Comparing λ-calculus translations in Sharing Graphs *

Andrea Asperti[1] and Cosimo Laneve[2]

[1] Dip. di Matematica, P.za di Porta S. Donato, 5, 40127 Bologna, Italy.
[2] INRIA Sophia Antipolis, 2004 Route des Lucioles BP 93, 06902 Valbonne, France.

Abstract. Since Lamping's seminal work [Lam90] on optimal graph reduction techniques for the λ-calculus, several different translations based on the same set of control operators (sharing graphs) have been proposed in the literature [GAL92a, GAL92b, AL93a, As94]. In this paper we clarify the correspondence between all these translations, passing through the so called bus-notation [GAL92a]. Indeed all the sharing graph encodings turn out to be equivalent modulo the way of counting bus levels.

1 Introduction

In [Lam90], Lamping proposed a complex graph reduction technique for the λ-calculus that was optimal in the sense of Lévy [Le78]. Lamping's approach was revisited in [GAL92a], where a restricted set of control nodes and reduction rules was proved sufficient for the implementation. In [GAL92b] Gonthier-Abadi-Lévy pointed out a strong analogy between optimal reductions and Linear Logic [Gi86]. In particular, the optimal implementation of λ-calculus is actually a refinement of Girard's proof-net representation. This refinement just provides a *local implementation* of the non-linear operations over shared data (nets into *boxes*), which in proof-nets are performed as a single, global step. By their encoding of Linear Logic in sharing graphs, and some encoding of λ-calculus in Linear Logic, a third (slightly different) translation was implicitly obtained. This encoding has been used (and explicited) in [AL93a], in the much more general case of Interaction Systems.

At the same time of [GAL92b], Asperti proposed yet another and sensibly different translation of λ-calculus into sharing graphs [As94]. This latter translation is strongly connected to the dynamic algebra interpretation of Linear Logic [Re92, DR93] (a relation that has been recently formalized in [ADLR94]).

Up to now, no correspondence has been established in the literature between all these different translations. In this paper we prove that such differences may be explained, up to minor syntactic details, by the way of counting the nesting of boxes into proof-nets. Precisely, the innermost box into a proof-net may be considered as a box at "depth 0" or a box at "depth n", if there are n outer boxes. In order to formalize this intuition, we shall use a further sharing graph implementation of λ-calculus: the so-called *bus-notation* [GAL92a]. The bus notation

* Partially supported by the ESPRIT Basic Research Project 6454 - CONFER.

was introduced with the aim of reducing the number of interaction rules between control nodes. This is achieved by considering edges as set of wires (buses) and an n-indexed control node as affecting the n-th wire only. The above conventions for counting nestings of boxes are reflected into the two ways of counting wires: "from the left" or "from the right". So, if we have $k+1$ wires in the bus, and a node on a wire, this node can be seen as an operator at level n (assuming n wires on the right) or $k-n$ (when counting from the left).

The two different choices underlie respectively the translations in [As94] and [GAL92b], which therefore are mapped onto the same sharing graph in the bus notation. This also explains why exactly the same set of rewriting rules works in both cases.

We remark that this work could have been written in the framework of proof nets [Gi86]. Indeed the results presented here could be transposed to proof nets without any difficulty. We choose to stick to λ-calculus for simplicity sake.

We warn the reader that some general knowledge of the literature (in particular [GAL92a] and [As94]) is a prerequisite for understanding this paper.

2 Sharing Graphs

In [GAL92a] it was proved that Lamping's original set of operators (nodes) needed for the optimal implementation of λ-calculus could be reduced to only five (indexed) *control nodes*, described in the following picture:

(root) (erasing) (croissant) (bracket) (fan)

Actually, only three of them (fan, croissant and bracket) are really important. The fan node represents sharing (or duplication), while croissant and brackets are needed for the correct interaction of fans during the reduction. In Lamping's original intuition, brackets and croissants must surround fans, implicitly defining their scope. Two different kinds of "brackets" are needed, since they propagate in opposite directions. This is essentially equivalent to consider brackets and croissants as delimiting the extent of the box of Linear Logic (recall that the box is just a sort of global bracket surrounding a datum that can be duplicated or erased) (see [GAL92a, GAL92b]). This relation was furtherly stressed in [As94], where a tight relation between croissant, bracket and the two categorical operations associated with the comonad "!" of Linear Logic was proved.

Erasing nodes are needed in the case of lambda abstractions over variables not appearing in the body of the function; until we do not consider garbage collection rules, these nodes have no operational effect.

In [AL93a] it was shown that all these control nodes could be considered as an abstract set of operators for implementing sharing in virtually every class of higher order rewriting systems. This suggested the name of "sharing graphs", that we shall adopt in the following.

The rules governing the interactions between control operators are drawn in Figure 1. We shall also explicitly consider two "proper nodes", for application

