March 1997

A General Theory of Sharing Graphs

Stefano Guerrini
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-97-04.

A General Theory of Sharing Graphs

Abstract
Sharing graphs are the structures introduced by Lamping to implement optimal reductions of lambda calculus. Gonthier's reformulation of Lamping's technique inside Geometry of Interaction, and Asperti and Laneve's work on Interaction Systems have shown that sharing graphs can be used to implement a wide class of calculi. Here, we give a general characterization of sharing graphs independent from the calculus to be implemented. Such a characterization rests on an *algebraic semantics* of sharing graphs exploiting the methods of Geometry of Interaction. By this semantics we can define an unfolding partial order between proper sharing graphs, whose minimal elements are unshared graphs. The *least-shared-instance* of a sharing graph is the unique unshared graph that the unfolding partial order associates to it. The algebraic semantics allows to prove that we can associate a semantical read-back to each unshared graph and that such a *read-back* can be computed via suitable *read-back reductions*. The result is then lifted to sharing graphs proving that any read-back (or unfolding) reduction of them can be simulated on their least-shared- instances. The sharing graphs defined in this way allow to implement in a distributed and local way any calculus with a global reduction rule in the style of the beta rule of lambda calculus. Also in this case the proof technique is to prove that sharing reductions can be simulated on least-shared-instances. The result is proved under the only assumption that the structures of the calculus have a *box nesting property*, that is, that two beta redexes may not partially overlap. As a result, we get a sharing graph machine that seems to be the most natural low-level computational model for functional languages. The paper concludes showing that optimality is a by-product of this technique: optimal reductions are lazy reductions of the *sharing graph machine*. We stress the proof strategy followed in the paper: it is based on an amazing interplay between standard rewriting system properties (strong normalization, confluence, and unique normal form) and algebraic properties definable via the techniques of Geometry of Interaction.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-97-04.

A General Theory of Sharing Graphs

Stefano Guerrini

University of Pennsylvania
3401 Walnut Street, Suite 400A
Philadelphia, PA 19104-6228

March 1997

Site of the NSF Science and Technology Center for
Research in Cognitive Science

IRCS Report 97--04

A general theory of sharing graphs

Stefano Guerrini

University of Pennsylvania, IRCS
3401 Walnut Street, Suite 400A - Philadelphia, PA 19104-6228
email: `stefanog@saul.cis.upenn.edu`

Sharing graphs are the structures introduced by Lamping to implement optimal reductions of lambda calculus. Gonthier's reformulation of Lamping's technique inside Geometry of Interaction, and Asperti and Laneve's work on Interaction Systems have shown that sharing graphs can be used to implement a wide class of calculi. Here, we give a general characterization of sharing graphs independent from the calculus to be implemented. Such a characterization rests on an *algebraic semantics* of sharing graphs exploiting the methods of Geometry of Interaction. By this semantics we can define an unfolding partial order between proper sharing graphs, whose minimal elements are unshared graphs. The *least-shared-instance* of a sharing graph is the unique unshared graph that the unfolding partial order associates to it. The algebraic semantics allows to prove that we can associate a semantical *read-back* to each unshared graph and that such a read-back can be computed via suitable *read-back reductions*. The result is then lifted to sharing graphs proving that any read-back (or unfolding) reduction of them can be simulated on their least-shared-instances. The sharing graphs defined in this way allow to implement in a distributed and local way any calculus with a global reduction rule in the style of the beta rule of lambda calculus. Also in this case the proof technique is to prove that sharing reductions can be simulated on least-shared-instances. The result is proved under the only assumption that the structures of the calculus have a *box nesting property*, that is, that two beta redexes may not partially overlap. As a result, we get a *sharing graph machine* that seems to be the most natural low-level computational model for functional languages. The paper concludes showing that optimality is a by-product of this technique: optimal reductions are lazy reductions of the sharing graph machine. We stress the proof strategy followed in the paper: it is based on an amazing interplay between standard rewriting system properties (strong normalization, confluence, and unique normal form) and algebraic properties definable via the techniques of Geometry of Interaction.

28 February 1997

1 Introduction

Techniques to implement functional calculi based on the use of pointers (and then on graphs) were known from long time (see [Wad71,PJ86]). In such solutions sharing of graphs was implemented by pointers connecting a shared subterm (the tree spanned by the paths rooted at a given node) to the nodes accessing it. The drawback of a naive implementation of such techniques was the possibility of unwanted side-effects during the execution of variable substitutions (see [Wad71]). In fact, to replace a term T_s for the occurrences of a variable x by substituting a pointer to T_s for any pointer to x would apply such a substitution in all the shared terms T_t containing x. To fix up such a problem of soundness, the usual solution was to create a new instance of T_t in which to safely replace T_s for x. Soundness were ensured but at the cost of the duplication of any redex in T_t. Therefore, any implementation of a functional calculus in which sharing be kept at the level of terms cannot avoid duplication of redexes.

1.1 A fine decomposition of the λ-calculus β rule

The basic point of sharing graphs is to implement a sharing finer than the one obtainable keeping multiple pointers to the same subterm. To have an idea of how to achieve this, let us see how the β rule of λ-calculus could be implemented via a step-by-step graph reduction system in which the objects rewritten are nodes rather than whole terms.

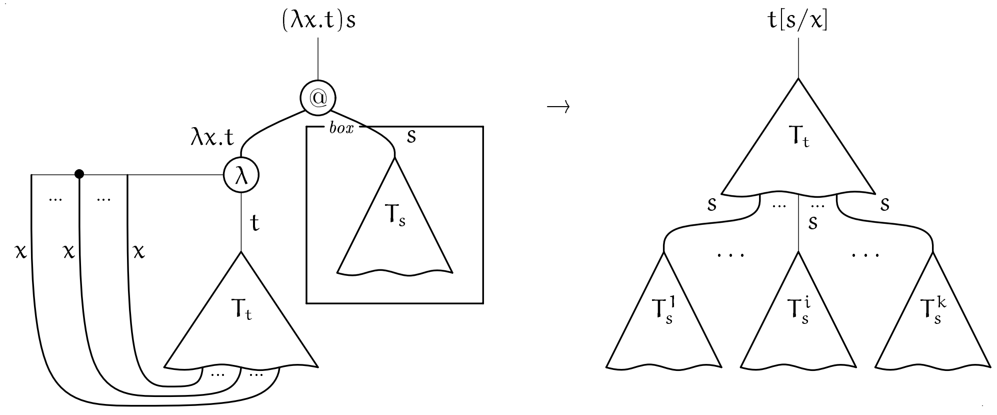

Fig. 1. λ-calculus β rule

Let us represent a λ-term by its abstract syntax tree and, to avoid any problem with the names of the variables, let us assume that all the occurrences of a variable x in a λ-term t are merged into a node (the bullet in Figure 1) connected to the λ abstraction $\lambda x.t$. According to our goal, the natural substitute for the β rule of Figure 1 is the sharing β rule of Figure 2.

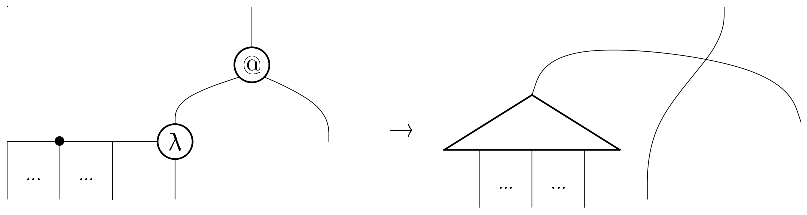

Fig. 2. λ-calculus sharing β rule

In the sharing β rule, a new node, say a multiplexer or *mux* for short, is inserted to connect the occurrences of x to the root of T_s. The second part of this decomposition is then the definition of a set of rules performing a step-by-step duplication of the term T_s, say a local and distributed duplication of T_s. Such rules are given in Figure 3 (for the sake of a clear design, all the muxes in that figure are binary; the generalization is straightforward). The meaning of most of them is self-evident, the only ones that deserve some thoughts are λ-up and absorption.

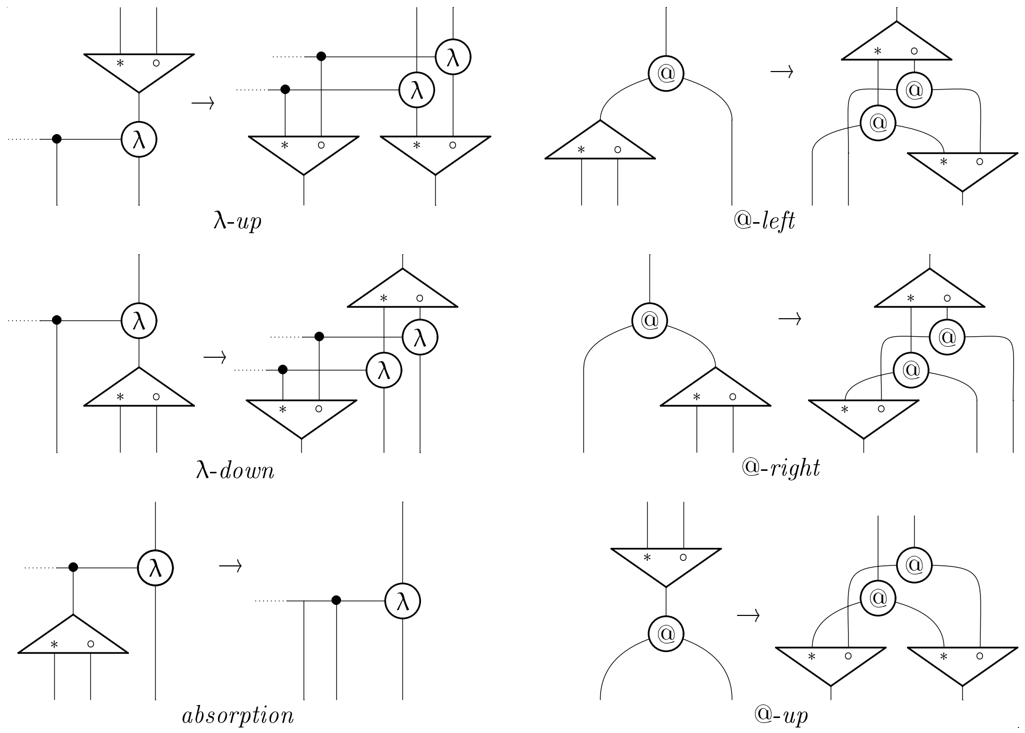

Fig. 3. Duplication rules

In λ-up, the mux not only duplicates the λ node that it points through its principal port (the vertex of the triangle), it also splits the occurrences of the corresponding bound variable (in Figure 3 we have drawn just one of such occurrences, but an analog duplication apply simultaneously to all the other occurrences). In practice, duplicating a λ node we also have to create a new instance of the variable that the node bound for each new instance of λ.

The λ-up duplication rule is the first divergence from standard graph reduction techniques, for after the execution of a λ-up we cannot say any more that the subgraph pointed by a mux is a shared subterm. In fact, the node inserted to split the occurrences of the variable is not a sharing or multiplexing operator, it is instead a sort of *demultiplexing* node. (Since in most of the graphs we will consider the orientation is not so clear as in the case of λ-calculus, instead to distinguish multiplexers into muxes and demuxes, in the body of the paper we will rather prefer a more neutral classification of multiplexers into positive and negative muxes.) In some sense, each occurrence of the variable has been replaced by a hole which may be filled in two different ways, according to which instance of the shared subterm we are considering: the one relative to the left instance of the duplicated λ if we assume to access the shared part from the left port of the mux, or the one relative to the right instance in the other case. Hence, each demux inserted to split the variable matches with the mux below the new instances of the λ node, where by matching we mean that the edges entering through the back port of a mux, say through its auxiliary ports, are distinct, and that the variable occurrence connected to a given auxiliary port is bound by the λ node at the matching auxiliary port of the matching mux. For instance, in each rule of Figure 3, the auxiliary ports of each mux are marked by a symbol, using a distinct marker for each pair of matching ports.

Furthermore, demuxes too can duplicate nodes, as shown by the @-left rule for instance. As a consequence the duplication of $\mathsf{T_s}$ is no more strictly top-down, for demuxes move bottom-up.

The absorption rule applies when a mux reaches a free variable of $\mathsf{T_s}$ (let us assume w.l.o.g. that the we restrict to closed λ-terms, so any variable free in $\mathsf{T_s}$ is bound by some λ external to $\mathsf{T_s}$). The main remark about this rule is that it cannot be safely applied in any configuration as the one drawn in the picture. In fact, it is easy to build a term $\mathsf{T_s}$ in which a mux reaches a variable bound by a λ internal to $\mathsf{T_s}$. Some thoughts allow to realize that in such a case to apply the absorption would not be sound. The mux of the rule would definitely have a matching mux above the λ, then in the place of the absorption it would be sound to let the mux duplicate the λ (to simplify the presentation, such a kind of λ rule has been dropped in Figure 3) as in the case of λ-down or λ-up.

The latter is a first bell ring warning us that we need some local information taking into account the global position of a node. In this case, such an information would help has to recognize when the λ is outside $\mathsf{T_s}$ and the absorption rule might be safely applied.

So far, absorption is the only rule that erases muxes. It would suffice to eliminate all the muxes and to complete the distributed duplication of $\mathsf{T_s}$, only if this process could be performed moving top-down only. But the duplication rules work independently from such an orientation. Apart for the free variables of T_s, after a few duplication steps, the residual shared part of T_s is a (possibly disjoint) graph delimited by a set of matching (de)muxes. The duplication may then continue by a random choice of any of such muxes, or in a concurrent setting, by the simultaneous firing of several duplication rules. Hence, proceeding with the reduction, a mux may stop duplicating nodes in two ways: reaching the binding port of a λ node external to T_s; reaching the principal port of a demux. In the first case the absorption rule applies. In the second case, the two facing muxes correspond to a boundary around an empty shared subgraph. As a consequence it is sound to remove (annihilate) them directly connecting each port of the mux to the matching port of the demux.

We are done. The duplication rules plus the latter annihilation of muxes allow to perform a complete duplication of T_s. The β rule of Figure 1 can then be split into the execution of a sharing β rule, followed by the application (in any order) of a sequence of duplication or annihilation rules.

What we have get so far is however not yet particularly appealing. In fact, what we achieved is just a detailed implementation of β; but there is up to now no relevant difference between an implementation based on the duplication rules and one obtained by directly coding the duplication of T_s in some programming language external to the graph rewriting system. Nevertheless, all the rules we have met, sharing β included, are local and can be fired independently from the shape of the graph. Hence, what does it happen if we freely execute them? Namely, what does it happen if we execute another sharing β rule, either internal or external to T_s, before that the duplication rules have accomplished their task of to completely duplicate T_s?

The first consequence is that the muxes around the graph are no more part of the same duplication process. Therefore, when two of them face it is no more sure that they are matching and that they can annihilate. Some thoughts allow to realize that, when a pair of non matching muxes faces, the correct rewriting is rather the swap rule of Figure 4.

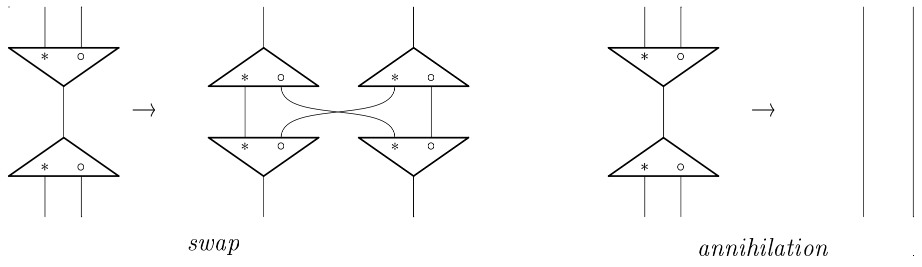

Fig. 4. Mux interactions

This is the second and more important bell ring. It warns us that we need a way to talk about the dependencies between the nodes of a graph due to their position in it, or to their origin. In fact, the rewriting rules we have got contain at least two kinds of rule that deserve to know such an information: the mux interactions of Figure 4 and the absorption rule. In the mux interactions, such an information should enable us to recognize when two facing muxes have to be considered matching—and the annihilation should then be applied—and when such muxes are not matching—and the swap rule should be applied instead.

The first attempt would be to mark each mux with a label corresponding to the redex that created it. Unfortunately, Lamping (who firstly introduced sharing graphs, see [Lam90]) showed that this solution does not work, as in certain cases, even two muxes that have been created by the same redex have to be considered non matching. The right solution is to have a dynamical labeling of muxes, say an index, ensuring that the annihilation rule applies when the muxes have the same index, and that the swap rule applies otherwise. This introduces the next relevant ingredient for the understanding of sharing graphs, the so-called boxes.

