimes v \otimes t) \xrightarrow{s} t$ and $(v \otimes (v \multimap t)) \xrightarrow{g} t$ from Section 2.3. An asynchronous send operation in the style of the asynchronous π-calculus, would have bigraphical arity $(0,2)$, which would be translated into $v \otimes v \xrightarrow{s'} t$ because of atomicity.

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

$$\mathsf{T}(n, X, \ell) = v^{\otimes n_g} \multimap \bigotimes_{i \in n} (v^{\otimes n_i} \multimap t), \tag{3}$$

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 Fig. 5 to those of Fig. 3.

We will here only describe the translation of morphisms on Fig. 5, for readability. The full translation is available in the companion preprint [16]. Starting from Fig. 5, 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 (i.e., the vertex of a cell) to a rightmost auxiliary port (i.e., a rightmost point in the opposed segment). 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 Fig. 6, where we have drawn the connectives to emphasise the relationship with Fig. 3. And indeed we have almost obtained the desired proof net. A first small problem is the direction of wires in the link 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 of the link graph (implicitly introducing trees of contractions).

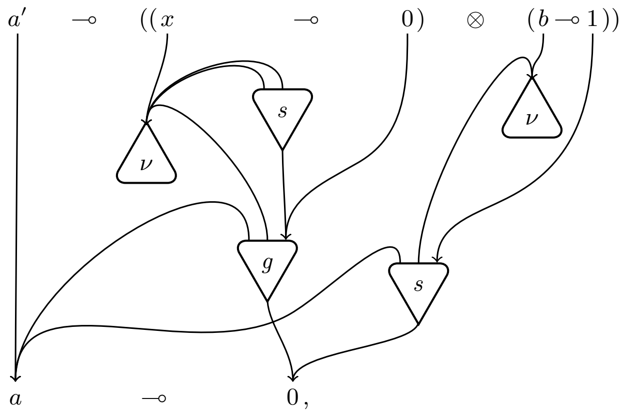


Fig. 6. A hybrid picture between bigraphs and proof nets

This does not completely correct the mismatch, however, because in the case of bound edges like x' in our example bigraph, the ν cell is absent from Fig. 3. 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 Fig. 3.

The procedure sketched on our example generalises, up to some subtleties with unused names (where should the weakenings point to?), and we have

Theorem 5. *The function* $\top$ *extends to a functor* $M(\mathcal{K}) \xrightarrow{\top} S(\mathcal{T}_{\mathcal{K}})$, *faithful, essentially injective on objects, and neither full nor surjective on objects.*

The functor is not strictly injective on objects, because any two interfaces equal up to their (ordered) choice of names have the same image. A counterexample to fullness is the canonical morphism $(v \multimap I \multimap t) \longrightarrow (I \multimap v \multimap t)$: it amounts to making a global variable local, which is forbidden in bigraphs.

Despite non-fullness, the overall scoping discipline of bigraphs is maintained, in the sense that $\top$ is full on whole programs, i.e., bigraphs with neither sites nor names in their interfaces. More generally, it is full on *ground* bigraphs, i.e., bigraphs in $M(\mathcal{K})((\emptyset, 0, \emptyset), U)$, for some interface U:

Theorem 6. *For any such* U*, we have* $S(\mathcal{T}_{\mathcal{K}})(I, \top(U)) \cong M(\mathcal{K})((\emptyset, 0, \emptyset), U)$.

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 [22, 1]. As mentioned in the introduction, our approach may be considered as an update and further investigation of Coccia et al. [4]. Also, the relation between our approach and Tanaka's work on variable binding in a linear setting [31] remains unclear to us.

A number of papers have been devoted to better understanding (various kinds of) bigraphs, be it as sortings [7], as cospans over graphs [30], through directed bigraphs [14], or as a language with variable binding [5]. We appear to be the first to reconcile a full treatment of scope (Theorem 6) with initial algebra semantics.

Future work. We should further investigate induction principles in our setting (see Section 3.1). We should also try to use our approach in an actual implementation.

On the bigraphical side, it might be useful to understand the scope rule induced by our functor T in bigraphical terms. Also, we should study RPOs in our approach, possibly by investigating (any form of) *concrete* bigraphs [20, 29].