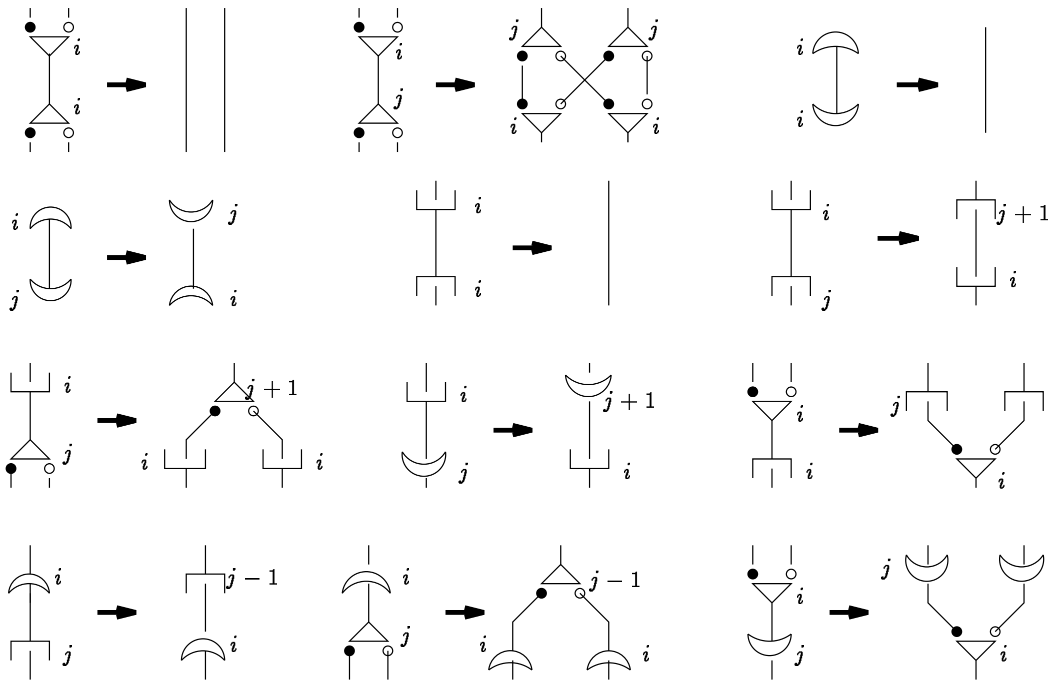


Fig. 1. The control rules ($0 \leq i < j$)

and lambda abstraction.

(abstraction) (application)

The abstraction and application nodes are *ternary* nodes, i.e. they are connected to three edges. In particular, in the case of abstraction nodes, one edge points to the root, one to the body of the function, and the third one to the bound variable. Thus, bound variables are supposed to be explicitly connected to their binders (an idea that goes back 7to Bourbaki).

As shown in [GAL92a], application and lambda nodes are not actually necessary, since from the operational point of view they can be assimilated to fan. Note in particular that the interaction between an application and an abstraction amounts to connect the root of the application to the root of the body of the function, and the bound variable to the argument of the application.

For the sake of clarity, we prefer to keep the distinction; however we shall not explicitly draw the interaction rules between proper operators and control nodes, that are exactly the same as for fans.

3 The box representation

The somewhat puzzling number of different translations of λ-calculus in sharing graphs is actually due to a double level of indeterminacy. Accepting the fact that the main problem is the optimal encoding of the box of Linear Logic [GAL92b], the first and evident choice is about different encodings of λ-calculus in Linear Logic. Up to now, two possible encodings have been investigated in the literature, respectively based on the type isomorphisms $D \cong !(D \multimap D)$ and $D \cong (!D) \multimap D$. The first one is adopted in [GAL92a], since it was closer to Lamping's original approach. The second one, is suggested by the "traditional" embedding of intuitionistic implication by means of linear implication [Gi86]. Both choices give rise to optimal implementations, due to the optimal implementation of the underlying Linear Logic (and many other encodings of λ-calculus into Linear logic would work as well). This level of indeterminacy in the translation is quite clear, and, actually is orthogonal to the subject of the paper. In the following we shall not consider it: in order to fix the ideas, we shall work with the more familiar type isomorphism $D \cong (!D) \multimap D$.

The second and more important choice is about the representation of the box in sharing graph. The translation function of [As94] is the function $\mathcal{G}$ in Figure 2(a) [3]. Remark that, according to the definition of $\mathcal{G}_n$, the box on the argument of the application is raised of one level w.r.t. the surrounding world. In [GAL92b] the box is "simulated" by adding a bracket of index 0 in front of

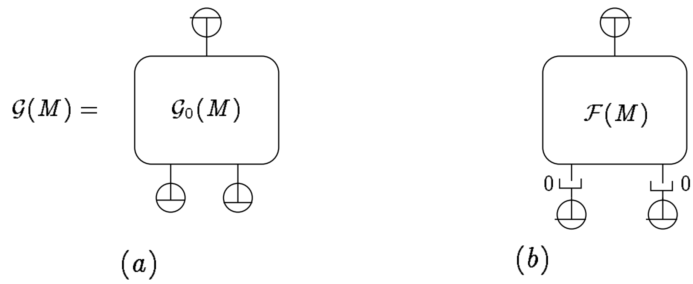


Fig. 2. The translation functions $\mathcal{G}$ and $\mathcal{F}$

its principal port (leaving the box at the same level of the context). This choice is described by the translation function $\mathcal{F}$ of Figure 4.

Note that the operational behaviour is sensibly different in the two cases. In particular, in [As94], the (external) control operators which are propagating inside the box are at a lower level than the (operators inside the) box. The contrary happens in [GAL92b] where the square bracket of index 0 on the principal

[3] Warning: in Figures 3 and 4, when the implementation of $\lambda x\, M$ is defined, we have assumed that $x \in \mathsf{fv}(M)$. The case $x \notin \mathsf{fv}(M)$ follows by connecting an erasing node to the port of the abstraction representing the bound variable.