1.2 Boxes

To generalize our analysis, let us try to identify the elements characterizing the β rule. The rule take a subgraph and after its duplication connects its new instances to some edges of the reducing graph. In the λ-calculus, the subgraphs that can be involved in this process are clearly recognizable: they are the subtrees rooted at the right edge of an @ node. More in general, when the graphs under analysis are not trees, we might rather have a frame, say a *box*, surrounding the subgraphs which might be argument of a β-like reduction rule. The β rule could then be depicted as a rewriting in which a box is duplicated and displaced somewhere in the graph. By the way, this point of view is sound, as far as boxes can be properly rearranged after the execution of a β rule.

A natural way to ensure this is asking that the only ways in which boxes overlap are the trivial ones, that is, to ask that two boxes are either disjoint or enclosed one into the other. In the λ-calculus this is definitely true, as it is readily seen that, replacing a term for a variable occurrence, the box of the replacing term is enclosed by the boxes that previously surrounded the replaced variable occurrence. More generally, we may say that the boxes of a graph must properly nest, say that the graph must have a *box nesting property*, and that the β rule must preserve such a box nesting property.

In the case of λ-calculus, the box nesting property is equivalent to say that λ-calculus is an orthogonal rewriting system. Therefore, to require that the graph rewriting systems we want to study have the box nesting property means to require that an analog orthogonality property hold for them.

The box nesting property gives as the tools to avoid the introduction of a global constructor for the representation of box frames. In fact, assuming that each node/edge is labeled by the number of boxes surrounding it, say the *box nesting level* of the node/edge, we see that each node with two edges whose levels are respectively n and $n+k$ (with $k>0$) is the border of k boxes, and that each of them can be recovered finding the maximal connected subgraph containing the nodes whose edges are at a level greater or equal than $n+i$, for $i=1,2,\ldots,k$.

These simple thoughts just give us a solution to the troubles with the application of the absorption rule. In fact, it is readily seen that a λ node external to T_s may bound a variable internal to T_s only if its level is lower than the level of the root of T_s. Furthermore, any node internal to T_s has a level greater or equal than the root of T_s. Hence, let us assume to label the mux inserted by the sharing β rule with an index equal to the level n of the $\lambda/@$ pair reduced by the rule. Since a box frames T_s (see Figure 1), its root is at level $n+1$. The index of the mux can then be seen as a sort of discriminant, say a *threshold*, that is capable to recognize when to stop the duplication. Namely, a mux duplicates all the nodes at a level greater or equal than its threshold m; if during this process the mux reaches a λ node with a level $n \leq m$, the absorption rule applies and the mux is absorbed by the λ node.

The use of levels opens however another problem: how to correctly reassign a level to each node/edge of a new instance of T_s. According to the shape of the boxes of a λ-term, the difference k between the level $n+k$ of the occurrence of a variable and the level n of the λ node binding it is the number of boxes in which the corresponding instance of T_s must be enclosed. Since firing the β rule the box around T_s disappears, it means that each node/edge of such a new instance of T_s must be increased by $k-1$. It is worth to note that since k might be equal to 0—in the case that the variable occurrence does not close any box—a β rule might also imply the decreasing by 1 of all the levels in T_s.

It is not difficult to see how to perform such a reindexing during the duplication of T_s. Each auxiliary port of a mux should bring an offset by which to increase the level of the instance of the node duplicated by it.

Our duplication rules of Figure 3 should then be completed by inserting thresholds and levels. We omit to redraw them at this point, for we will detailedly discuss such a rewriting system (see Figure 20) in section 8. We only remark that the idea that a mux with threshold m_1 lifts the levels above m_1 apply also in the swap rule. In fact, assume that the thresholds of the two muxes are m_1, m_2 and that $m_1 < m_2$. Each new instance of the mux with threshold m_2 is lifted by the offset q of the corresponding port of the mux with threshold m_1 (see Figure 20).

The soundness of the approach based on levels is easily provable for the case in which a sharing β rule is followed by a whole duplication of T_s. In this case muxes never change their thresholds and the only case of mux interaction that can arise is the annihilation rule. When sharing β and duplications are interleaved (hereafter, when not otherwise specified, we will include mux interactions into duplications), things become instead greatly involved. Muxes thresholds can arbitrarily change and it is not difficult to build a case in which two muxes that was created by the same redex meet with different thresholds.

The core of the paper is devoted to prove soundness in the general case of a rewriting system whose graphs have the box nesting property, and whose rewriting rule is the analog of the β rule of λ-calculus. We will prove that for any *sharing graph*—intuitively a graph obtained by applying a sequence of sharing β and duplication rules—there is a corresponding graph without muxes, write $\mathsf{G} \mapsto \mathscr{R}(\mathsf{G})$ and say $\mathscr{R}(\mathsf{G})$ is the *read-back* of G, s.t. sharing β and duplication rules are sound w.r.t. such a read-back. Namely, we will prove that the following diagrams commute:

$$\begin{tikzcd}
\mathsf{G} \arrow[r, "\beta_s"'] \arrow[d, mapsto] & \mathsf{G}' \arrow[d, mapsto] \\
\mathscr{R}(\mathsf{G}) \arrow[r, "+", "\beta"'] & \mathscr{R}(\mathsf{G}')
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathsf{G} \arrow[r, "\pi"'] \arrow[d, mapsto] & \mathsf{G}' \arrow[d, mapsto] \\
\mathscr{R}(\mathsf{G}) \arrow[r, phantom, "="] & \mathscr{R}(\mathsf{G}')
\end{tikzcd}
\tag{1}$$

where β_s denotes sharing β, and π denotes duplication rule.

According to the diagram on the left, we see that each β_s correspond to a (finite) non empty sequence of β reductions: the execution of a β inside a shared part implies the simultaneous reduction of all the instances of that redex.

The diagram on the right-hand side expresses instead that the read-back of a sharing graph is invariant under π. Furthermore, strong normalization and confluence—that are easily provable in the case β_s followed by a maximal sequence of π—still hold in the general case. Hence, the syntactical read-back we could define via π, *i.e.*, interpreting a sharing graph G as a shared representation of its π normal form $\mathsf{NF}_\pi(\mathsf{G})$, coincides with the semantical one:

$$\begin{tikzcd} & \mathsf{NF}_\pi(\mathsf{G}) \arrow[dd, phantom, "\parallel"] \\ \mathsf{G} \arrow[ur, "*", "\pi"'] \arrow[dr, mapsto] & \\ & \mathscr{R}(\mathsf{G}) \end{tikzcd} \qquad (2)$$

The proof of the previous results requires a careful study of the unfoldings of a sharing graph. In fact, the definition of the map $\mathscr{R}$ splits in two steps: firstly, the individuation of a proper unfolding U of G, write $\mathsf{U} \mathrel{-\!\!\!\prec\!\!\!\prec} \mathsf{G}$, in which all the muxes are unary; then, the definition of $\mathscr{R}$ for such a graph U. Assuming the uniqueness of the unfolding U, say the *least-shared-instance* of G, the read-back can then be defined in the general case taking $\mathscr{R}(\mathsf{G}) = \mathscr{R}(\mathsf{U})$.

Intuitively, the least-shared-instance U of G is a representation of $\mathscr{R}(\mathsf{G})$ in which the reindexing of the nodes has not yet been accomplished. At the same time the unary muxes of U mark the border of a sharable subgraph, for the image in G of each of them is a mux. Nevertheless, to associate the correct graph $\mathscr{R}(\mathsf{U})$ to the unshared graph U (a graph is *unshared* when all its muxes are unary) is not an easy task. Apart for levels, $\mathscr{R}(\mathsf{U})$ is obtained from U just replacing its unary muxes by direct connections. But to reassign levels is not direct in the general case, despite its simplicity for λ-calculus graphs, in which the tree shape forces the existence of a unique level assignment.

Also the definition of least-shared-instance is not direct, as we cannot simply take an unshared graph for which there is a graph morphism $\mathsf{M} : \mathsf{U} \to \mathsf{G}$. Because of the matching of muxes, not all the paths are admissible unfolding G, and thus not all the U definable via graph morphisms are correct.

Most of the paper is devoted to define the correct unfolding partial order $\mathrel{-\!\!\!\prec\!\!\!\prec}$, by which to obtain the least-shared-instance of a sharing graph G as the minimal element s.t. $\mathsf{U} \mathrel{-\!\!\!\prec\!\!\!\prec} \mathsf{G}$. In order to do this we have to develop an algebraic semantics of sharing graphs by which to characterize the shape of a proper unfolding morphism $\mathsf{M} : \mathsf{U} \mathrel{-\!\!\!\prec\!\!\!\prec} \mathsf{G}$. In this way, we will get an algebraic characterization of the proper unshared graphs for which there is a shared representation, and at the same time this will give us a characterization of the (proper) sharing graphs as the ones for which $\mathsf{U} \mathrel{-\!\!\!\prec\!\!\!\prec} \mathsf{G}$ is defined.

The read-back of proper unshared graphs is an immediate consequence of their algebraic semantics. Furthermore, such a semantics will also allow to prove that diagram (1) and diagram (2) hold when G and G' are proper unshared structures, provided that β_s is replaced by an unshared version of the β rule, say β_u rule, positioned midway between β and β_s—in a β_u rule the duplication of a box is still executed globally, the reindexing is instead demanded to some unary muxes inserted at the principal doors of the duplicated boxes (the principal door of a box is the generalization of the root of T_s). Such a decomposition of β reflects in a decomposition of the proof of soundness. In fact, given a sharing graph G, we will prove that any β_s or π reduction $\mathsf{G} \to \mathsf{G}'$ can be simulated on its least-shared-instance U obtaining as result the least-shared-instance of G'. Therefore, diagram (1) is just diagram chasing of the previous results:

$$\begin{tikzcd}
\mathsf{G} \arrow[r, "\beta_s"'] \arrow[d, phantom, "\Yup"] & \mathsf{G}' \arrow[d, phantom, "\Yup"] & & \mathsf{G} \arrow[r, "\pi"'] \arrow[d, phantom, "\Yup"] & \mathsf{G}' \arrow[d, phantom, "\Yup"] \\
\mathsf{U} \arrow[r, "+", "\beta_u"'] \arrow[d, mapsto] & \mathsf{U}' \arrow[d, mapsto] & & \mathsf{U} \arrow[r, "+", "\pi"'] \arrow[d, mapsto] & \mathsf{U}' \arrow[d, mapsto] \\
\mathscr{R}(\mathsf{G}) \arrow[r, "+", "\beta"'] & \mathscr{R}(\mathsf{G}') & & \mathscr{R}(\mathsf{G}) \arrow[r, phantom, "="] & \mathscr{R}(\mathsf{G}')
\end{tikzcd} \tag{3}$$

Strong normalization, confluence, uniqueness of the π normal form, and diagram (2) can then be lifted to sharing graphs exploiting the upper part of diagram (3).

All these results will be proved under the only assumption that the graph rewriting system has the box nesting property. Furthermore, we will see that the algebraic semantics is indeed an abstract characterization of that property, since we will see that the proper sharing graphs defined by it are the ones which π normalize to a graph without muxes for which the box nesting property holds.

1.4 Optimality and other related works

Sharing graphs were introduced by Lamping [Lam90] to implement Lévy's optimal reductions of λ-terms [Lév80]. Several refinements of them where successively proposed by Gonthier et al. [GAL92a,GAL92b], and by Asperti and Laneve [AL94,Asp95]. The work of Gonthier et al. addressed how Lamping's formalism could be interpreted inside the so-called Geometry of Interaction (GOI) of Girard [Gir89]; Asperti presented a more categorical justification of the technique; Asperti and Laneve gave a generalization of the methodology to the so called Interaction Systems, the subclass of the Combinatory Reduction Systems for which it is possible to find a Curry-Howard analogy with a suitable intuitionistic logic. Furthermore, Asperti used sharing graphs to implement an optimal version of an ML-like functional language [AGN95].

The main concern of all these studies was the implementation of optimal reductions. Hence, the set of rules that they proposed was the minimal one useful for such a result. Here, we revert instead the point of view. The main concern is to get a distributed and local implementation of β-like rules.

The general framework in which our results will be achieved will not allow to connect our sharing graphs to optimality in the usual way based on redex families or labels. Nevertheless, in the case in which this makes sense, optimal implementations are obtained just assuming that the duplication rules are applied following a lazy reduction strategy.

Let us be more precise showing how this applies in the example developed so far. To execute the rules of Figure 3 following a lazy strategy means that a mux duplicates a node only when the presence of the mux might hide a β redex. For instance, a mux whose principal port is connected to the left port of an @ node might hide a redex, for one of its auxiliary port might be (or become) connected to a λ. According to this, the only two duplication rules unavoidable in such a lazy strategy are @-left and λ-up. Further, dropping the other rules, not only has no impact on the computational power, it also improves efficiency, since in this way no β redex is unnecessarily duplicated. For instance, let us assume that there be a mux above the @ node of a β redex. The application of an @-up should be followed by a λ-up. But this would create an instance of the redex for each auxiliary port of the mux.

According to the interpretation of muxes in terms of fans and brackets (see section 4.1), such a lazy system is exactly the optimal one.

One of the main consequences of the previously described change of perspective is that the proof technique is completely different from the usual ones based on a direct interpretation of sharing graphs into GOI. For instance, the proof technique of Gonthier et al. rests on the fact that the optimal rules define an interaction system (for the definition of interaction system see [Laf90]) and that this kind of rules are sound w.r.t. the paths definable using GOI. In fact, the paths of GOI give a way to extract the normal form of a proof net without reducing it or, in the case of λ-calculus where a term might not have a normal form, to extract its weak head-normal-form. Hence, any rule sound w.r.t. this set of paths can be safely applied, as it does not change the denotation of the graph. It might seem that such an approach make use of the weakest conditions necessary for a proof of soundness, and then we would expect that it would be able to prove the widest set of sound rules. On the contrary, apart for @-left and λ-up, the duplication rules are hardly provable following it. The point is that it does not exploit the knowledge about the shape of a sharing graph that we can infer knowing that it is the result of a sharing reduction.

Besides, even our algebraic semantics characterizes proper sharing graphs without assuming that they are the result of a reduction. Nevertheless, it exploits the methods of GOI to find the paths that define the read-back of a sharing graph (This point is not explicit in the paper. It is indeed implicit in the definition of least-shared-instance we will give. For the case of λ-calculus, a more direct presentation based on paths can be found in [Gue97].) The proper sharing graphs defined in this way contain the ones obtainable as a result of a sharing reduction and maybe it could be also proved that the two classes coincide.

The result of such an approach is the integration between some algebraic techniques in the style of GOI and a more traditional proof technique exploiting properties of the duplication rules as confluence and strong normalization. We believe that such an integration is one of the principal novelty of the paper.

Another relevant point of the solution we propose is that it represents a sort of abstract *sharing graph machine*. Differing from optimal implementations, in such a sharing graph machine we do not need any external machinery to read-back the result of a computation. The duplication rules internalize the read-back into the system, and at the same time, a lazy application of them allows to compute via optimal reductions.

Finally, the point of view of the paper also has a good proof theoretic motivation. It fits into a well-known approach in which dependencies between the formulas of a proof (net) are represented by means of indexes. For a detailed discussion of these connections we refer the reader to [GMM97a].

1.5 Overview of the paper

The body of the paper starts with a formal definition of the structures we will study (section 2). The main difference w.r.t. what done so far is that we will present sharing graphs as hypergraphs. Hence, what was a node up to now will become a hyperedge, say a link, and what was an edge will become a vertex. Furthermore, we will introduce two special kinds of link: box door links, to delimit the border of boxes; contraction links to merge vertices. We will define levels of links and vertices, and we will define boxes in terms of levels.

In section 3 we will show the relevant calculi fitting in our structures of links.

The set of links used to build sharing graphs will be completed in section 4, after the introduction of muxes. There, we will also give the equivalence between muxes and Gonthier's fan and brackets.

In section 5 we will define sharing morphisms, the basic tool by which to unfold a sharing graph. We will also address why the unfolding problem is so difficult and requires the introduction of the algebraic semantics that we will give in section 7.

The following part, starting from section 7, is the technical core of the paper. A summary of the results achieved in it is contained in section 6.

2 Leveled structures

We take the point of view that nets (the graphs of the calculi to be implemented) are hypergraphs. A leveled net is then a set of named hyperarcs, the so-called *links*, connected by vertices that we will call *arrows*. To both, links and arrows, we associate a *level*—despite we will later see that some levels are indeed redundant. The name of a link, say its *type*, fixes its cardinality and gives the constraints to which the link level and the levels of its incident arrows must accord.

2.1 Hypergraphs

Let $V = \{v_1, v_2, \dots\}$ be a denumerable set of primitive objects called *vertices*. A directed hyperedge, or *hyperarc*, is an ordered pair $e = (e_t, e_h)$ of (possibly empty) disjoint sequences of vertices. The first element (e_t) of the pair is the *tail* of e, the second one (e_h) the *head* of e. Since we will usually draw hyperarcs according to the top-down orientation—the tail of a hyperarc above its head—we will also say that any vertex v_t in the tail of a hyperarc e is above it; that any vertex v_h in the head of e is below it; and conversely, that e is above v_h and that e is below v_t.