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] Barber, A., Gardner, P., Hasegawa, M., Plotkin, G.: From action calculi to linear logic. In: Nielsen, M. (ed.) CSL 1997. LNCS, vol. 1414. Springer, Heidelberg (1998)
[2] Bruni, R., Montanari, U.: Cartesian closed double categories, their lambda-notation, and the pi-calculus. In: LICS 1999. IEEE Computer Society, Los Alamitos (1999)
[3] Cardelli, L., Gordon, A.: Mobile ambients. In: Nivat, M. (ed.) FOSSACS 1998. LNCS, vol. 1378, p. 140. Springer, Heidelberg (1998)
[4] Coccia, M., Gadducci, F., Montanari, U.: GS·Λ theories: A syntax for higher-order graphs. In: CTCS 2002. ENTCS, vol. 69. Elsevier, Amsterdam (2003)
[5] Damgaard, T., Birkedal, L.: Axiomatizing binding bigraphs. Nordic Journal of Computing 13(1-2) (2006)
[6] Danos, V., Regnier, L.: The structure of multiplicatives. Archive for Mathematical Logic 28 (1989)
[7] Debois, S.: Sortings & bigraphs. PhD thesis, IT University of Copenhagen (2008)
[8] Despeyroux, J., Felty, A., Hirschowitz, A.: Higher-order abstract syntax in Coq. In: Dezani-Ciancaglini, M., Plotkin, G. (eds.) TLCA 1995. LNCS, vol. 902, pp. 124–138. Springer, Heidelberg (1995)
[9] Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. In: LICS 1999. IEEE Computer Society, Los Alamitos (1999)
[10] Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax involving binders. In: LICS 1999. IEEE Computer Society, Los Alamitos (1999)
[11] Garner, R.H.G., Hirschowitz, T., Pardon, A.: Graphical presentations of symmetric monoidal closed theories. CoRR, abs/0810.4420 (2008)
[12] Di Gianantonio, P., Honsell, F., Lenisa, M.: RPO, second-order contexts, and λ-calculus. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 334–349. Springer, Heidelberg (2008)
[13] Girard, J.-Y.: Linear logic. Theoretical Comput. Sci. 50 (1987)

[14] Grohmann, D., Miculan, M.: Directed bigraphs. ENTCS 173 (2007)
[15] Hirschowitz, A., Maggesi, M.: Modules over monads and linearity. In: Leivant, D., de Queiroz, R. (eds.) WoLLIC 2007. LNCS, vol. 4576, pp. 218–237. Springer, Heidelberg (2007)
[16] Hirschowitz, T., Pardon, A.: Binding bigraphs as symmetric monoidal closed theories. CoRR, abs/0810.4419 (2008)
[17] Hofmann, M.: Semantical analysis of higher-order abstract syntax. In: LICS 1999. IEEE Computer Society, Los Alamitos (1999)
[18] Honsell, F., Miculan, M., Scagnetto, I.: Pi-calculus in (co)inductive-type theory. Theor. Comput. Sci. 253(2) (2001)
[19] Hughes, D.J.D.: Simple free star-autonomous categories and full coherence. ArXiv Mathematics e-prints, math/0506521 (June 2005)
[20] Jensen, O.H., Milner, R.: Bigraphs and mobile processes (revised). Technical Report TR580, University of Cambridge (2004)
[21] Lafont, Y.: Interaction nets. In: POPL. ACM, New York (1990)
[22] Lambek, J., Scott, P.: Introduction to Higher-Order Categorical Logic. Cambridge University Press, Cambridge (1986)
[23] Mac Lane, S.: Categories for the Working Mathematician, 2nd edn. Graduate Texts in Mathematics, vol. 5. Springer, Heidelberg (1998)
[24] Melliès, P.-A.: Double categories: a modular model of multiplicative linear logic. Mathematical Structures in Computer Science 12 (2002)
[25] Milner, R.: Bigraphs whose names have multiple locality. Technical Report TR603, University of Cambridge (2004)
[26] Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes. Information and Computation 100(1) (1992)
[27] Pfenning, F., Elliott, C.: Higher-order abstract syntax. In: PLDI 1988. ACM, New York (1988)
[28] Rathke, J., Sobocinski, P.: Deconstructing behavioural theories of mobility. In: TCS 2008. Springer, Heidelberg (2008)
[29] Sassone, V., Sobociński, P.: Deriving bisimulation congruences using 2-categories. Nordic Journal of Computing 10(2) (2003)
[30] Sassone, V., Sobociński, P.: Reactive systems over cospans. In: LICS 2005. IEEE Press, Los Alamitos (2005)
[31] Tanaka, M.: Abstract syntax and variable binding for linear binders. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, p. 670. Springer, Heidelberg (2000)
[32] Trimble, T.: Linear logic, bimodules, and full coherence for autonomous categories. PhD thesis, Rutgers University (1994)
[33] Wadsworth, C.: Semantics and pragmatics of the lambda calculus. PhD thesis, University of Oxford (1971)