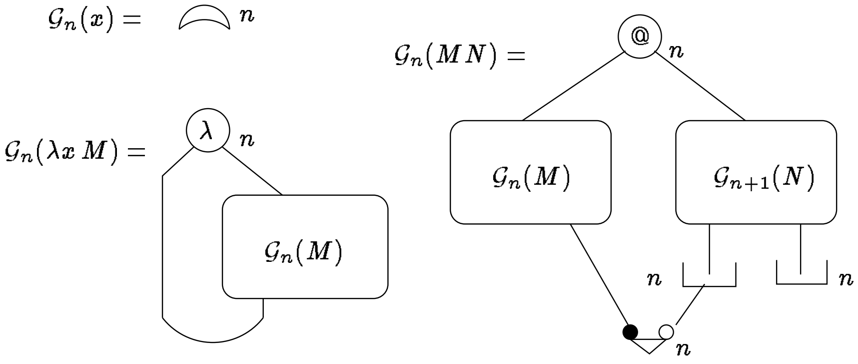


Fig. 3. The implementation of [As94]

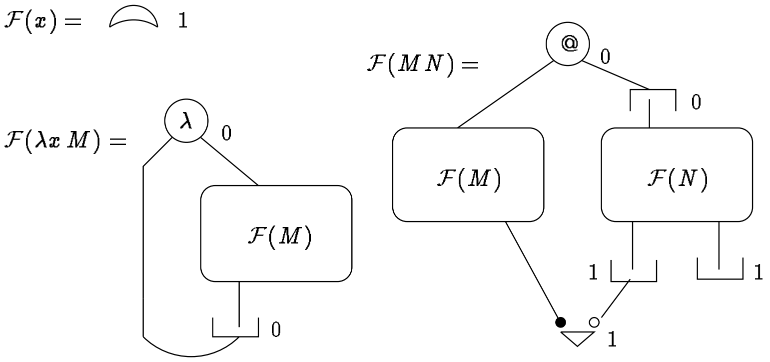


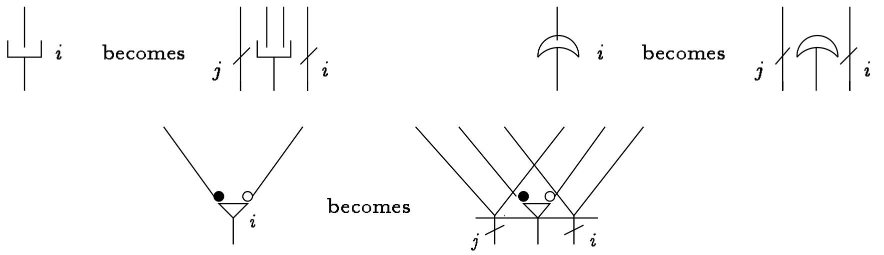
Fig. 4. The implementation of [GAL92b]

port of the box has the effect to raise the level of incoming control operators, avoiding their conflict with inner operators proper to the box. Due to this consideration, it can appear a bit surprising that *exactly the same set of rewriting rules* works in both cases. Moreover, apart the above informal discussion on the idea underlying the two box representations (i.e. their different ways to avoid conflicts), there is a priori no much evidence of a formal correspondence between them.

This is the main problem we shall address in this paper; to this purpose, we need to recall an interesting and odd notation, introduced in [GAL92a]: the *bus notation.*

4 The bus notation

A more explicit graph notation is obtained by interpreting edges in the systems $\mathcal{G}(M)$ and $\mathcal{F}(M)$ as buses, namely set of wires. In this view, an i-indexed node is considered as an operator acting on the wire at depth i. For instance:



where $_{j}\!\not|$ means a bus of width j. The rules governing the interaction of control nodes in the bus notation are depicted in Figure 5. Remark that no interaction

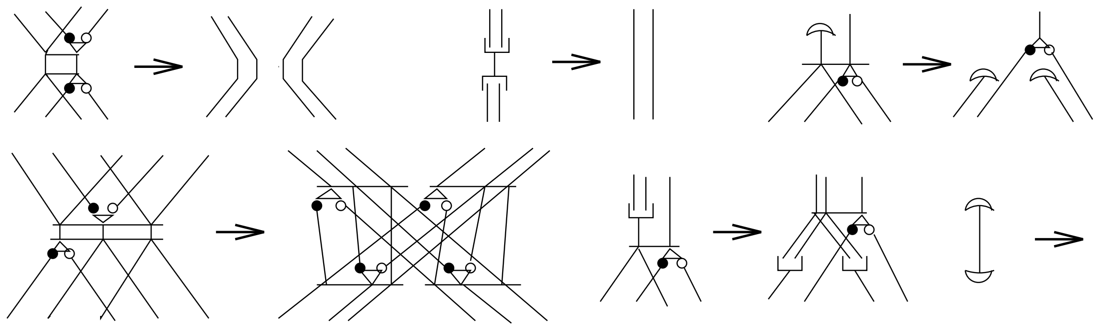


Fig. 5. The interactions of control nodes in the bus notation

between croissants or brackets on different wires is defined. This because the bus notation eliminates such commutations. Nevertheless it is not clear how much it is gained in efficiency with buses, since interactions of fans are no more local but concern every wire of the bus.

5 The bus notation of the translation $\mathcal{F}$

Trying an encoding *ad literam* through buses of the translation $\mathcal{F}$, we immediately realize that we cannot fix a priori the dimension of the bus. This is in contrast with what happens in [GAL92a] where the bus has always dimension 3; we shall come back soon to this point. For the moment, after having fixed some wire "0", we shall suppose to have an infinite numbers of wires at its left. The resulting translation is defined by the function $\mathsf{B}^{\mathcal{F}}$ in Figure 6 (note that $\mathsf{B}^{\mathcal{F}}(M)$ is exactly the bus-counterpart of $\mathcal{F}(M)$). Remark that $\mathsf{B}^{\mathcal{F}}(M)$ uses a *finite number* of wires. This number depends on the syntactical structure of the term; precisely we exactly need an extra-wire at the left every time we have a (nested) application. Alternatively we may add an index to the translation $\mathcal{F}$, mimicking the definition of $\mathcal{G}$. This index expresses now the number of extra-wires required to the left of the term (see Figure 10).