A *hypergraph* is a pair $G = (V, E)$, where V is a denumerable set of vertices and E is a denumerable set of hyperarcs whose vertices range over V.

A vertex v of the hypergraph G is an *arrow* if there is at least a hyperarc above or below it, and there is no pair of links e' and e'' s.t. v is either below both e' and e'', or above both e' and e''. In other words, assuming to say that $\varnothing$ is below (above) v when there is no hyperarc below (above) v, an arrow is determined by a pair $(e_a, e_b) \in (E \uplus \{\varnothing\})^2 \setminus \{(\varnothing, \varnothing)\}$ s.t. e_a is above v and e_b is below v. When $\varnothing \in \{e_a, e_b\}$ the arrow v is called a *root arrow*. In particular, when $e_a = \varnothing$, the root v is said a *source arrow*; otherwise, when $e_b = \varnothing$, it is said a *target arrow*.

Remark 1 Let G be a hypergraph whose vertices are arrows. Its dual G^* is a directed graph, as all its edges—the arrows of G—have only one source and one target node. This explains why we call arrows the vertices of our hypergraphs and why, in drawing G, a vertex will be represented by an "arrow" from the link above to the link below it.

An (undirected) path of G is an alternated sequence $\phi = v_0 e_1 v_1 \dots e_k v_k$ of vertices v_i and hyperarcs e_i, s.t.: *(i)* $e_i \neq e_{i+1}$; *(ii)* v_{i-1} and v_i are distinct doors of e_i. According to this definition, a sequence ϕ is a path of G iff it is an undirected path (a sequence of edges) of G^*. Therefore, we will say that G is connected and acyclic when G^* is.

2.2 Structures of links

Definition 2 (link) *A* link *is a hyperarc* e *with:*

(i) *a name, the* type *of* e*;*
(ii) *a set of named* input ports*, one for each arrow above* e*;*
(iii) *a set of named* output ports*, one for each arrow below* e*.*

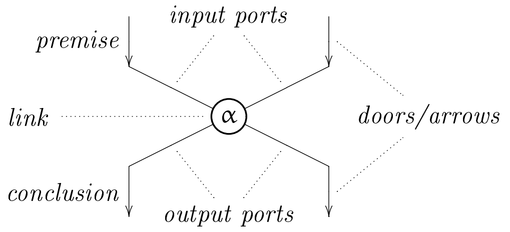

Fig. 5. A link of type α with two input ports and two output ports.

The arrows above a link e are the *premises* of e; the arrows below e are the *conclusions* of e; both premises and conclusions are the *doors* of e. (See Figure 5 for the graphical representation of links.) A link with no premises is a *source link*. A link with no conclusions is a *target link*. Isolated links with no doors are not allowed.

Definition 3 (structure of links) *A* structure *(of links)* G *over the signature* Σ *(a set of link names) is a hypergraph in which:*

(i) *all the vertices of* G *are arrows;*
(ii) *there is at least a (source or target) root arrow;*
(iii) *the names of the links range over* Σ*.*

The source arrows of a structure G are the *premises* of G; the target arrows are the *conclusions* of G. Since a structure does not contain isolated vertices it is immediate that no vertex can be both premise and conclusion of G. Premises and conclusions are the *doors* of G (and $\mathsf{doors}(G)$ denotes their set). The links above a conclusion of G or below a premise are the *door links* of G (and $\mathsf{dlinks}(G)$ denotes their set).

2.2.1 Substructures

Given a structure G over Σ, we denote by $\mathsf{lnk}(\mathsf{G})$ the set of its links and by $\mathsf{vtx}(\mathsf{G})$ the set of its arrows. Furthermore, if $\alpha \in \Sigma$, we denote by $\mathsf{lnk}_\alpha(\mathsf{G})$ the set of the links of G of type α.

Let G be a structure. A substructure R of G, or G-*substructure*, is a structure s.t. $\mathsf{vtx}(\mathsf{R}) \subseteq \mathsf{vtx}(\mathsf{G})$ and $\mathsf{lnk}(\mathsf{R}) \subseteq \mathsf{lnk}(\mathsf{G})$. Since a structure does not contain isolated vertices, it is readily seen that the G-substructures coincide with the parts of $\mathsf{lnk}(\mathsf{G})$. In fact, not only any G-substructure R is uniquely determined by the set $\mathsf{lnk}(\mathsf{S}) \subseteq \mathsf{lnk}(\mathsf{G})$ but, given a set of links $\mathsf{E} \subseteq \mathsf{lnk}(\mathsf{G})$, there exists a (unique) G-substructure s.t. $\mathsf{lnk}(\mathsf{R}) = \mathsf{E}$. The standard inclusion relation and set operations apply to G-substructures according to their interpretation as set of links. Further, $\mathscr{P}(\mathsf{G})$ will denote the set of the G-substructures.

2.3 Boxes

To implement boxes we use two reserved link types called *box door links*: the ! (of-course) or *principal door link*; the ? (why-not) or *auxiliary door link*. Both these links have two doors: the *external* and the *internal* door (denoted by $\mathsf{edoor}(e)$ and $\mathsf{idoor}(e)$, respectively). For the reader acquainted with linear logic proof-nets, the reasons of the names ? and ! are self-evident, even if at this level of abstraction we could have chosen any other pair of names.

Remark 4 According to Figure 6, we will generally assume that the external door of a box door link is a conclusion of the link and that the internal one is a premise. Such an orientation is however not mandatory, for instance, in the case of the λ-calculus we will swap the arrows of the ! links in order to preserve the natural orientation of λ-terms.

Definition 5 (box) *Let* G *be a structure of link. A* box *of* G *is a connected* G-*substructure* B *s.t.:*

(i) *there is a unique arrow* $\mathsf{pdoor}(\mathsf{B}) \in \mathsf{doors}(\mathsf{B})$*—the* principal door *of* B*—which is external door of an ! link* $\mathsf{pdlink}(\mathsf{B})$*;*

(ii) *all the arrows* $\mathsf{adoors}(\mathsf{B}) = \mathsf{doors}(\mathsf{B}) \setminus \{\mathsf{pdoor}(\mathsf{B})\}$*—the* auxiliary doors *of* B*—are external doors of ? links.*

According to the previous definition, a box contains its door links. This choice is just a matter of taste. For instance, as a consequence of the merging of ? and contraction links, in [GMM97a] we prefered to not include box door links into the corresponding boxes.

Remark 6 The relevant point of Definition 5 is the *connectedness* of boxes.

The reader acquainted with linear logic should have already noted that this means to forbid weakenings, for weakening links might split boxes in several disjoint components. A solution to this problem has been proposed in [GMM97b], where, by a small modification of the underlying sequent calculus, weakening links are connected to axioms. We will come back to this issue discussing garbage collection (section 11.4).

2.3.1 Box nesting property

The only way in which boxes may overlap is the trivial one. Namely, two boxes are either disjoint or enclosed one into the other. Further, in the second case they cannot have the same principal door link.

Definition 7 (box nesting property) *Let* $\mathsf{box}_\mathsf{G} : \mathsf{lnk}_!(\mathsf{G}) \to \mathscr{P}(\mathsf{G})$ *be an assignment of boxes to the ! links of the structure* G*. We say that* G *with the boxes* $\mathsf{BX}_\mathsf{G} = \{\mathsf{box}_\mathsf{G}(e) \mid e \in \mathsf{lnk}_!(\mathsf{G})\}$ *has the* box nesting property*, when* $\mathsf{B}_1 \cap \mathsf{B}_2 \neq \varnothing$ *implies either* $\mathsf{B}_1 \subset \mathsf{B}_2$ *or* $\mathsf{B}_2 \subset \mathsf{B}_1$*, for any* $\mathsf{B}_1, \mathsf{B}_2 \in \mathsf{BX}_\mathsf{G}$*.*

As an immediate consequence of this definition, we see that:

(i) If $\mathsf{pdlink}(\mathsf{B}_1) = \mathsf{pdlink}(\mathsf{B}_2)$, then $\mathsf{B}_1 = \mathsf{B}_2$, for any $\mathsf{B}_1, \mathsf{B}_2 \in \mathsf{BX}_\mathsf{G}$.
(ii) The box nesting property does not forbid $\mathsf{doors}(\mathsf{B}_1) \cap \mathsf{doors}(\mathsf{B}_2) \neq \varnothing$. In fact, even though two boxes always have distinct principal doors, they might share some auxiliary door links.

2.3.2 Box nesting level

The box nesting property gives us a way to avoid the introduction of any global link for the representation of boxes. The technique rests on the assignment of a level to each link/arrow corresponding to the number of boxes enclosing it. According to this intended interpretation, the levels of the arrows incident to a box door link are the ones given in Figure 6.

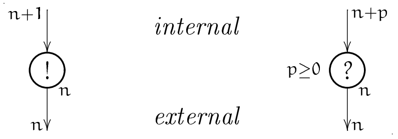

Fig. 6. Exponential links.

In an ! link we always have a difference of 1 between the internal and the external door—for an ! link is always principal door of just one box. Such a difference is rather equal to $p \geq 0$ in a ? link—for a ? link may be auxiliary door of several or even 0 boxes. Note that the levels of the links are an harmless exception to our intended interpretation. In fact, even if box door links belong to the boxes of which they are door links, the level of an ! or a ? link is equal to the level of its external door.

In the case of any other link of the signature, the level of the link and the levels of its doors coincide (see Figure 7).

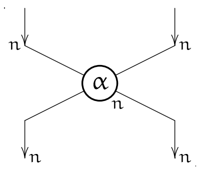

Fig. 7. Levels of a link.

Definition 8 (leveled structure) *An* ℓ-structure G *over the signature* $\Sigma \uplus \{!, ?\}$ *is a structure of links with a* level assignment $\ell_{\mathsf{G}} : (\mathsf{vtx}(\mathsf{G}) \cup \mathsf{lnk}\,\mathsf{G}) \to \mathbb{N}$ *which associates a non-negative* level *to each vertex and each link of* G *s.t.: (*i*)* $\ell_{\mathsf{G}}(\nu) = 0$ *for any* $\nu \in \mathsf{doors}(\mathsf{G})$*; (*ii*) levels accord with the constraints of Figure 6, for the box door links, and of Figure 7, for the links of type* $\alpha \in \Sigma$.

Remark 9 We stress that all the conclusions of an ℓ-structure have level 0.

Remark 10 To label both arrows and links with a level is actually redundant: we might get the same result assigning levels to arrows only, or to links only (in the latter case, provided that levels strictly represent box nesting depths). In fact, links' levels might be recovered from arrows' levels, and vice versa. Both such choices correspond to solutions presented in literature. To label links is more faithful to the original presentation of optimal reduction algorithms (see [Lam90,GAL92a,GAL92b,AL94,Asp95]). To label arrows has instead a more tight correspondence with the logical interpretation of nets (see [MM95,GMM97a]).

2.4 Leveled boxes

Levels can be used to avoid the introduction on an explicit box constructor.

Definition 11 (leveled box) *Let* G *be an* ℓ*-structure and let* $\mathsf{e}_! \in \mathsf{lnk}_!(\mathsf{G})$*. The* ℓ-box *of* $\mathsf{e}_!$ *is the* G*-substructure* $\mathsf{box}^{\ell}_{\mathsf{G}}(\mathsf{e}_!)$ *s.t., being* $\mathsf{B} = \mathsf{box}^{\ell}_{\mathsf{G}}(\mathsf{e}_!)$*:*

(i) $\mathsf{pdlink}(\mathsf{B}) = \mathsf{e}$, i.e., $\mathsf{e} \in \mathsf{dlinks}(\mathsf{B})$*;*
(ii) $\mathsf{e} \in \mathsf{lnk}_?(\mathsf{G})$*, for any* $\mathsf{e} \in \mathsf{dlinks}(\mathsf{B})$*:*
(iii) B *is connected;*
(iv) $\ell_{\mathsf{G}}(\nu) \leq \ell_{\mathsf{G}}(\mathsf{e}_!)$*, for any* $\nu \in \mathsf{doors}(\mathsf{B})$*;*
(v) $\ell_{\mathsf{G}}(\nu) > \ell_{\mathsf{G}}(\mathsf{e}_!)$*, for any arrows* $\nu \in \mathsf{vtx}(\mathsf{B}) \setminus \mathsf{doors}(\mathsf{B})$.

Note that we explicitly exploit connectedness of boxes.

According with previous notations, $\mathsf{BX}^{\ell}_{\mathsf{G}}$ is the set of boxes defined by the function $\mathsf{box}^{\ell}_{\mathsf{G}}$, and "the ℓ-structure G has the box nesting property" means that G with the boxes $\mathsf{BX}^{\ell}_{\mathsf{G}}$ has the box nesting property.

Fact 12 *Let BX_{G} be the boxes of a structure of links G. Let ℓ_{G} be the map assigning to each arrow/link of G its box nesting level. If G has the box nesting property, then $\mathsf{BX}_{\mathsf{G}} = \mathsf{BX}^{\ell}_{\mathsf{G}}$.*

Remark 13 The converse of the previous fact is not true. In fact, we may easily construct an ℓ-structure for which the box nesting property does not hold. Furthermore, even in the case that the ℓ-structure G have the box nesting property, its levels might differ from the ones induced by the box nesting. The latter case has however no impact on our study, Independently from the origin of the levels, it suffices that the corresponding ℓ-structure has the box nesting property. One of the key step of our theory will be the introduction of an algebraic semantics (see section 7) characterizing the ℓ-structures for which the box nesting property holds (see Proposition 51).

2.5 Contraction

To complete the assumptions on the calculi we are interested in, we assume that they contain a contraction operator. Namely, a *contraction* link—being $\curlyvee$ its name—with $\mathsf{k}+1$ doors: the $\mathsf{k} > 0$ *contracting doors* and one *contracted door*. All the contracting doors of a $\curlyvee$ link e are external doors of ? links (see Figure 8), and are connected to ports of e having the same name (in other words the contracting ports are indistinguishable). In the following, we will use $\mathsf{adoors}(\mathsf{e})$ to denote the set of the contracting (auxiliary) doors of a $\curlyvee$ link e and $\mathsf{pdoor}(\mathsf{e})$ to denote its contracted (principal) door.

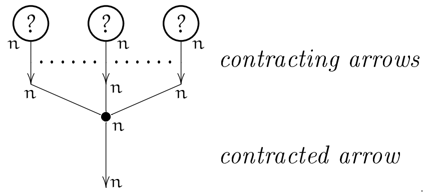

Fig. 8. Contraction link.

Because of the previous assumptions, the orientation of $\curlyvee$ links must accord with the orientation of ? links—therefore, as in the following we will only meet ? links whose external door is their conclusion, the orientation of $\curlyvee$ links will correspond to the one in Figure 8.

Remark 14 We admit the presence of unary contraction links. Their intended interpretation is a direct connection between their doors. In spite of this, for technical reasons, we add the assumption that a ? link is always followed by a $\curlyvee$ link, which implies to insert a dummy unary contraction link below some ? links (this allows a more directed definition of sharing morphism, Definition 17). A different possibility would have been to merge ? and $\curlyvee$ link (as for instance has been done in [GMM97a]).

3 Examples

The next are the relevant examples to which our methodology applies. They are indeed the ones for which sharing graphs have been introduced. Their description is not complete, for we just want to point out their box rewriting rules. For each of them we give anyhow a pointer to an unabridged presentation.

3.1 λ*-calculus*

The use of box door links to explicitly delimit the border of boxes implies that $\lambda\ell$-structures slightly differ from the graphs used in the introduction—apart for the fact that what was a node in one of such graphs becomes a link in a $\lambda\ell$-structure.

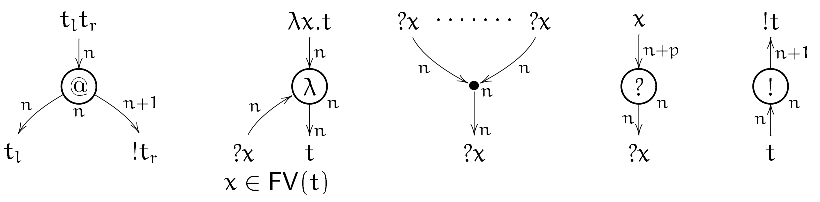

Fig. 9. $\lambda\ell$-terms.

Figure 9 gives the links used to build $\lambda\ell$-structures. The labels at the vertices of the arrows define the formation rules by which to build the correct $\lambda\ell$-structures, say $\lambda\ell$-nets. The introduction of box door links and contraction also cause a small change in the β rule. In fact, the β rule does not involve any more just a pair @/λ, but also the ! link connected to the right port of the @ link, and the contraction and the ? links connected to the binding port of the λ link. Taking into account these considerations, the reformulation of β is direct, and because of this we omit to draw it.