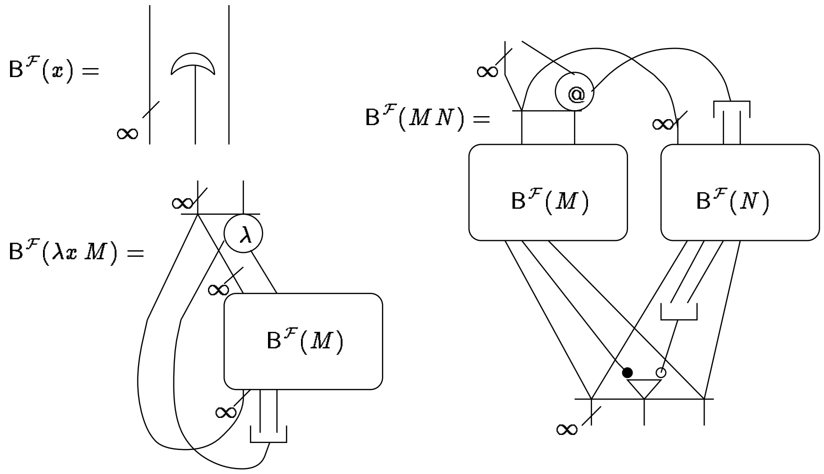


Fig. 6. The bus encoding of the function $\mathcal{F}$

The reader will have probably noted another difference w.r.t. to [GAL92a]: the dimension of the bus at the root of the term is different from its dimension at free variables (in particular, it augments of 1). However, this is merely a consequence of the different encoding of λ-calculus in Linear Logic and we shall not furtherly discuss this point.

It is possible to avoid the infinite (or, in any case, growing) number of wires needed during the translation by introducing suitable operators (square brackets) on the leftmost wire. The resulting translation $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$ is described in Figure 7. Remark that, in $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$, the bus has always dimension two at the root of a (sub) term, and 3 at its free variables. This is actually the way followed in [GAL92a]. The main difference with [GAL92a] is in the rule for the application, since $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$ puts a box in the second argument of the application, whilst in [GAL92b] it was on the rule for abstraction. The purpose of the two brackets added on the leftmost wire is exactly that of "absorbing" and "recreating" the extra-wire required by the box.

The correspondence between $\mathsf{B}^{\mathcal{F}}$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$ is established by Theorem 10 below. Few preliminary properties about $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$ are required. We say that a node is *at level* n if, once fixed the direction for counting wires, the node is located on the n-th wire.

Proposition 1. *Let us fix a right-to-left numbering of wires. In the translation* $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$*, the nodes located on the leftmost wire never change the level of other nodes through interactions.*

This is an immediate consequence of the shape of the rules in Figure 5.

Proposition 2. *Let G be a graph obtained by evaluating λ-terms translated according to* $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$*. Only brackets may appear on the leftmost wires of G. Moreover such nodes will always remain on the leftmost wire till they disappear.*

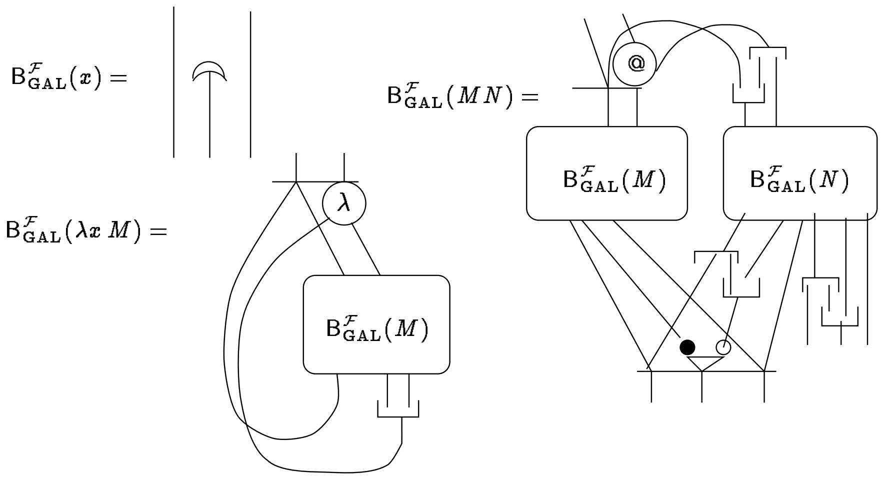


Fig. 7. The translation function $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$

Also this property is immediate because it is true initially, and it is obviously preserved by the reduction rules in Figure 5.

Let a *deadlock* be two nodes, one in front of the other, that do not interact, namely no rule of Figure 5 applies. Remark that Proposition 2 guarantees the absence of deadlocks on the leftmost wires of $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ or graphs obtained by its evaluation. Indeed, the nodes on leftmost wires are brackets and two brackets, one in front of the other, can be simplified by the second rule of Figure 5.

Definition 3. The graphs obtained by evaluating $\mathsf{B}^{\mathcal{F}}(M)$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ are related by the following function $\imath$:

- $\imath : \mathsf{B}^{\mathcal{F}}(M) \to \mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ is defined by structural induction. It is immediate when M is a variable or an abstraction. When $M = M'M''$, let $\imath'$ and $\imath''$ the mappings on the graphs for M' and M''. Let us define $\imath$ for the nodes and the wires which aren't in the domains of $\imath'$ and $\imath''$. The application node, the control nodes at level 0 or 1 and the wires at level 0 of $\mathsf{B}^{\mathcal{F}}(M)$ are mapped into the corresponding nodes of $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$. The wire at level k, $k \geq 1$, of the bus departing from the second argument of the application node of $\mathsf{B}^{\mathcal{F}}(M)$ is mapped onto a path of $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ traversing the bracket at level 1. The wires at level 2 representing free variables of $\mathsf{B}^{\mathcal{F}}(M'')$ are mapped onto the path traversing the bracket and outgoing from the right-branch. Those at level k, $k \geq 3$ are mapped onto the path traversing the bracket and outgoing the left-branch;
- Suppose to have established a correspondence $\imath$ between a graph G obtained by evaluating $\mathsf{B}^{\mathcal{F}}(M)$ and a graph G' resulting from $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ and let $G' \Rightarrow F'$. If this reduction involves a bracket on the leftmost wire, then the same correspondence $\imath$ relates G and F'. Otherwise the redex contracting $G' \Rightarrow F'$ has a counter-image in G (namely it is a redex of the same type). Let $G \Rightarrow F$

be the contraction of such a redex. The mapping $\imath'$ between F and F' is defined by changing $\imath$ according to the redex fired.

Lemma 4. *The correspondence $\imath$ of Definition 3 between the evaluations of $\mathsf{B}^{\mathcal{F}}(M)$ and of $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ is well-defined.*

PROOF: Suppose to have a correspondence $\imath$ between G and G' obtained from $\mathsf{B}^{\mathcal{F}}(M)$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$, respectively, and such that there is a bijection between redexes not involving brackets on the leftmost wires of G'.

Therefore a rewriting not involving the leftmost wire may be performed both in G and G'. A rewriting, let us say r, involving the leftmost wire concerns only G'. But r does not modify $\imath$ for the following reasons:

1. r does not change the level of nodes located on wires different from the leftmost (by Proposition 1);
2. r does not duplicate the nodes on wires different from the leftmost (since, by Proposition 2, there are only brackets on leftmost wires). ■

Remark. We recall that, in sharing graphs, brackets and croissants are used only with the purpose to guarantee the correct matching of fans and abstractions/applications (namely guaranteeing their correct interaction). Observe that Lemma 4 implies that such matchings are performed in the same way both in $\mathsf{B}^{\mathcal{F}}$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}$.

An immediate consequence of the foregoing lemma is stated by the following corollary.

Corollary 5. *The implementations $\mathsf{B}^{\mathcal{F}}(M)$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ always reduce interactions of application-abstraction nodes and between fans which are in corresponding positions of the graphs, where the correspondence is fixed by the function $\imath$ of Definition 3.*

Therefore the interactions of fans and applications/abstractions is completely unsensible to the removal of leftmost brackets in $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$. Corollary 5 fixes a syntactic correspondence between $\mathsf{B}^{\mathcal{F}}(M)$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ which is quite strong, since it ensures that the amount of sharing performed by fans is the same both in $\mathsf{B}^{\mathcal{F}}(M)$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$. Below we show that this correspondence is also semantics, where with "semantics" we mean the interpretation of a sharing graph G, namely the λ-term associated to G. To this aim we recall from [GAL92a] the notion of *consistent path*, which is the basic brick for the semantics of sharing graphs.

Definition 6. Let a *context* be a term generated by the following grammar:

$$a \;::=\; \square \;|\; \circ\cdot a \;|\; \bullet\cdot a \;|\; \langle a,\, b\rangle$$

The semantics of control nodes is given by relations on sequences of contexts as described by the following picture:
The semantics of abstraction and application nodes may be defined in terms of fans, by using the equations in Figure 9.

Let φ be a path into a sharing graph (with the bus notation). We say that φ is *consistent* if

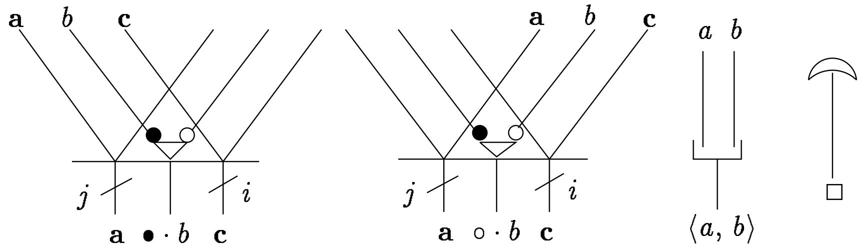


Fig. 8. The semantics of control nodes

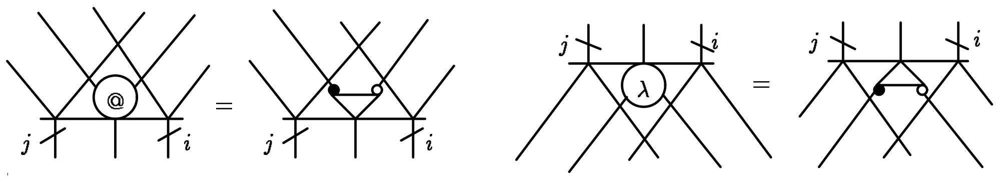


Fig. 9. The interpretation of application and abstraction nodes in terms of fans

1. every (edge of every) wire of φ is labeled by a context;
2. every pair of consecutive wires satisfy the constraints in Figure 8.