3.2 MELL

The links for the multiplicative-exponential fragment of linear logic (MELL) are drawn if Figure 10. Since contraction, ? and ! links have been borrowed from the exponentials of MELL, to complete the system it suffices to add multiplicative and identity links. Because of the restrictions we have imposed in the definition of box, weakening cannot be directly introduced (for a treatment of weakening see [GMM97a]). Hence, our MELL is without weakening.

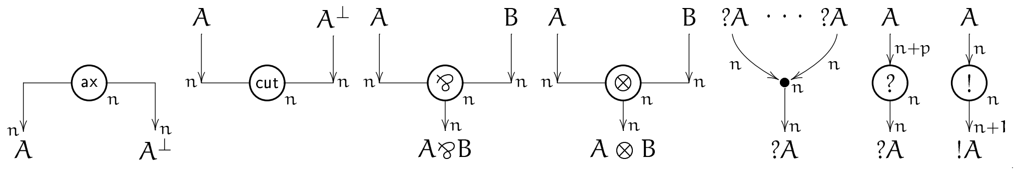

Fig. 10. MELL proof ℓ-nets

The correct MELL ℓ-structures, or MELL proof ℓ-nets, are the ones obtainable by the step-by-step translation of a MELL proof, or by checking that a structure built according to the labeling given in Figure 10 satisfies a given correctness criterion (see [Gir87]), for instance the Danos and Regnier's one.

The rewriting rules are the usual ones, see [Gir87] for their complete set. The only one relevant for our purposes is however the exponential cut-elimination that will be depicted in Figure 22.

3.3 Pure proof nets

Pure proof nets (see [Reg92]) are the nets corresponding to the interpretation of λ-calculus inside linear logic given by the isomorphism $!O \multimap O \simeq O$; that using $⅋$ instead of $\multimap$ becomes $?I ⅋ O \simeq O$, where I and O are two constants s.t. $I = O^\perp$ and $O = I^\perp$. The links are then the same of MELL, while formulas differ. There are only four types of formulas: $I, ?I$, say inputs; and $O, !O$, say outputs. Any input formula is the dual of the corresponding output formula, *i.e.*, $I = O^\perp$, $?I = (!O)^\perp$, and $X^{\perp\perp} = X$, for any X. The rules for link composition are given in Figure 11.

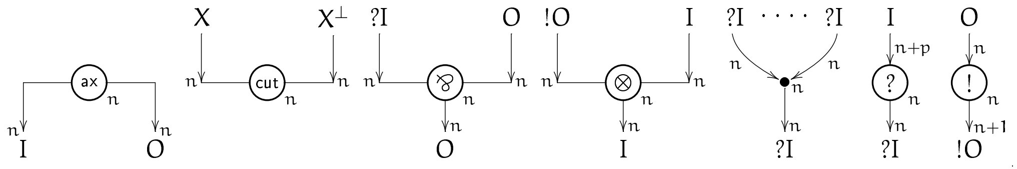

Fig. 11. Pure proof ℓ-nets

It is immediate to see that apart for the identity links, and for name and orientation of the other links, there is a direct embedding of $\lambda\ell$-nets into pure proof ℓ-nets: replace each λ link with a $⅋$ link and each @ link with a $\otimes$ link, then, insert a suitable cut or ax link in any place where the orientation of the arrows would otherwise be inconsistent. The set of the pure proof ℓ-nets is however much wider than the one obtainable translating $\lambda\ell$-nets and is defined by the same correctness criteria used to define MELL proof ℓ-nets (see [Reg92]). Nevertheless, not any pure proof ℓ-net can be interpreted as a MELL proof ℓ-net—for instance, for the nets obtained by translating $\lambda\ell$-nets, the pure proof ℓ-net is also a MELL proof net only if the corresponding λ-term is typable.

The rewriting rules are the same of MELL. Hence, the relevant one for our purposes is the box interaction that will be depicted in Figure 22. The main difference between $\lambda\ell$-nets and pure proof ℓ-nets is that the β rule of the first ones splits in two phases in the second ones. Firstly, an interaction between the $\parr$ and $\otimes$ links above the box door links of the redex, then a box interaction as the one of Figure 22. This explains why there are pure proof ℓ-nets which are not image of any $\lambda\ell$-net—executing some β rules, we might have done the $\parr/\otimes$ interactions only. Besides, assuming to complete all such β rules the result is the image of a $\lambda\ell$-term. The latter property holds in general for any pure proof ℓ-net and not only for the one obtained by reducing the image of a $\lambda\ell$-net (see [Reg92], for more details).

4 Sharing structures

The link in charge of the lazy duplication of boxes is the multiplexer. As for contraction and box door links, we assume that its type $\triangledown$ is a reserved name.

Definition 15 (multiplexer) *A* k*-ary multiplexer or* k-mux *is a link of type* $\triangledown$ *with:*

(i) *one* principal port*;*
(ii) *a sequence of* $k > 0$ auxiliary ports*, whose names* $a_1, \ldots, a_k$ *are chosen over a denumerable set of symbols, say* $\mathbb{N}$*, with the proviso* $a_i = a_j$ *iff* $i = j$*;*
(iii) *an associate non-negative integer* m*, the* threshold *of the mux;*
(iv) *an associate sequence of* k *integers* $q_1, \ldots, q_k$*, the (auxiliary port)* offsets*, s.t.* $q_i \geq -1$ *for* $i = 1, 2, \ldots, k$*.*

The level of a mux $e_\triangledown$ *is equal to its threshold, that is,* $\ell(e_\triangledown) = m$*.*

In a mux $e_\triangledown$ the principal door $\mathsf{pdoor}(e_\triangledown)$ may be either a conclusion or a premise—in the first case $e_\triangledown$ is a *positive mux*, in the second case it is a *negative mux*. The auxiliary doors $\mathsf{adoors}(e)$ accord with the orientation of the principal one—in a positive mux they are premises, in a negative mux they are conclusions.

Figure 12 gives the relations between the levels assigned to the doors of a mux (and the graphical representation of a mux link). Namely:

– each offset q_i is the difference between the level of the i-th auxiliary door and the principal door;
– the threshold m is lower than the level of the principal door.

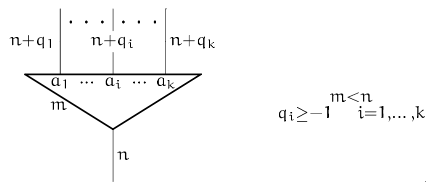

Fig. 12. The mux (multiplexer) link.

Definition 16 ($s\ell$-structure) *A (leveled)* sharing structure *or $s\ell$-structure over $\Sigma \uplus \{!, ?, \curlyvee\}$ is an ℓ-structure over $\Sigma \uplus \{!, ?, \curlyvee, \triangledown\}$ in which, for any $\alpha \in \Sigma$, all the links of type α have a fixed set of named ports.*

It is worth to summarize some consequences of the previous definition. In any $s\ell$-structure:

(i) For any $\alpha \in \Sigma$, all the links of type α have the same cardinality.
(ii) The port names of a link of type $\alpha \in \Sigma$ are distinct.
(iii) Box door links and contractions are the only links whose cardinality is not strictly fixed.
(iv) Non unary contractions are the only links in which several distinct ports (all the contracting arrows) have the same name.

4.1 Muxes, fans, and brackets

Muxes could be easily reformulated in terms of fans and brackets. Figure 13 gives the translation for the binary case. We see that a mux is a way to aggregate suitable patterns of fans and brackets. The offset q of each auxiliary port corresponds to a sequence of $q + 1$ brackets followed by a croissant. The unique fan used in the binary case is replaced by a tree of fans with k leaves in the case of a k-mux.

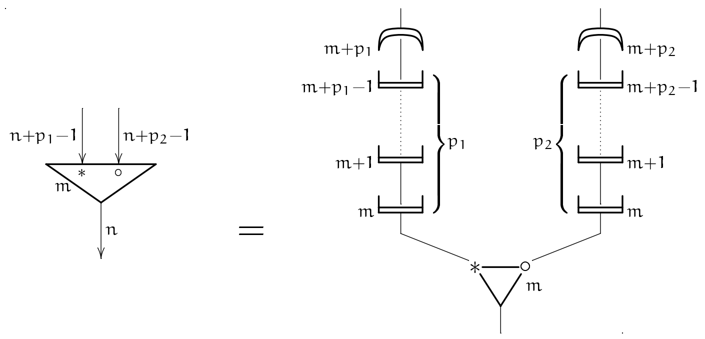

Fig. 13. Equivalence between muxes and brackets.

5 Unfolding the sharing

Definition 17 (sharing morphism) *An* s-morphism *(sharing morphism) is a surjective homomorphism of* $s\ell$*-structures* $\mathsf{M} : \mathsf{G}_0 \to \mathsf{G}_1$ *whose restriction to* $\mathsf{doors}(\mathsf{G}_0)$ *is injective. Namely,* M *is a map from* $\mathsf{lnk}(\mathsf{G}_0) \uplus \mathsf{vtx}(\mathsf{G}_0)$ *to* $\mathsf{lnk}(\mathsf{G}_1) \uplus \mathsf{vtx}(\mathsf{G}_1)$ *s.t.:*

(i) $\mathsf{M}(\mathsf{lnk}(\mathsf{G}_0)) = \mathsf{lnk}(\mathsf{G}_1)$ *and* $\mathsf{M}(\mathsf{vtx}(\mathsf{G}_0)) = \mathsf{vtx}(\mathsf{G}_1)$*;*
(ii) the links e *and* $\mathsf{M}(e)$ *are of the same type;*
(iii) if the arrow v *is connected to the port with name* a *of the link* e*, then* $\mathsf{M}(v)$ *is connected to the port with name* a *of the link* $\mathsf{M}(e)$*;*
(iv) $\ell_{\mathsf{G}_0}(x) = \ell_{\mathsf{G}_1}(\mathsf{M}(x))$*, for any* $x \in \mathsf{lnk}(\mathsf{G}_0) \uplus \mathsf{vtx}(\mathsf{G}_0)$*;*
(v) if $v, v' \in \mathsf{doors}(\mathsf{G}_0)$ *and* $\mathsf{M}(v) = \mathsf{M}(v')$*, then* $v = v'$*.*

It is worth to note that an s-morphism establishes a bijection between the doors of G_0 and G_1. In fact, as it maps roots to roots, a door of G_1 is definitely image of a (unique) door of G_0.

Contraction and muxes are the unique links that may change cardinality via an s-morphism. Nevertheless, muxes and $\curlyvee$ links present a relevant difference. In fact, while it is impossible to equate two auxiliary ports of a mux via an s-morphism, we may definitely have $\mathsf{M}(v) = \mathsf{M}(v')$ for some $\curlyvee$ link $e_\curlyvee$ and some pair $v, v' \in \mathsf{adoors}(e_\curlyvee)$.

Figure 14 is an example of s-morphism between two $\lambda s\ell$-structures.

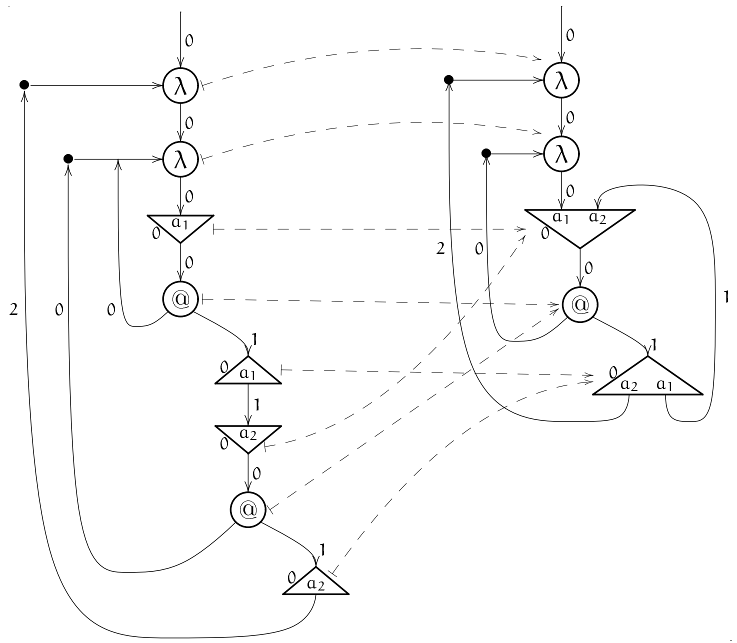

Fig. 14. Sharing morphism.

Proposition 18 *The* $s\ell$*-structures are a partial order w.r.t. the relation* $\preccurlyeq$

defined by "$\mathsf{G} \preccurlyeq \mathsf{G}'$ *when there exists an* s*-morphism* $\mathsf{M} : \mathsf{G} \to \mathsf{G}'$*".*

Proof. Reflexivity and transitivity of $\preccurlyeq$ are immediate. So we need to show antisymmetry only. Namely, we have to prove that $\mathsf{G} \preccurlyeq \mathsf{G}'$ and $\mathsf{G}' \preccurlyeq \mathsf{G}$ implies $\mathsf{G} \simeq \mathsf{G}'$ (*i.e.*, they are isomorphic). Let $\mathsf{M} : \mathsf{G} \to \mathsf{G}'$ and $\mathsf{M}' : \mathsf{G}' \to \mathsf{G}$. These two s-morphisms induce two corresponding pairs of surjective maps between $\mathsf{lnk}(\mathsf{G})$ and $\mathsf{lnk}(\mathsf{G}')$, and between $\mathsf{vtx}(\mathsf{G})$ and $\mathsf{vtx}(\mathsf{G}')$. Thus, $|\mathsf{lnk}(\mathsf{G})| = |\mathsf{lnk}(\mathsf{G}')|$ and $|\mathsf{vtx}(\mathsf{G})| = |\mathsf{vtx}(\mathsf{G}')|$, that is, both M and M' are isomorphisms. $\square$

5.1 Unshared structures

A relevant case of $\mathsf{s}\ell$-structures are the ones in which there is no sharing at all, that is, in which all the muxes are unary. In our intention, they should be the minimal elements of the equivalence classes that $\preccurlyeq$ defines.

Definition 19 (lift) *A* lift *is a multiplexer with only one auxiliary port.*

Definition 20 (unshared structure) *A* $\mathsf{u}\ell$*-structure over the signature* Σ *is an* $\mathsf{s}\ell$*-structure over* Σ *in which all the muxes are lifts.*

Unfortunately, the relation $\preccurlyeq$ is not yet the unfolding ordering we are looking for, since the presence of contraction links implies that a $\mathsf{u}\ell$-structure might not be minimal. For instance, let us take a $\mathsf{u}\ell$-structure U such that all its doors are conclusions of a contraction; there exist a denumerable set of $\mathsf{u}\ell$-structures U' s.t. $\mathsf{U}' \preccurlyeq \mathsf{U}$—the k-th of them can be built making k instances of U and merging the $\curlyvee$ links above corresponding conclusions.

5.2 Correctness of the unfolding

The absence of a minimal element is not the only lack of $\preccurlyeq$. The quest for a correct definition of the unfolding partial order faces with an even stronger problem: not all the less-shared-instances definable via $\preccurlyeq$ can be considered a "correct" unfolding of an $\mathsf{s}\ell$-structure G.

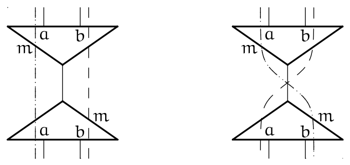

Fig. 15. Matching of ports

In fact, let us assume that G contain a pair of binary muxes e_1 and e_2 with $\ell(e_1) = \ell(e_2)$. In our aims this situation corresponds to a case of matching muxes. Therefore, in a correct $s\ell$-structure there should be a perfect matching between the ports of e_1 and the ports of e_2, and any "correct" unfolding of G should preserve such a property. For instance, let U be a $u\ell$-structure s.t. $M : U \preccurlyeq G$. Any pair of lifts e'_1, e'_2 of U s.t. $M(e'_i) = e_i$, with $i = 1, 2$, should be matching. Referring to Figure 15, this means that each pair e'_1, e'_2 should be the image of one of the paths drawn on the left-hand side pair of muxes. Any unfolding of G containing an image of one of the paths drawn on the right-hand side pair of muxes should instead be forbidden in any correct unfolding of G.

6 Overview of the main results

The algebraic semantics we are going to present in section 7 formalizes the matching problem we mentioned in the previous section. Such a semantics is an algebraic characterization of the *proper $u\ell$-structures*—the only $u\ell$-structures which are "correct" unfoldings of sharing graphs. Via a suitable restriction of the s-morphisms, we get at the same time a characterization of the *proper $s\ell$-structures*—the $s\ell$-structures for which there exists a correct unfolding (*i.e.*, s.t. among their less-shared-instances there is a proper $u\ell$-structure).