Definition 7. Let G be a sharing graph (in the bus notation). Let R_{ij} be the relation on pairs $(\overline{d}, \overline{d'})$, where $\overline{d}$ and $\overline{d'}$ are sequence of contexts, and $(\overline{d}, \overline{d'}) \in R_{ij}$ if and only if there exists a consistent path φ connecting the *root nodes* i and j and such that the context at i is $\overline{d}$ and the context at j is $\overline{d'}$. The *context semantics* of G, denoted $\mathcal{C}(G)$, is the set of relations R_{ij}, where i and j are root nodes.

Theorem 8. *The context semantics of* $\mathsf{B}^{\mathcal{F}}(M)$ *and* $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ *is invariant by sharing graph reductions. Namely, let* G *be a sharing graph obtained by evaluating* $\mathsf{B}^{\mathcal{F}}(M)$ *or* $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$*, if* $G \Rightarrow G'$ *then* $\mathcal{C}(G) = \mathcal{C}(G')$.

PROOF: Because the rewriting rules of Figure 5 do not change the initial and final contexts of consistent paths traversing the redexes. ■

In [GAL92a] the semantics of a sharing graph is defined by reading-back the Böhm-tree of a λ-term. This tree is obtained by taking only the consistent paths root-to-root [4]. The following lemma states that these paths are the same both in $\mathsf{B}^{\mathcal{F}}(M)$ and $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$, up-to some brackets on leftmost wires of paths in $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$, which nevertheless do not affect consistency.

[4] In the more general setting of proof-nets, we may take the *execution formula* as the semantics of a proof net. This is obtained by the consistent paths root-to-root (see [DR93, ADLR94]).

Lemma 9. *For every root-to-root consistent path in $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$, the corresponding path in $\mathsf{B}^{\mathcal{F}}(M)$ – where the "correspondence" is fixed by $\imath$ in Definition 3 – is root-to-root and consistent, and* vice versa.

PROOF: Let φ be a consistent path in $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ and ψ be its counter-image according to the embedding of Lemma 4. Remark that we are considering *every* consistent path, not only those starting and finishing at root-nodes. The case of root-to-root consistent paths follows easily. We assume that, for every section of the bus in $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$, the corresponding section in $\mathsf{B}^{\mathcal{F}}(M)$ has a strictly larger width. This may be easily obtained by taking enough wires "on the left" in the encoding $\mathsf{B}^{\mathcal{F}}(M)$.

We prove, by induction on the length of ψ, that φ is consistent. In particular, if the width of φ at a given section is k and the width of ψ at the corresponding section is n $(n \geq k)$ then

1. the context of the i-th wire, $0 \leq i < k-1$ is the same both in φ and ψ;
2. the context of φ of the $(k-1)$-th wire is $\langle \cdots \langle a_{n-1}, a_{n-2} \rangle \cdots a_{k-1} \rangle$ and the context of the i-th wire of ψ, $k-1 \leq i \leq n-1$, is a_i.

Remark: since we are taking paths φ which are images, according to $\imath$, of paths ψ in $\mathsf{B}^{\mathcal{F}}(M)$, φ never terminates at a bracket on the leftmost wire. The basic case has two subcases: (a) when φ is a bus in between two nodes which aren't brackets on the leftmost wires (and φ does not traverse any other node) and (b) when φ traverses exactly one bracket on the leftmost wire. These two cases are easy and left to the reader.

When $\psi = \psi' \cdot u$, let φ' be the (consistent) path corresponding to ψ' and w be the image of u. If w is a single edge (it does not traverse a bracket on the leftmost wire) u terminates at a node which is not on the leftmost wire, the statement of the induction is easy. Indeed the decision how to lengthen φ' depends on the context at the wire where it is placed the final node of φ' (which cannot be a bracket on the leftmost wire, by definition of $\imath$ – see Definition 3 –).

Otherwise $w = w' \cdot w''$ and let $\mathfrak{b}$ be the bracket on the leftmost wire traversed by w. There are two subcases:

1. w' enters from the principal port of the bracket. Since the width of the bus u is greater than the corresponding width of w lengthening ψ', the context labeling the leftmost edge of w', let us say the k-th, must have the shape $\langle a, b \rangle$. Hence, by hypothesis, b must mark the k-th wire of u and a stores the contexts of u at wires $k+1$, $k+2$, etc. It is clear that $\varphi' \cdot w'$ may be consistently lengthened with w''.
2. w' enters from an auxiliary port of the bracket. This case is an easy consequence of the inductive hypothesis.

The *vice versa* may be proved in the same way. ∎

Theorem 10. *The implementations $\mathsf{B}^{\mathcal{F}}_{\mathrm{GAL}}(M)$ and $\mathsf{B}^{\mathcal{F}}$ are syntactically and semantically equivalent. This equivalence is up-to brackets on leftmost wires, which are inessential.*

PROOF: immediate consequence of Lemma 4 and Lemma 9. ∎

6 The correspondence of $\mathcal{F}$ and $\mathcal{G}$

The correspondence between $\mathcal{F}$ and $\mathcal{G}$ will be proved by reasoning again on the bus-notation. The main idea underlying our proof is: *instead of counting levels from right to left, we shall count them from left to right.* The application of this criterion to the function $\mathsf{B}^{\mathcal{F}}$ gives $\mathsf{B}^{\mathcal{F},r}$, which is described in Figure 10 (we assume $\mathsf{B}^{\mathcal{F},r}(M) = \mathsf{B}_0^{\mathcal{F},r}(M)$).