In more details. Once defined the proper $u\ell$-structures, we will restrict $\preccurlyeq$ to a partial order $\prec\!\!\!\preccurlyeq$ (is a proper unfolding of), and we will define proper an $s\ell$-structure G when $U \prec\!\!\!\preccurlyeq G$ for some proper $u\ell$-structure U.

That $\prec\!\!\!\preccurlyeq$ is the right unfolding partial order will be proved in section 8 showing that:

(i) Any proper ℓ-structure is the only element of its equivalence class w.r.t. the transitive, reflexive and symmetric closure of $\prec\!\!\!\preccurlyeq$.
(ii) The proper $u\ell$-structures are minimal w.r.t. $\prec\!\!\!\preccurlyeq$.
(iii) For any proper $s\ell$-structure G, there is exactly one proper $u\ell$-structure U, say the *least-shared-instance* of U, for which $U \prec\!\!\!\preccurlyeq G$.

The basic tool to prove such properties is the read-back reduction system, or π rules, that we will define in section 8—a direct implementation of the interpretation of muxes as reindexing-duplication operators. In fact, properness will be proved stable under the step-by-step unfolding performed applying the π rules. Moreover, we will see that the proper $s\ell$-structures are indeed the $s\ell$-structures which π normalize to an ℓ-structure with the box nesting property.

The uniqueness of the least-shared-instance of a proper $\mathsf{s}\ell$-structure will allow to semantically associate an ℓ-structure to each proper $\mathsf{s}\ell$-structure. In fact, the notion of *solution* of a $\mathsf{u}\ell$-structure, that we will introduce to define properness, will give us a natural way to *semantically read-back* an ℓ-structure $\mathscr{R}(\mathsf{U})$ from a proper $\mathsf{u}\ell$-structure U: to take as semantical read-back of a proper $\mathsf{s}\ell$-structure the semantical read-back of its least-shared-instance. Such a definition will be proved sound w.r.t. the π rules and, moreover, we will show that the π normal form of a proper $\mathsf{s}\ell$-structure coincides with its semantical read-back.

Soundness of properness and read-back w.r.t. β rule will be shown in section 10.

7 Algebraic semantics

To introduce the algebraic part, a prelude on the naive attempts that originated this kind of approach.

7.1 Lifting functions

It is natural to interpret a $\mathsf{u}\ell$-structure U as a representation of the structure N obtainable removing the lifts of U and merging the pairs of corresponding arrows that otherwise would remain dangling. Nevertheless, it is readily seen that this procedure would erase the boxes of U (if any), for in general there would not be a unique way to reassign a level to each arrow/link of N (the λ-structures are a remarkable exception to this, the levels of their links/arrows are determined once given the underlying λ-tree).

To understand the nature of the problem, it is worth to attempt a direct algebraic interpretation of lifts, following the idea that a lift with offset q is a sort of bracket delimiting the part of a structure, say its scope, that has to be lifted by the offset q. As an arrow might be in the scope of several lifts (to simplify, let us assume that lift scopes are substructures whose border is made of lifts with the same parameters, and that lift scopes have a well-nesting property similar to the one of boxes), we need a general way to reason about the displacement of levels that take place at a given arrow. Hence, let us assume that each arrow carry a function $f_\nu : \mathbb{Z} \to \mathbb{Z}$ assigning a local name to each level, that is, let $f_\nu(n)$ be the actual value of the level n at the arrow ν. By the way, in order to get a sound renaming at each arrow, we should also impose some restrictions on the functions f, as for instance their monotonicity. Anyhow, let us go on without too details, our purpose is just to show why such an approach is still too weak.

The ℓ-structures are the base case. In them, there is no reindexing at all, for there are no lifts around. Thus, any arrow of an ℓ-structure should carry the identity function. In the $u\ell$-structures instead, the interpretation of a lift $e_\triangledown$ as a reindexing (or shifting) operator for the levels above its threshold leads to associate to $e_\triangledown$ a functional $\delta_{m,q} : \mathbb{Z}^\mathbb{Z} \to \mathbb{Z}^\mathbb{Z}$ (being m the threshold and q the offset of $e_\triangledown$) defined by

$$\delta_{m,q}(f)(i) = \begin{cases} f(i) & \text{when } i \leq m \\ f(i+q) & \text{otherwise} \end{cases}$$

and to assume that

$$f_{\nu_p} = \delta_{m,q}(f_{\nu_a})$$

where $\nu_p = \mathsf{pdoor}(e_\triangledown)$ and $\{\nu_a\} = \mathsf{adoors}(e_\triangledown)$.

Therefore, the functions assigned to the arrows ν of U should have the shape

$$f_\nu = \delta_{m_1,q_1}\delta_{m_2,q_2}\cdots\delta_{m_k,q_k}(id) \tag{4}$$

with a factor δ_{m_i,q_i} for each lift that contains ν in its scope.

Provided to succeed assigning a function f_ν to each arrow ν. We see that $f_{\nu_p}(\ell(\nu_p)) = f_{\nu_a}(\ell(\nu_a))$ for any lift $e_\triangledown$. Hence, given a $u\ell$-structure U we could read-back an ℓ-structure N just assigning the level $f_\nu(\ell_U(\nu))$ to each arrow of U and removing the lifts contained in it. (For the sake of completeness, some other constraints should be added to the one for lifts in order to ensure that proper levels are assigned to the doors of ? and ! links, but for the moment this is irrelevant.)

The latter procedure is the base of the semantical read-back we will give in section 7.9. Nevertheless, the approach based on the functional δ is still inadequate: it does not ensure that two lifts with the same threshold connected through their principal ports have the same name and offset (see section 5.2). Besides, even when this happens, it does not ensure that the function assigned to the premise of the positive lift is equal to the function assigned to the conclusion of the negative one, which is mandatory if we want that, after the replacement of a symmetric pair of facing lifts by an arrow connecting their auxiliary ports, the assignment of the functions f be preserved.

7.2 Lifting operators

Having in mind the last considerations of the previous section, we aim at finding a better algebraic interpretation of lifts by a further step of abstraction.

Let $\mathbb{D}$ be a domain with a family $\{\mathcal{L}[m, q, a] \mid m, q, a \in \mathbb{Z}, m \geq 0, q \geq -1\}$ of indexed endomorphisms, say *lifting operators* (see section 7.4), for which the equations:

$$\begin{array}{c}\mathcal{L}[m, q_1, a_1](d_1) = \mathcal{L}[m, q_2, a_2](d_2) \\ \textit{implies} \quad q_1 = q_2 \wedge a_1 = a_2 \wedge d_1 = d_2\end{array} \tag{LO1}$$

$$\mathcal{L}[m_2, q_2, a_2]\, \mathcal{L}[m_1, q_1, a_1] = \mathcal{L}[m_1, q_1, a_1]\, \mathcal{L}[m_2 + q_1, q_2, a_2] \tag{LO2}$$

$$\begin{array}{l}\mathcal{L}[m_1, q_1, a_1](d_1) = \mathcal{L}[m_2, q_2, a_2](d_2) \\ \quad \textit{iff} \quad \exists\, d : \ \mathcal{L}[m_2 + q_1, q_2, a_2](d) = d_1 \wedge \mathcal{L}[m_1, q_1, a_1](d) = d_2\end{array} \tag{LO3}$$

hold provided $m_1 < m_2$.

Remark 21 Equation (LO1) encodes the matching problem described in section 5.2. Equation (LO2) corresponds to the idea that, when $m_1 < m_2$, to lift the levels above m_1 by q_1 and the levels above m_2 by q_2 is equivalent at to lift the levels above $m_2 + q_1$ by q_2 and the levels above m_1 by q_1. This fact is immediate for positive offsets and is true even when $q_i \geq -1$. It corresponds to the analogous commutativity equation:

$$\delta_{m_2,q_2}\, \delta_{m_1,q_1} = \delta_{m_1,q_1}\, \delta_{m_2+q_1,q_2}.$$

Equation (LO3) is the analog of the swap equation (LO2), but for the case in which a certain d_0 can been obtained applying two different operators, that is, $\mathcal{L}[m_1, q_1, a_1](d_1) = d_0 = \mathcal{L}[m_2, q_2, a_2](d_2)$. Its aim is to force the uniqueness, modulo Equation (LO3), of the way in which such a d_0 is constructed, *i.e.*, that $d_0 = \mathcal{L}[m_1, q_1, a_1]\, \mathcal{L}[m_2, q_2, a_2](d)$, for some (unique) d.

We want to find a semantics that assign a suitable product like the one of equation (4) to each arrow of a $u\ell$-structure, assuming that the lifting operators $\mathcal{L}[m, q, a]$ take the place of the integer functionals $\delta_{m,q}$. But to accomplish our project, we still need a better axiomatization of lifting operators (in the style of Danos and Regnier's dynamic algebra, *e.g.*, see [DR93]) and a detailed study of their properties.

7.3 Left inverses of lifting operators

The endomorphisms $\mathcal{L}[m, q, a]$ are injective (by equation (LO1)). Hence, each $\mathcal{L}[m, q, a]$ has a left inverse $\overline{\mathcal{L}}[m, q, a]$. Such an inverse is however not unique in general, since it may assume any value outside the codomain of $\mathcal{L}[m, q, a]$. But, if we consider partial functions too, the natural left inverse $\mathcal{F}^*$ of a partial endomorphism $\mathcal{F}$ is the less defined partial transformation $\mathcal{F}^*$ s.t. $\mathcal{F}\,\mathcal{F}^*\,\mathcal{F} = \mathcal{F}$ (*i.e.*, $\mathsf{dom}(\mathcal{F}^*) = \mathsf{codom}(\mathcal{F})$ and $\mathcal{F}^*\,(\mathcal{F}(d)) = d$, for any $d \in \mathsf{dom}(\mathcal{F})$).

In more details, for any endomorphism $\mathcal{L}[\mathsf{m}, \mathsf{q}, \mathsf{a}]$, we have that:

$$(\mathcal{L}[\mathsf{m}_1, \mathsf{q}_1, \mathsf{a}_1] \cdots \mathcal{L}[\mathsf{m}_k, \mathsf{q}_k, \mathsf{a}_k])^* = \overline{\mathcal{L}}[\mathsf{m}_k, \mathsf{q}_k, \mathsf{a}_k] \cdots \overline{\mathcal{L}}[\mathsf{m}_1, \mathsf{q}_1, \mathsf{a}_1],$$

where

$$\overline{\mathcal{L}}[\mathsf{m}, \mathsf{q}, \mathsf{a}](\mathsf{d}) = \begin{cases} \bar{\mathsf{d}} & \text{when } \mathsf{d} = \mathcal{L}[\mathsf{m}, \mathsf{q}, \mathsf{a}](\bar{\mathsf{d}}) \\ \perp & \text{when } \mathsf{d} \notin \mathsf{codom}(\mathcal{L}[\mathsf{m}, \mathsf{q}, \mathsf{a}]) \end{cases}$$

(being $\overline{\mathcal{L}}[\mathsf{m}, \mathsf{q}, \mathsf{a}](\mathsf{d}) = \perp$ just a denotation for $\mathsf{d} \notin \mathsf{dom}(\mathcal{L}[\mathsf{m}, \mathsf{q}, \mathsf{a}])$).

We get in this way a monoid LSeq^* of injective partial transformations of $\mathbb{D}$ that is closed under left inversion, for it is immediate that

$$\overline{\mathcal{L}}[\mathsf{m}, \mathsf{q}, \mathsf{a}]^* = \mathcal{L}[\mathsf{m}, \mathsf{q}, \mathsf{a}].$$

Furthermore, LSeq^* is a left inverse semigroup, and the equations ($\mathsf{LO1\text{-}3}$) can be nicely reformulated in it.

Remark 22 All the results of the paper could be indeed obtained without left inverse lifting operators. In fact, finding the solutions of a $\mathfrak{u}\ell$-structure (section 7.8) we will use lifting operators only (*cf.* [GMM97a]). Nevertheless, their introduction simplify the proof of some algebraic properties and, more important, gives a better idea of the relations between our algebraic approach and Geometry of Interaction.

7.4 The inverse semigroup LSeq^*

First, a remind of the definition of (left) inverse semigroup.

Definition 23 (left inverse semigroup) *An* inverse semigroup *with* 0 *is a monoid* $\mathbb{S}$ *with an absorbing element (*i.e., *an element* 0 *s.t.* $0\,\mathcal{F} = 0 = \mathcal{F}\,0$*, for any* $\mathcal{F}$*) closed under an involution operation* $(\cdot)^*$ *s.t., for any* $\mathcal{F}, \mathcal{F}_1, \mathcal{F}_2 \in \mathbb{S}$ *:*

$$\begin{gathered} \mathcal{F}^{**} = \mathcal{F} \\ (\mathcal{F}_1\,\mathcal{F}_2)^* = \mathcal{F}_2^*\,\mathcal{F}_1^* \\ \langle\mathcal{F}\rangle\,\mathcal{F} = \mathcal{F} \\ \langle\mathcal{F}_1\rangle\,\langle\mathcal{F}_2\rangle = \langle\mathcal{F}_2\rangle\,\langle\mathcal{F}_1\rangle \end{gathered}$$

where $\langle\mathcal{F}\rangle \equiv \mathcal{F}\mathcal{F}^*$.

The axioms of inverse semigroups allow to immediately prove that:

(i) Each $\langle \mathcal{F} \rangle$ is invariant under involution and is an idempotent of $\mathbb{S}$, *i.e.*,

$$\langle \mathcal{F} \rangle^* = \langle \mathcal{F} \rangle$$
$$\langle \mathcal{F} \rangle \langle \mathcal{F} \rangle = \langle \mathcal{F} \rangle$$

since $\langle \mathcal{F} \rangle \langle \mathcal{F} \rangle = \langle \mathcal{F} \rangle \mathcal{F} \mathcal{F}^* = \mathcal{F} \mathcal{F}^* = \langle \mathcal{F} \rangle$.

(ii) Each idempotent of LSeq^* may be written as $\langle \mathcal{F} \rangle$ for some $\mathcal{F}$ and is then invariant under involution, *i.e.*,

$$\mathcal{F}\mathcal{F} = \mathcal{F} \quad \textit{implies} \quad \langle \mathcal{F} \rangle = \mathcal{F}$$
$$\mathcal{F}\mathcal{F} = \mathcal{F} \quad \textit{implies} \quad \mathcal{F}^* = \mathcal{F}$$

since, if $\mathcal{F}\mathcal{F} = \mathcal{F}$, then $\mathcal{F} = \langle \mathcal{F} \rangle \mathcal{F} = \langle \mathcal{F} \rangle \langle \mathcal{F}^* \rangle = \langle \mathcal{F}^* \rangle \langle \mathcal{F} \rangle = \mathcal{F}^* \langle \mathcal{F} \rangle = (\langle \mathcal{F} \rangle \mathcal{F})^* = \mathcal{F}^*$, and then $\langle \mathcal{F} \rangle = \mathcal{F}$. Which in particular implies $1^* = 1$ and $0^* = 0$.

A *lifting operator* is a triple of integers $\mathcal{L}[m, q, a]$, with $m, a \geq 0$ and $q \geq -1$. The index m is the *threshold* of the lifting operator; the index q is its *offset*; the index a is its *name*. For each lifting operator there is a corresponding barred triple $\overline{\mathcal{L}}[m, q, a]$ called its *left inverse*.

Definition 24 (LSeq^*) *The inverse semigroup* LSeq^* *is the smallest inverse semigroup generated by composition of lifting operators and left inverse lifting operators according to the axioms:*

$$\mathcal{L}[m, q, a]^* = \overline{\mathcal{L}}[m, q, a] \qquad (\mathsf{LS0})$$
$$\overline{\mathcal{L}}[m, q, a]\, \mathcal{L}[m, q, a] = 1 \qquad (\mathsf{LS1})$$
$$\overline{\mathcal{L}}[m, q_2, a_2]\, \mathcal{L}[m, q_1, a_1] = 0 \qquad \textit{if } q_1 \neq q_2 \textit{ or } a_1 \neq a_2 \qquad (\mathsf{LS2})$$
$$\mathcal{L}[m_2, q_2, a_2]\, \mathcal{L}[m_1, q_1, a_1] = \mathcal{L}[m_1, q_1, a_1]\, \mathcal{L}[m_2 + q_1, q_2, a_2] \qquad (\mathsf{LS3})$$
$$\overline{\mathcal{L}}[m_2, q_2, a_2]\, \mathcal{L}[m_1, q_1, a_1] = \mathcal{L}[m_1, q_1, a_1]\, \overline{\mathcal{L}}[m_2 + q_1, q_2, a_2] \qquad (\mathsf{LS4})$$

when $m_1 < m_2$.