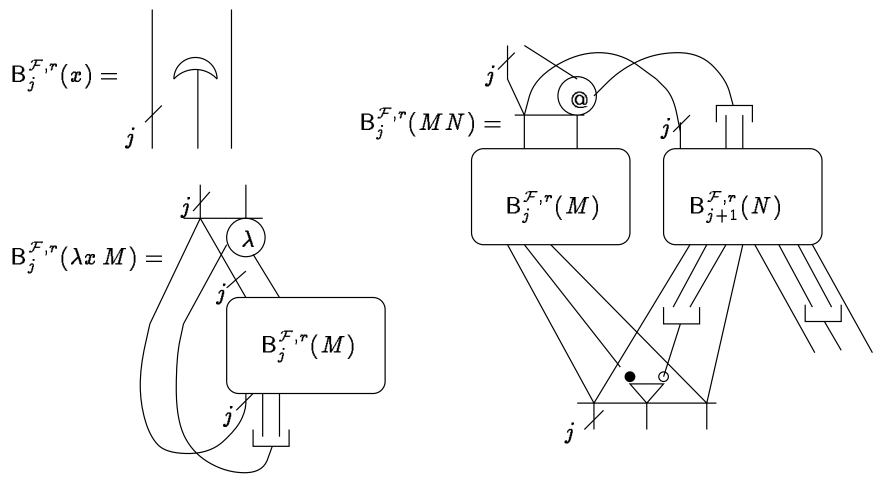


Fig. 10. The translation $\mathsf{B}^{\mathcal{F},r}$

On the other hand, the bus interpretation of $\mathcal{G}$, namely $\mathsf{B}^{\mathcal{G}}$, is illustrated in Figure 11 (as usual $\mathsf{B}^{\mathcal{G}}(M) = \mathsf{B}_0^{\mathcal{G}}(M)$). Observe that, in the definition of $\mathsf{B}^{\mathcal{G}}$, wires are actually counted from the left to the right. We mean that, as far as buses are concerned, the translation $\mathcal{G}$ implicitly uses a dual criterion w.r.t. $\mathcal{F}$ for counting levels.

Remark also that $\mathsf{B}^{\mathcal{F},r}$ differs from $\mathsf{B}^{\mathcal{G}}$ only for the presence of some extra-brackets at the root of the argument of an application and on bound variables. The relation between $\mathsf{B}^{\mathcal{F},r}$ and $\mathsf{B}^{\mathcal{G}}$ may be fixed in a way similar to the correspondence fixed in Section 5. By mimicking Definition 3, it is possible to define a mapping $\jmath$ from the evaluation of $\mathsf{B}^{\mathcal{G}}(M)$ to the evaluation of $\mathsf{B}^{\mathcal{F},r}(M)$ which forgets about the brackets on the rightmost wires of the implementation $\mathsf{B}^{\mathcal{F},r}$ and of their interaction (we leave to the reader the formalization of the details).

Lemma 11. *The above correspondence $\jmath$ between the evaluations of $\mathsf{B}^{\mathcal{G}}(M)$ and of $\mathsf{B}^{\mathcal{F},r}(M)$ is well-defined.*

Establishing the above lemma is less obvious than for Lemma 4, because of the presence of applications and abstractions on rightmost wires (in the situation of Lemma 4, there were only brackets on leftmost wires). In particular, what

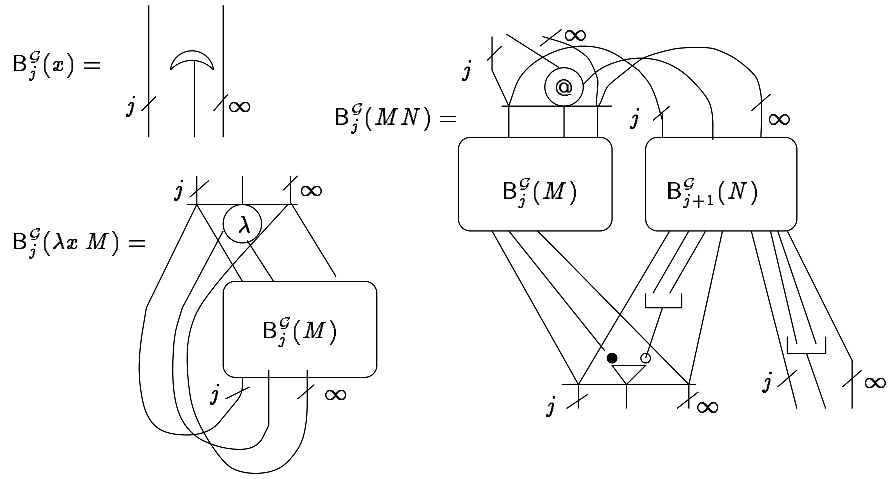


Fig. 11. The bus interpretation of the translation $\mathcal{G}$

could invalidate the lemma is the eventual presence of a deadlock between a bracket and an abstraction (or application) node on the rightmost wire. Such a configuration is forbidden by the following proposition.

Proposition 12. *Never a bracket is in front of an abstraction or an application on the rightmost wires of graphs obtained evaluating* $\mathsf{B}^{\mathcal{F},r}(M)$.

Remark. This property has been already proved for the general setting of Interaction Systems (see Lemma 8.13 in [AL94]).

PROOF: We prove by induction on the length of the derivation $\mathsf{B}^{\mathcal{F},r}(M) \Rightarrow^* G$ that

1. on the rightmost wires there are only abstraction, applications or brackets;
2. there is always a bracket in front of a bound port of an abstraction or the second argument of an application;
3. if on a 0-level wire there is a sequence of brackets, such a sequence may be simplified to a single bracket (by firing instances of the second rule of Figure 5) which is in front of a bound port of an abstraction or the second argument of an application.

Items 1, 2 and 3 are clearly true for $\mathsf{B}^{\mathcal{F},r}(M)$. Let us check the inductive step. That is $\mathsf{B}^{\mathcal{F},r}(M) \Rightarrow^* G' \Rightarrow G$. The interesting case is when $G' \Rightarrow G$ is due to the interaction of an application m and an abstraction n. Then, according to the definition of the rewriting rule, the bus entering on the output port of m is connected with the body of n and the bus entering into the bound port of n is connected with the second argument of m. Let u and w be these new connections, respectively.

Now remark that the bracket which, by induction hypothesis, is in front of the bound port of n interacts with the bracket on the second argument of m, after the reduction m-n. Therefore, as far as the wire 0 of w is concerned, properties 1, 2 and 3 do not change after firing m-n. The properties 1, 2 and 3 follows easily by induction for the wire 0 of u. It is not necessary to check the other wires, since the rewriting $G' \Rightarrow G$ does not modify them.

The statement of the lemma follows immediately from 1, 2 and 3. ∎

Lemma 14 gives a strong syntactic correspondence between the implementations $\mathsf{B}^{\mathcal{G}}$ and $\mathsf{B}^{\mathcal{F},r}$. This relation may be lifted to the semantics, by showing the existence of a bijection between root-to-root consistent paths in $\mathsf{B}^{\mathcal{G}}$ and $\mathsf{B}^{\mathcal{F},r}$.

Lemma 13. *For every root-to-root consistent path in $\mathsf{B}^{\mathcal{F},r}(M)$, the corresponding path in $\mathsf{B}^{\mathcal{G}}(M)$ – where the "correspondence" is fixed by $\jmath$ – is root-to-root and consistent, and* vice versa.

The proof of this lemma is omitted, in order to meet the space requirements. Anyway the proof is similar to the one of Lemma 9, apart some difficulties due to the presence of abstractions and applications on rightmost wires. These problems are accommodated by means of Proposition 12.

Lemma 11 and Lemma 13 imply:

Theorem 14. *The implementations $\mathsf{B}^{\mathcal{G}}$ and $\mathsf{B}^{\mathcal{F},r}$ are syntactically and semantically equivalent. This equivalence is up-to brackets on rightmost wires, which are inessential.*

Remark. The above correspondence and the bus notation definitely clarify that the propagation of a control operator inside a box is the same both for $\mathcal{F}$ and for $\mathcal{G}$. The criterion adopted for counting wires is the unique reason for the operators that are propagating inside a box are at a lower level than the (nodes of) the box in $\mathcal{G}$, whilst it is the contrary in $\mathcal{F}$.

7 Conclusions

In this paper we have shown the correspondence between several different translations of λ-calculus into sharing graphs that have been proposed in the literature. We did it by passing through the so called *bus notation* [GAL92a], and showing that, apart from some syntactical details, the main difference is in the different way of reading levels of wires in the bus: from left-to-right, or from right-to-left.

Another source of difference of the translations is due to different encodings of λ-calculus into Linear Logic, based on different type isomorphisms (this topic looks quite clear, and it has not been really addressed in this paper). Then, the several translations can be essentially classified as follows.

By this work, we get some evidence in favour of Asperti's translation, since it uses the minimal number of operators. However, it seems possible that the additional information provided by the extra-operators of other translations could

isomorphism	right-to-left	bus	left-to-right
$D \cong (!D) \multimap D$	[AL93a]	this paper	[As94]
$D \cong !(D \multimap D)$	[GAL92b]	[GAL92a]	

Fig. 12. Classification of translations

be of some theoretical and even practical use (for instance in trying to solve the well known and crucial problem of accumulation of control operators).

References

[As94] A. Asperti. *Linear Logic, Comonads, and Optimal Reductions.* To appear in Fundamenta Informaticae, Special Issue devoted to Categories in Computer Science. 1994.

[AL93a] A. Asperti, C. Laneve. *Optimal Reductions in Interaction Systems.* Proc. of the 4th Joint Conference on the Theory and Practice of Software Development, TAPSOFT'93, Orsay (France). April 1993.

[AL94] A. Asperti, C. Laneve. *Interaction System 2: The practice of optimal reductions.* May 1993. A revised version of this paper may be got by anonymous ftp at `cma.cma.fr` as file `pub/papers/cosimo/newIS2.ps.Z`.

[ADLR94] A. Asperti, V. Danos, C. Laneve, L. Regnier . *Paths in the λ-calculus: three years of communications without understandings.* Proc. of the 9th Annual Symposium on Logic in Computer Science (LICS 94), Paris 1994.

[DR93] V. Danos, L. Regnier. *Local and asynchronous beta-reduction.* Proc. of the 8th Annual Symposium on Logic in Computer Science (LICS 93), Montreal. 1993.

[Gi86] J. Y. Girard. *Linear Logic.* Theoretical Computer Science, 50. 1986.

[GAL92a] G. Gonthier, M. Abadi, J.J. Lévy. *The geometry of optimal lambda reduction.* Proc. of the 19th Symposium on Principles of Programming Languages (POPL 92). 1992.

[GAL92b] G. Gonthier, M. Abadi, J.J. Lévy. *Linear Logic without boxes.* Proc. of the 7th Annual Symposium on Logic in Computer Science (LICS'92). 1992.

[Lam90] J. Lamping. *An algorithm for optimal lambda calculus reductions.* Proc. of the 17th Symposium on Principles of Programming Languages (POPL 90). San Francisco. 1990.

[Le78] J.J.Levy. *Réductions correctes et optimales dans le lambda-calcul.* Thèse de doctorat d'état, Université de Paris VII. 1978.

[Re92] L. Regnier. *Lambda Calcul et Réseaux.* Thèse de doctorat, Université Paris VII. 1992.