The natural model of LSeq^* is the monoid generated by the indexed endomorphisms of $\mathbb{D}$ (the lifting operators) and by their left inverses (the left inverse lifting operators). Under this interpretation, it is readily seen that:

(i) The constant 0 is the nowhere defined partial transformation of $\mathbb{D}$.

(ii) Any idempotent $\langle \mathcal{F} \rangle$ of LSeq^* is the identity function restricted to the codomain of $\mathcal{F}$, *i.e.*, $\mathsf{dom}(\langle \mathcal{F} \rangle) = \mathsf{codom}(\mathcal{F})$. In particular, 1 is the identity endomorphism of $\mathbb{D}$.

(iii) Axioms ($\mathsf{LS1}$) and ($\mathsf{LS2}$) are equivalent to equation ($\mathsf{LO1}$). Axiom ($\mathsf{LS3}$) coincide with equation ($\mathsf{LO2}$). Axiom ($\mathsf{LS4}$) corresponds to the *only if* part of equation ($\mathsf{LO3}$) (the *if* part is subsumed by equation ($\mathsf{LO3}$)).

7.5 Lifting sequences

A *lifting sequence* $\mathcal{H}$ is a finite product of lifting operators. The monoid LSeq is the smallest one containing the lifting operators (*i.e.*, no left inverse operator $\overline{\mathcal{L}}[m, q, a]$ belongs to it).

By iterated application of axiom ($\mathsf{LS3}$), each lifting sequence is equivalent to a (unique) sequence in *canonical form* with the thresholds non-decreasingly ordered, *i.e.*, for any $\mathcal{H}$ there exists a sequence $\mathcal{H} = \prod_{0<i\leq k} \mathcal{L}[m_i, q_i, a_i]$, with $m_i \leq m_j$ if $i < j$.

By induction on the length $|\mathcal{H}| = k$ of the lifting sequence, it is indeed direct to see that $\mathcal{H}^* \mathcal{H} = \langle \mathcal{H}^* \rangle = 1$ (note that this accord with the interpretation in terms of partial endomorphisms of $\mathbb{D}$, for $\mathsf{codom}(\mathcal{H}^*) = \mathsf{dom}(\mathcal{H}) = \mathbb{D}$).

Let $n_0 \leq n_1$. A *lifting sequence from* n_0 *to* n_1 is a product of lifting operators $\mathcal{H} = \prod_{0<i\leq k} \mathcal{L}[m_i, q_i, a_i]$ in which

$$n_0 \leq m_i < n_1 + \sum_{0<j<i} q_j,$$

for $i = 1, 2, \ldots, k$.

Definition 25 ($\mathsf{LSeq}[n_0, n_1]$) *The monoid* $\mathsf{LSeq}[n_0, n_1]$ *is the smallest one containing the lifting sequences from* n_0 *to* n_1.

The last definition is sound as it is invariant under application of axiom ($\mathsf{LS3}$). Furthermore,

$$\mathsf{LSeq} = \mathsf{LSeq}[0, \omega] = \bigcup_{n\in\omega} \mathsf{LSeq}[0, n]$$

for $\mathsf{LSeq}[n_0, n_1] \subseteq \mathsf{LSeq}[m_0, m_1]$, when $m_0 \leq n_0$ and $n_1 \leq m_1$.

The *global offset* $||\mathcal{H}||$ of a lifting sequence is the sum of the offsets of its lifting operators, *i.e.*, if $\mathcal{H} = \prod_{0<i\leq k} \mathcal{L}[m_i, q_i, a_i]$, then

$$||\mathcal{H}|| = \sum_{0<i\leq k} q_i.$$

Let $\mathcal{H} = \prod_{0<i\leq k} \mathcal{L}[m_i, q_i, a_i]$ be a lifting sequence and let r be an integer s.t. $m_i + r \geq 0$, for $i = 1, 2, \ldots, k$. The *lifting of* $\mathcal{H}$ *by the offset* r is the lifting sequence

$$\mathcal{H}^{\uparrow r} = \prod_{0<i\leq k} \mathcal{L}[m_i + r, q_i, a_i].$$

Fact 26 *Let* $n_0 \le n_1 \le n_2$.

(i) $n_1 + \|\mathcal{H}\| \ge n_0$, *for any* $\mathcal{H} \in \mathsf{LSeq}[n_0, n_1]$.

(ii) If $\mathcal{H}_1 \in \mathsf{LSeq}[n_0, n_1]$ *and* $\mathcal{H}_2 \in \mathsf{LSeq}[n_1, n_2]$, *then:*

(a) $\mathcal{H}_2\,\mathcal{H}_1 \in \mathsf{LSeq}[n_0, n_2]$;

(b) $\mathcal{H}_2\,\mathcal{H}_1 = \mathcal{H}_1\,\mathcal{H}_2^{\uparrow\|\mathcal{H}_1\|}$;

(c) $\mathcal{H}_2^*\,\mathcal{H}_1 = \mathcal{H}_1\,(\mathcal{H}_2^{\uparrow\|\mathcal{H}_1\|})^*$.

(iii) For any $\mathcal{H} \in [n_0, n_2]$, *there exists a unique pair* $\mathcal{H}_1 \in \mathsf{LSeq}[n_0, n_1]$ *and* $\mathcal{H}_2 \in \mathsf{LSeq}[n_1, n_2]$ *s.t.* $\mathcal{H} = \mathcal{H}_2\,\mathcal{H}_1$.

Proof. The first two items are immediate by induction on $|\mathcal{H}|$. For the last one, let $\mathcal{H} = \prod_{0<i<k} \mathcal{L}[m_i, q_i, a_i]$ be in canonical form. Let h be the first index for which $n + Q_i \ge m_i$ (where $Q_i = \sum_{0\le j<i} q_j$) if any, or let $h = k+1$ otherwise. Let us take $\mathcal{H}_1 = \prod_{0<i<h} \mathcal{L}[m_i, q_i, a_i]$ and $\mathcal{H}_2 = \prod_{h\ge i<k} \mathcal{L}[m_i - Q_h, q_i, a_i]$. By definition, $\mathcal{H}_1 \in \mathsf{LSeq}[n_0, n_1]$ and $\mathcal{H}_2 \in \mathsf{LSeq}[n_1, n_2]$. Furthermore, $\mathcal{H}_2\,\mathcal{H}_1 = \mathcal{H}_1\,\mathcal{H}_2^{\uparrow Q_h} = \mathcal{H}$. The construction of $\mathcal{H}_1$ and $\mathcal{H}_2$ also proves their uniqueness. □

Proposition 27 (canonical form) *For any* $\mathcal{F} \in \mathsf{LSeq}^*$, *with* $\mathcal{F} \ne 0$, *there is a unique pair* $\mathcal{H}_+, \mathcal{H}_- \in \mathsf{LSeq}$ *s.t.* $\mathcal{F} = \mathcal{H}_+\,\mathcal{H}_-^*$.

Proof.

(existence) By induction on $|\mathcal{F}|$. The cases $|\mathcal{F}| = 0$ and $\mathcal{F} = \mathcal{L}[m, q, a]\,\mathcal{F}_1$ are direct. So, let us take $\mathcal{F} = \overline{\mathcal{L}}[m, q, a]\,\mathcal{F}_1$. By the induction hypothesis, we have $\mathcal{F} = \overline{\mathcal{L}}[m, q, a]\,\mathcal{H}_+\,\mathcal{H}_-^*$, for some $\mathcal{H}_+, \mathcal{H}_- \in \mathsf{LSeq}$. Let $\mathcal{H}_+$ be in canonical form. By Fact 26, there are $\mathcal{H} \in \mathsf{LSeq}[0, m]$ and $\mathcal{H} \in \mathsf{LSeq}[m, \omega]$ s.t. $\mathcal{H} = \mathcal{H}_2\,\mathcal{H}_1$. If $\mathcal{H}_2 \in \mathsf{LSeq}[m+1, \omega]$, then $\overline{\mathcal{L}}[m, q, a]\,\mathcal{H} = \mathcal{H}_2^{\uparrow q}\,\overline{\mathcal{L}}[m, q, a]\,\mathcal{H}_1 = \mathcal{H}_2^{\uparrow q}\,\mathcal{H}_1\,\overline{\mathcal{L}}[m, q + \|\mathcal{H}_1\|, a]$, and thus the thesis. Otherwise, $\mathcal{H}_2 = \mathcal{L}[m, q, a]\,\mathcal{H}_2'$ with $\mathcal{H}_2' \in \mathsf{LSeq}$ (by the hypothesis $\mathcal{F} \ne 0$) and thus $\overline{\mathcal{L}}[m, q, a]\,\mathcal{F} = \mathcal{H}_2'\,\mathcal{H}_-^*$.

(uniqueness) Let us start proving that the following claims hold for any $\mathcal{H}_1, \mathcal{H}_2 \in \mathsf{LSeq}$: (*i*) $\mathcal{H}_1\,\mathcal{H}_2^* = 1$ iff $|\mathcal{H}_1| = |\mathcal{H}_2| = 0$; (*ii*) $\langle\mathcal{H}_1\rangle = \langle\mathcal{H}_2\rangle$ iff $\mathcal{H}_1 = \mathcal{H}_2$. For the first claim, let us assume $\mathcal{H}_1 = \mathcal{L}[m, q, a]\,\mathcal{H}'$. For any pair q', a' s.t. $q \ne q'$ or $a \ne a'$, we would get $1 = \overline{\mathcal{L}}[m, q', a']\,\mathcal{H}_1\,\mathcal{H}_2^*\,\mathcal{L}[m, q', a'] = 0$. Hence, $|\mathcal{H}_+| = 0$, etc. For the second claim, let $\mathcal{H}_+\,\mathcal{H}_-^*$ be a canonical form of $\mathcal{H}_1^*\,\mathcal{H}_2$. We have $1 = \mathcal{H}_1^*\,\langle\mathcal{H}_1\,\rangle\,\mathcal{H}_1 = \mathcal{H}_1^*\,\langle\mathcal{H}_2\rangle\,\mathcal{H}_1 = \mathcal{H}_1^*\,\mathcal{H}_2\,(\mathcal{H}_1^*\,\mathcal{H}_2)^* = \mathcal{H}_+\,\mathcal{H}_-^*\,\mathcal{H}_-\,\mathcal{H}_+^* = \mathcal{H}_+\,\mathcal{H}_+^*$. From which, (by the previous claim) $|\mathcal{H}_+| = 0$. In an analogous way we prove that $|\mathcal{H}_-| = 0$ and then that $\mathcal{H}_1^*\,\mathcal{H}_2 = 1$. Hence, $\mathcal{H}_2 = \langle\mathcal{H}_2\rangle\,\mathcal{H}_2 = \langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \mathcal{H}_1$.

Let $\mathcal{H}_+\,\mathcal{H}_-^*$ and $\widehat{\mathcal{H}}_+\,\widehat{\mathcal{H}}_-^*$ be two canonical forms of $\mathcal{F}$. We have $\langle\mathcal{H}_+\rangle = \langle\mathcal{F}\rangle = \langle\widehat{\mathcal{H}}_+\rangle$. Which implies $\mathcal{H}_+ = \widehat{\mathcal{H}}_+$ and $\mathcal{H}_- = \widehat{\mathcal{H}}_-$. Hence, to conclude we just have to show the uniqueness of the canonical form of a lifting sequence $\mathcal{H}$.

The case $\mathcal{H} = \mathsf{1}$ is subsumed by the claim. So, let $\mathcal{L}[m_1, q_1, a_1]\,\mathcal{H}_1 = \mathcal{H} = \mathcal{L}[m_2, q_2, a_2]\,\mathcal{H}_2$ be canonical forms, and w.l.o.g. let $m_1 \leq m_2$. If $m_1 < m_2$, then $\mathcal{H}_1 = \mathcal{L}[m_2 + q_1, q_2, a_2]\,\mathcal{H}_2^{\dagger q_1}\overline{\mathcal{L}}[m_1, q_1, a_1]$, that leads to the contradiction $\mathcal{H}_1\,\mathcal{L}[m_1, q_1', a_1'] = 0$ when $q_1' \neq q_1$ or $a_1' \neq a_1$. Hence, the only possibility is $m_1 = m_2$, and then also $q_1 = q_2, a_1 = a_2$. Concluding, from the initial canonical forms we have got two shorter equivalent ones $\mathcal{H}_1 = \mathcal{H}_2$. □

The elements of LSeq^* can then be written in *canonical form* assuming that both $\mathcal{H}_+$ and $\mathcal{H}_-$ of Proposition 27 are canonical.

Remark 28 A key assumption in the latter proof is $\mathsf{1} \neq \mathsf{0}$. We could then reformulate the proposition saying: The only model of LSeq^* in which Proposition 27 does not hold is the trivial one.

7.6 LSeq *lower semilattice*

The lifting sequences are partially ordered by the binary relation:

$$\mathcal{H}_1 \sqsubseteq \mathcal{H}_2 \quad \textit{when} \quad \mathcal{H}_2 = \mathcal{H}_1\mathcal{H}$$

for some lifting sequence $\mathcal{H} \in \mathsf{LSeq}$. In fact,

- the reflexivity: $\mathcal{H} = \mathcal{H}\,\mathsf{1}$;
- the transitivity: $\mathcal{H}_0 \sqsubseteq \mathcal{H}_1 \sqsubseteq \mathcal{H}_2$ implies $\mathcal{H}_1 = \mathcal{H}_0\,\mathcal{H}$ and $\mathcal{H}_2 = \mathcal{H}_0\,\mathcal{H}\,\mathcal{H}'$;
- the antisymmetry: $\mathcal{H}_2 = \mathcal{H}_1\,\mathcal{H}$ and $\mathcal{H}_1 = \mathcal{H}_2\,\mathcal{H}'$ implies $\mathsf{1} = \mathcal{H}_1^*\,\mathcal{H}_1 = \mathcal{H}_1^*\,\mathcal{H}_2\,\mathcal{H}' = \mathcal{H}_1^*\,\mathcal{H}_1\,\mathcal{H}\,\mathcal{H}' = \mathcal{H}\,\mathcal{H}'$, and thus $\mathcal{H} = \mathcal{H}' = \mathsf{1}$.

Fact 29 *Let* $\mathcal{H}_1, \mathcal{H}_2 \in \mathsf{LSeq}$.

$$\mathcal{H}_1 \sqsubseteq \mathcal{H}_2 \quad \textit{iff} \quad \mathcal{H}_1^*\,\mathcal{H}_2 \in \mathsf{LSeq} \quad \textit{iff} \quad \langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \mathcal{H}_2.$$

Proof.

(i) $\langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \mathcal{H}_2$ *implies* $\mathcal{H}_1^*\,\mathcal{H}_2 \in \mathsf{LSeq}$: Let $\mathcal{H}_+\,\mathcal{H}_-^*$ be the canonical form of $\mathcal{H}_1^*\,\mathcal{H}_2$. By hypothesis, $\mathcal{H}_1\,\mathcal{H}_1^*\,\mathcal{H}_2 = \mathcal{H}_1\,\mathcal{H}_+\,\mathcal{H}_-^* = \mathcal{H}_2$, which implies $\mathcal{H}_-^* = \mathsf{0}$. Thus, $\mathcal{H}_1^*\,\mathcal{H}_2 = \mathcal{H}_+ \in \mathsf{LSeq}$.

(ii) $\mathcal{H}_1^*\,\mathcal{H}_2 \in \mathsf{LSeq}$ *implies* $\mathcal{H}_1 \sqsubseteq \mathcal{H}_2$: Let $\mathcal{H} = \mathcal{H}_1^*\,\mathcal{H}_2$. We have $\mathcal{H}_2 = \mathcal{H}_2\,\mathcal{H}^*\,\mathcal{H} = \langle\mathcal{H}_2\rangle\,\langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \langle\mathcal{H}_1\rangle\,\langle\mathcal{H}_2\rangle\,\mathcal{H}_2 = \langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \mathcal{H}_1\,\mathcal{H}$, and thus $\mathcal{H}_1 \sqsubseteq \mathcal{H}_2$.

(iii) $\mathcal{H}_1 \sqsubseteq \mathcal{H}_2$ *implies* $\langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \mathcal{H}_2$: If $\mathcal{H}_2 = \mathcal{H}_1\mathcal{H}$, then $\langle\mathcal{H}_1\rangle\,\mathcal{H}_2 = \langle\mathcal{H}_1\rangle\,\mathcal{H}_1\,\mathcal{H} = \mathcal{H}_1\,\mathcal{H} = \mathcal{H}_2$. $\square$

The *meet* of two lifting sequences $\mathcal{H}_1$ and $\mathcal{H}_2$ is defined by:

$$\mathcal{H}_1 \sqcap \mathcal{H}_2 = \begin{cases} \mathcal{L}[m,q,a](\mathcal{H}_1' \sqcap \mathcal{H}_2') & \text{if } \mathcal{H}_i = \mathcal{L}[m,q,a]\,\mathcal{H}_i', \text{ for } i = 1,2 \\ 1 & \text{otherwise} \end{cases}$$

By an easy induction on $\mathcal{H}_1$ it is not difficult to check that: the definition of $\sqcap$ is sound; $\mathcal{H}_1 \sqcap \mathcal{H}_2 \sqsubseteq \mathcal{H}_i$, for $i = 1,2$; and that $\mathcal{H} \sqsubseteq \mathcal{H}_i$ for $i = 1,2$, implies $\mathcal{H} \sqsubseteq \mathcal{H}_1 \sqcap \mathcal{H}_2$.

Fact 30 *The sets* LSeq *and* $\mathsf{LSeq}[n_0, n_1]$ *are lower semilattices for the partial order* $\sqsubseteq$ *with* 1 *as minimum and* $\sqcap$ *as greatest-lower-bound operator.*

It is also not difficult to see that $\sqcap$ distributes on the composition of lifting sequences in the following relevant case.

Fact 31 *Let* $\mathcal{H}_0, \mathcal{H}_0' \in \mathsf{LSeq}[n_0, n_1]$ *and* $\mathcal{H}_1, \mathcal{H}_1' \in \mathsf{LSeq}[n_1, n_2]$. *We have that* $\mathcal{H}_1\mathcal{H}_0 \sqcap \mathcal{H}_1'\mathcal{H}_0' = (\mathcal{H}_1 \sqcap \mathcal{H}_1')(\mathcal{H}_0 \sqcap \mathcal{H}_0')$.

7.7 Lifting assignments

Definition 32 (lifting assignment) *A* lifting assignment *for a* $u\ell$*-structure* U *is a map* $\mathscr{A} : \mathsf{vtx}(U) \to \mathsf{LSeq}$ *s.t. (see Figure 16):*

(i) $\mathscr{A}(v) \in \mathsf{LSeq}[0, \ell(v)]$, *for any* $v \in \mathsf{vtx}(U)$;

(ii) $\mathscr{A}(v_i) = \mathcal{S}_e\,\mathscr{A}(v_e)$, *for some* $\mathcal{S}_e \in \mathsf{LSeq}[\ell(v_e), \ell(v_i)]$, *when* v_i *and* v_e *are respectively the internal and the external doors of a box door link* e *(*i.e.*, a ? or an ! link);*

(iii) $\mathscr{A}(v_p) = \mathcal{L}[m,q,a]\,\mathscr{A}(v_a)$, *when* $v_p = \mathsf{pdoor}(e_\triangledown)$ *and* $\{v_a\} = \mathsf{adoors}(e_\triangledown)$ *for a lift* $e_\triangledown$ *s.t.* $m = \ell(e_\triangledown)$, $q = \ell(v_a) - \ell(v_p)$, *and* a *is the name of the auxiliary port of* $e_\triangledown$;

(iv) $\mathscr{A}(v_2) = \mathscr{A}(v_1)$, *when* v_1 *and* v_2 *are doors of the same link* e, *and* e *is neither a box door link nor a lift.*

The second and third items of the previous definition are compatible with the first one (by Fact 26); the fourth item is a special case of the second one, for in such a case $\ell(v_1) = \ell(v_2)$ and $\mathsf{LSeq}[n,n] = \{1\}$. Further, for any box door link e, the parameter $\mathcal{S}_e$ for which $\mathcal{A}(v_i) = \mathcal{S}_e\,\mathcal{A}(v_e)$ is uniquely determined by the lifting sequences assigned to the internal and external doors v_i and v_e (it is indeed determined by $\mathcal{A}(v_i)$ only, see Fact 26).

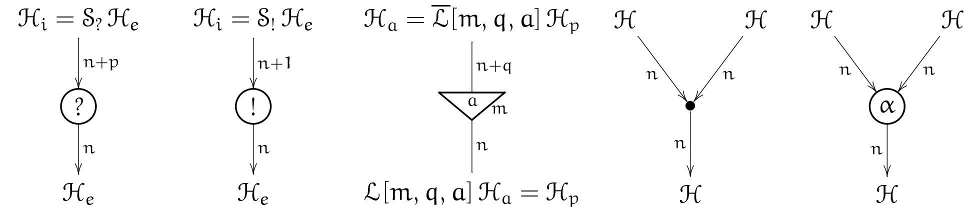
$\mathcal{H}_i = \mathcal{S}_? \mathcal{H}_e$
n+p
?
n
$\mathcal{H}_e$
$\mathcal{H}_i = \mathcal{S}_! \mathcal{H}_e$
n+1
!
n
$\mathcal{H}_e$
$\mathcal{H}_a = \overline{\mathcal{L}}[m, q, a]\, \mathcal{H}_p$
n+q
a
m
n
$\mathcal{L}[m, q, a]\, \mathcal{H}_a = \mathcal{H}_p$
$\mathcal{H}$
n
$\mathcal{H}$
n
n
$\mathcal{H}$
$\mathcal{H}$
n
$\mathcal{H}$
n
α
n
$\mathcal{H}$

Fig. 16. Lifting assignment ($\mathcal{S}_? \in \mathsf{LSeq}[n, n+p]$ and $\mathcal{S}_! \in \mathsf{LSeq}[n, n+1]$).

Remark 33 The constraints of the box door links encode that such links are global boundaries for the scope of the reindexing operators associated to lifts. Moreover, the introduction of the parameters $\mathcal{S}_e$ also corresponds to the idea that we can take the interior of a box and lift it by any quantity, provided that no level inside the box become lower than the level of the box doors. (To this purpose, let us note that, given $\mathcal{H} \in \mathsf{LSeq}[n_0, n_1]$, we have $n_0 \leq n_1 + \|\mathcal{H}\|$ (Fact 26). Hence, interpreting $\|\mathcal{H}\|$ as the global quantity by which we increase the levels above $n_1 - 1$, we see that the previous constraint on the levels is not violated at the box door links.) There is however a remarkable difference between principal and auxiliary door links. The reindexing parameter of an ! link $e_!$—the lifting sequence $\mathcal{S}_{e_!}$ relating the values assigned to the internal and the external doors of $e_!$—is an independent parameter, at least from the interior of $\mathsf{box}^\ell(e_!)$. On the contrary, the reindexing parameter of a ? link is not independent. In fact, a ? link may only erase the reindexing operators introduced by the principal doors of the boxes of which the link is auxiliary door, or introduced by reductions executed inside such boxes. We will farther see that this has a direct correspondence in the dynamics of boxes we will discuss is section 10, and in the read-back reduction system that we will introduce in section 8.

7.8 Solutions of a $\mathcal{u}\ell$*-structure*

As a consequence of Remark 33, we see that, among all the lifting assignments for a $\mathcal{u}\ell$-structure U, we need a way to pick up one of them for each choice of the parameters $\mathcal{S}_!$ associated to the ! links.

In the relevant cases of λ-calculus and MELL the problem has an immediate solution—in the second case, provided that the levels of the structure correspond to the ones induced by the box nesting, see [GMM97a]. In fact, the topology of the corresponding structures forces the uniqueness of the lifting assignment once fixed the parameters $\mathcal{S}_!$. Unfortunately, in the general case this is not true. The solution is to exploit the $\sqcap$ operator to prune away the parts of a lifting assignment for U which are not introduced by lifts of U or by parameters $\mathcal{S}_!$.

A map $\mathscr{S}$ from the ! links of a $\mathcal{u}\ell$-structure U to LSeq is said an *internal state* of U when $\mathscr{S}(\mathsf{e}_!) \in \mathsf{LSeq}[\ell(\mathsf{pdoor}(\mathsf{e}_!)), \ell(\mathsf{pdoor}(\mathsf{e}_! + 1))]$.

By what previously said on the parameters $\mathcal{S}_\mathsf{e}$, any lifting assignment determines a unique internal state of U s.t. $\mathscr{S}(\mathsf{e}) = \mathcal{S}_\mathsf{e}$.

A lifting assignment for U is an *$\mathscr{S}$-assignment* when, for any $\mathsf{e}_! \in \mathsf{lnk}_!(\mathsf{U})$, the constraints of Definition 32 hold with $\mathcal{H}_{\mathsf{e}_!} = \mathscr{S}(\mathsf{e}_!)$.

The partial order relation and the meet operation between lifting sequences extend (pointwise) to lifting assignments:

$$\mathscr{A}_1 \sqsubseteq \mathscr{A}_2 \qquad \textit{iff} \qquad \forall \nu \in \mathsf{vtx}(\mathsf{U}) : \ \mathscr{A}_1(\nu) \sqsubseteq \mathscr{A}_2(\nu)$$

and analogously for the internal states of U, replacing vertices with ! links.

Fact 34 *If $\mathcal{A}_1$ is an $\mathscr{S}_1$-assignment for the $\mathfrak{u}\ell$-structure U and $\mathcal{A}_2$ is an $\mathscr{S}_2$-assignment for U, then $\mathcal{A}_1 \sqcap \mathcal{A}_2$ is an $(\mathscr{S}_1 \sqcap \mathscr{S}_2)$-assignment for U.*

Proof. Let e be an ! link, we have $\mathscr{A}_1(\nu_\mathsf{e}) \sqcap \mathscr{A}_2(\nu_\mathsf{e}) = (\mathscr{S}_1(\mathsf{e}) \sqcap \mathscr{S}_2(\mathsf{e}))\,(\mathscr{A}_1(\nu_\mathsf{e}) \sqcap \mathscr{A}_2(\nu_\mathsf{e}))$ (by Fact 31). And so on for the other kinds of link. $\square$

The latter fact implies that, for any internal state $\mathscr{S}$ with at least an $\mathscr{S}$-assignment, the set $\{\mathscr{A} \mid \mathscr{A}$ is an $\mathscr{S}$-assignment of $\mathsf{U}\}$ is closed under meet and has a minimum.

Definition 35 (solutions) *Let U be a $\mathfrak{u}\ell$-structure with an $\mathscr{S}$-assignment. The minimum $\mathscr{S}$-assignment for U is the $\mathscr{S}$-*solution *of U. In particular, for the quiescence internal state $\mathscr{I}$ (being $\mathscr{I}(\mathsf{e}) = 1$ for any $\mathsf{e} \in \mathsf{lnk}_!(\mathsf{U})$), the $\mathscr{I}$-solution is said the* quiescence solution *of U.*

An example of $\mathfrak{s}\ell$-structure with a quiescence solution is the MELL $\mathfrak{s}\ell$-structure in Figure 17. Near each arrow of the $\mathfrak{s}\ell$-structure we have written (framed) the lifting sequence that the quiescence solution assigns to that arrow. Near each ? link $\mathsf{e}_?$ there is instead the value of the corresponding internal parameter $\mathcal{H}_{\mathsf{e}_?}$—note that such parameters are not equal to 1, even though we are considering a quiescence solution.

7.9 Semantical read-back

The relevance of the solutions of a $\mathfrak{u}\ell$-structure, and in particular of its quiescence solution, is that they allow to remove the lifts from the structure recovering at the same time a sound level assignment (*cf.* section 7.1). In fact, for any $\mathscr{S}$-solution $\mathscr{A}$ of a $\mathfrak{u}\ell$-structure U, it is readily seen that:

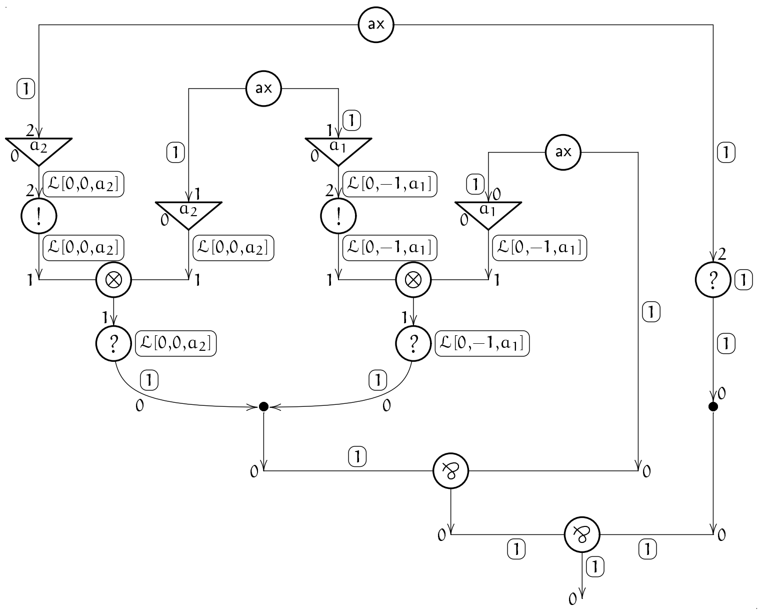

Fig. 17. An example of quiescence solution.

(i) $\ell_{\mathsf{U}}(\nu) + ||\mathscr{A}(\nu)|| \geq 0$, for any $\nu \in \mathsf{vtx}(\mathsf{U})$;
(ii) $\ell_{\mathsf{U}}(\nu_e) + ||\mathscr{A}(\nu_e)|| \leq \ell_{\mathsf{U}}(\nu_i) + ||\mathscr{A}(\nu_i)||$, when ν_i and ν_e are respectively the internal and the external door of a box door link e;
(iii) $\ell_{\mathsf{U}}(\nu_p) + ||\mathscr{A}(\nu_p)|| = \ell_{\mathsf{U}}(\nu_a) + ||\mathscr{A}(\nu_a)||$, when $\nu_p = \mathsf{pdoor}(e_\nabla)$ and $\{\nu_a\} = \mathsf{adoors}(e_\nabla)$ for some lift e_∇.

Definition 36 (read-back) *Let* U *be a* $\mathfrak{u}\ell$*-structure with quiescence solution* $\mathscr{Q}$*. Its* read-back $\mathscr{R}(\mathsf{U})$ *is the* ℓ*-structure obtainable from* U*:*

(i) associating to each arrow ν *its* actual level $\ell_{\mathsf{U}}^{\mathscr{I}}(\nu) = \ell_{\mathsf{U}}(\nu) + ||\mathscr{Q}(\nu)||$*;*
(ii) replacing each lift e_∇ *and its doors* ν_p *and* ν_a *by a unique arrow with level* $\ell_{\mathsf{U}}^{\mathscr{I}}(\nu_p) = \ell_{\mathsf{U}}^{\mathscr{I}}(\nu_a)$ *connecting the ports to which* ν_p *and* ν_a *were connected.*

The ℓ-structure in Figure 18 is the read-back of the $\mathfrak{u}\ell$-structure in Figure 17.

7.10 Proper structures

Using the solutions of a $\mathfrak{u}\ell$-structure we are now able to characterize the “proper” unfoldings of $\mathsf{s}\ell$-structures.

Definition 37 (complete unfolding) *A triple* $\mathsf{M} : \mathsf{U} \prec\!\!\!\prec \mathsf{G}$ *is a* complete unfolding *of the* $\mathsf{s}\ell$*-structure* G*, and* U *is a* least-shared-instance *of* G*, when:*

(i) $\mathsf{M} : \mathsf{U} \preccurlyeq \mathsf{G}$*;*
(ii) U *has an* $\mathscr{S}$*-solution for any internal state* $\mathscr{S}$*;*
(iii) if $\mathscr{A}$ *is a solution of* U*, then* $\mathsf{M}(\nu) = \mathsf{M}(\nu')$ *and* $\mathscr{A}(\nu) = \mathscr{A}(\nu')$ *implies*

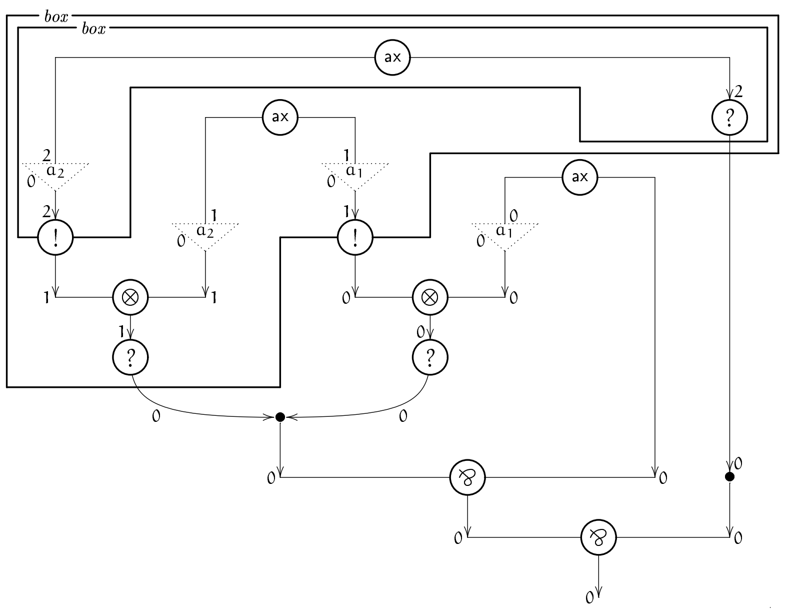

Fig. 18. Read-back.

$\nu = \nu'$, *for any* $\nu, \nu' \in \mathsf{vtx}(\mathrm{U}) \setminus \bigcup_{e_Y \in \mathsf{lnk}_Y(\mathrm{U})} \mathsf{adoors}(e_Y)$.

Remark 38 The reasons because of which in item (iii) we exclude $\nu, \nu' \in \bigcup_{e_Y \in \mathsf{lnk}_Y(\mathrm{U})} \mathsf{adoors}(e_Y)$ are merely technical. The distinction between two contracting arrows s.t. $\mathscr{A}(\nu) = \mathscr{A}(\nu')$ and $\mathrm{M}(\nu) = \mathrm{M}(\nu')$ is ensured by the fact that the same property cannot hold for the internal doors of the ? links above them.

Remark 39 To require the existence of a solution for any internal state accords with the idea that the reindexing parameter $\mathrm{H}_!$ of the ! lifts is an independent parameter.

Definition 40 (proper sℓ-structure) *An* sℓ*-structure is* proper *when it has a complete unfolding.*

Remark 41 According to Definition 37 the partial order $\lessdot\!\!\prec$ is flat: a proper ℓ-structure is related only with its least-shared-instance (later we will prove that the least-shared-instance of a proper sℓ-structure is unique). However, the relation $\lessdot\!\!\prec$ could be extended defining $\mathrm{G}_1 \lessdot\!\!\prec \mathrm{G}_2$, say G_1 is a *proper unfolding* of G_2, when G_1 and G_2 are proper sℓ-structures with the same least-shared-instance and $\mathrm{G}_1 \preccurlyeq \mathrm{G}_2$. Besides, in our study we will use only the case in which G_1 is unshared. For this reason, in the following we will use $\mathrm{U} \lessdot\!\!\prec \mathrm{G}$ to denote that U is the least-shared-instance of G.

Any uℓ-structure with an $\mathscr{S}$-solution for any internal state $\mathscr{S}$ is proper, *e.g.*, the uℓ-structure in Figure 17, and the identity s-morphism is its complete unfolding. An example of proper sℓ-structure is instead drawn in Figure 19, its least-shared-instance is the uℓ-structure in Figure 17.

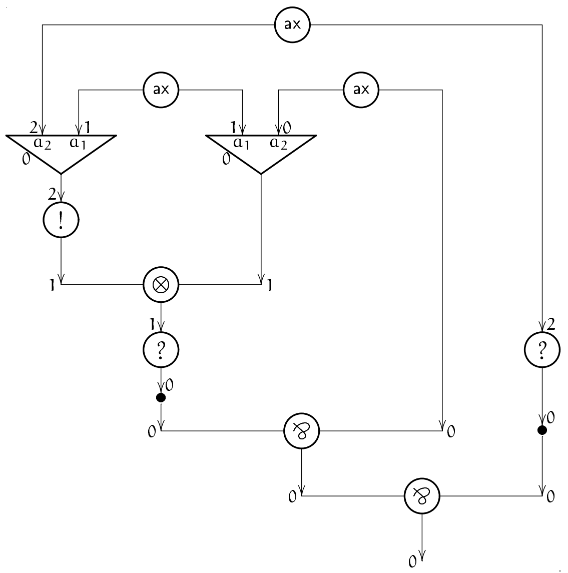

Fig. 19. A proper $s\ell$-structure.

The first step toward the acknowledgment of $\prec\!\!\!\prec$ as the proper unfolding partial order is the proof that a proper $s\ell$-structure G has a unique least-shared-instance. Such a result is direct when G is an ℓ-structure. The proof for the general case (see Proposition 50) rests instead on the read-back reduction system we are going to introduce in section 8.

8 Read-back reductions

Let G be a proper $s\ell$-structure whose complete unfolding is unique. The natural way to associate an ℓ-structure to G is defining the read-back of G as the read-back of its least-shared-instance.

Definition 42 (read-back) *Let* G *be a proper* $s\ell$*-structure with a unique* $\mathsf{U} \prec\!\!\!\prec \mathsf{G}$*. The* read-back *of* G *is the* ℓ*-structure* $\mathscr{R}(\mathsf{G}) = \mathscr{R}(\mathsf{U})$.

8.1 π rules

The test bed for the latter definition of semantical read-back is the *read-back rewriting system*, or π rules, of Figure 20 (all the muxes in the picture are binary since the extension to the k-ary case is direct). Such rules implement the interpretation of lifts that guided us so far, taking into account the additional duplicating task that k-muxes have w.r.t. lifts. We recognize three sets of rules:

(i) the *propagations* or duplications (at the top in Figure 20), in which α stands for a generic type;

(ii) the *absorption* (at the center in Figure 20);
(iii) the *mux rules* (at the bottom in Figure 20).

Note that the stem of the arrows have been omitted in the propagation and mux rules of Figure 20. The reason is that the drawings for such rules are indeed schemata valid independently from the orientation of the arrows (*e.g.*, in the !/?-propagation the principal door of the mux might be the internal door of the box door link).

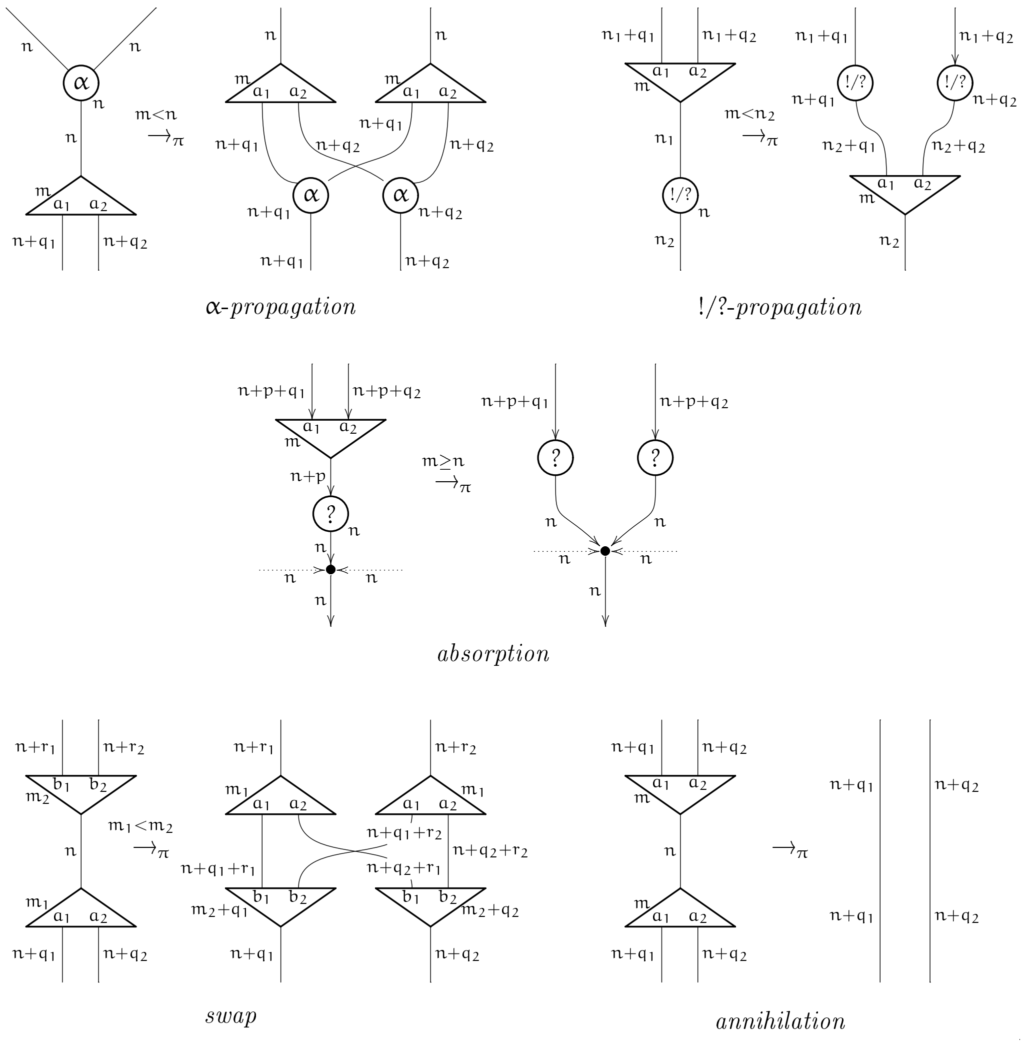

Fig. 20. Read-back rewriting system (π rules).

The *propagation* rules correspond to the idea that a mux has to reindex and duplicate every link it founds at a level greater than its threshold—see the side-condition of the rules. (The proviso $m < n$ in the α-propagation rule is actually redundant; by definition, $\ell(e_\triangledown) \leq \ell(\mathsf{pdoor}(e_\triangledown))$ for any mux $e_\triangledown$. Nevertheless, to stress that in an α-propagation the mux is acting on a link whose level is above its threshold, we prefered to restate it adding the proviso.)

The proviso on the propagation rules forbids their application in the case the principal door of a mux with threshold m is the internal door of a box door link whose level is lower or equal than m. The *absorption* rule covers such a case when the box door link $e_?$ is a ? link—it corresponds to the idea that $e_?$ is a boundary to the reindexing operators originated inside the boxes that $e_?$ closes. There is instead no rule for the case in which the box door link $e_!$ is an ! link—it corresponds to the fact that the reindexing (and duplication) that $e_!$ may impose do not depend on the contents of its box (see also Remark 33). Hence, the missing ! link case has to be regarded as an unlikely situation—it is not incidentally that this configuration cannot arise in a proper $s\ell$-structure, see Fact 47.

The two mux π rules implement the idea that mux thresholds are a machinery to correctly match muxes. The *swap* rule corresponds to the fact that two muxes with different thresholds denote independent reindexing-duplication operators. Therefore, the interaction of a pair of them must preserve all the I/O connections of the pair. At the same time, the interaction must properly lift the threshold m_2 of the higher mux; since a mux with threshold m_1 lifts all the levels above it, and then $m_2 > m_1$ too. (We stress that, generalizing the rule to the k-ary case, the interacting muxes may have different cardinalities.) The *annihilation* rule corresponds instead to the matching property described in section 5.2: the only legal paths crossing a pair of facing muxes with the same threshold connect ports with the same name and offset.

The first property to check is that any proper $s\ell$-structure G is rewritten by π reduction into a proper $s\ell$-structure. This will be done in Lemma 44 showing at the same time that any π reduction of a proper $s\ell$-structure can be simulated on its least-shared-instance. But for this purpose we need the invariance of properness at least when G is a $u\ell$-structure.

Lemma 43 *If $U \to_\pi U'$, then U is a proper $u\ell$-structure iff U' is a proper $u\ell$-structure.*

Proof. Let $r : U \to_\pi U'$. We see that when r is any π rule but an !-propagation, the $u\ell$-structure U has an $\mathscr{S}$-solution iff U' has a corresponding $\mathscr{S}$-solution. Hence, let $r = (e_\triangledown, e_!)$ be an !-propagation s.t. q is the offset of $e_\triangledown$ (assume that a π redex r is a pair $(e_\triangledown, e)$, where $e_\triangledown$ is the mux of r and e is the link s.t $\mathsf{pdoor}(e_\triangledown) \in \mathsf{doors}(e)$). In this case, U has an $\mathscr{S}$-solution iff U' has a corresponding $\mathscr{S}'$-solution, being $\mathscr{S}'$ the internal state s.t. $\mathscr{S}'(e_!) = \mathscr{S}(e_!)^{\uparrow q}$ and $\mathscr{S}'(e) = \mathscr{S}(e)$ otherwise. $\square$

The key property of the π rules is the possibility to simulate any π reduction of a proper $s\ell$-structure by a π reduction of its least-shared-instance.

Lemma 44 *Let* G_0 *be a proper* $s\ell$*-structure and let* $\mathsf{U}_0 \prec\!\!\prec \mathsf{G}_0$*. For any* $\mathsf{G}_0 \rightarrow_\pi \mathsf{G}_1$*, there exists* $\mathsf{U}_0 \rightarrow_\pi^+ \mathsf{U}_1$ *s.t.* $\mathsf{U}_1 \prec\!\!\prec \mathsf{G}_1$*.*

Proof. Let $\mathsf{M}_0 : \mathsf{U}_0 \prec\!\!\prec \mathsf{G}_0$ and let r be a redex of G_0. The counterimage of r is a set of redexes $\mathsf{M}_0^{-1}(r)$ that may contain only a case of critical pair: two (or more) lifts whose principal doors are contracting doors of the same $\curlyvee$ link. When r is an absorption, any of such critical pairs is trivially confluent. When r is a ?-propagation, all the lifts in the critical pairs are equal, since U_0 is proper. Thus, it is readily seen that in this case too all the critical pairs are confluent. Hence, let us execute in any order the redexes in the set $\mathsf{M}_0^{-1}(r)$, closing the critical pairs present in it. The result is the proper $u\ell$-structure U_1 (by Lemma 43, U_1 is definitely proper). The s-morphism between $\mathsf{M}_1 : \mathsf{U}_1 \prec\!\!\prec \mathsf{G}_1$ is the one induced by the function mapping any residual of $e \in \mathsf{lnk}(\mathsf{U}_0)$ into the corresponding residual of $\mathsf{M}_0(e)$. $\square$

Finally, we can show that the definitions of proper $s\ell$-structure and read-back are sound.

Proposition 45 *Let* G *be a proper* $s\ell$*-structure. For any read-back reduction* $\mathsf{G} \rightarrow_\pi \mathsf{G}'$*, we have that:*

(i) G' *is proper;*
(ii) G *has a unique* $\mathsf{U} \prec\!\!\prec \mathsf{G}$ *iff there is a unique* $\mathsf{U}' \prec\!\!\prec \mathsf{G}'$*; in which case*
(iii) $\mathscr{R}(\mathsf{G}) = \mathscr{R}(\mathsf{G}')$*.*

Proof.

(i) It is an immediate consequence of Lemma 43 and Lemma 44.
(ii) (*if*) Let $\mathsf{U}_1 \prec\!\!\prec \mathsf{G} \succ\!\!\succ \mathsf{U}_2$. By Lemma 44, $\mathsf{U}_i \rightarrow_\pi^+ \mathsf{U}'$ by a sequence of reductions corresponding to $\mathsf{G} \rightarrow_\pi \mathsf{G}'$. By inspection of the rules, we see that this is possible only if $\mathsf{U}_1 = \mathsf{U}_2$. (*only if*) Revert the arrows of the π rules and proceed as in the previous case.
(iii) By inspection of the π rules, we see that $\mathscr{R}(\mathsf{U}) = \mathscr{R}(\mathsf{U}')$. $\square$

It is worth to note that the invariance of properness under π reduction holds for the $s\ell$-structures too. Such an invariance will be farther exploited to prove that an $s\ell$-structure is proper iff it normalizes to a proper ℓ-structure.

Lemma 46 *If* $\mathsf{G} \rightarrow_\pi \mathsf{G}'$*, then* G *is a proper* $s\ell$*-structure iff* G' *is a proper* $s\ell$*-structure.*

Proof. The if part has been already proved. By Lemma 43, the only if part holds when G is a $\mathfrak{u}\ell$-structure. To lift the result to the $\mathfrak{s}\ell$-structures, let us note that even the simulation property can be reversed. Namely, if $\mathsf{G} \to_\pi \mathsf{G}'$ and $\mathsf{U}' \prec\!\!\!\prec \mathsf{G}'$, then there is $\mathsf{U} \prec\!\!\!\prec \mathsf{G}$ s.t. $\mathsf{U} \to_\pi^+ \mathsf{U}'$. $\square$

8.2 Local confluence of the π rules

Unfortunately, the π rules are not locally confluent in general. Nevertheless, the π rules are strongly normalizing and confluent on proper $\mathfrak{s}\ell$-structures (see Proposition 49). To prove such a result let us start proving that local confluence holds, modulo some mux permutations, for the $\mathfrak{s}\ell$-structures in which each mux eventually interacts with some other link (Lemma 48).

Let $\mathsf{e}_\triangledown$ be a mux whose principal door is not auxiliary door of another mux. We say that $\mathsf{e}_\triangledown$ forms a *deadlock* when no π rule can be applied to it.

Fact 47 (deadlock-freeness) *An* $\mathfrak{s}\ell$*-structure* G *is* deadlock-free *when there is no* $\mathsf{G} \to_\pi^* \mathsf{G}'$ *s.t.* G' *contains a deadlock. Any proper* $\mathfrak{s}\ell$*-structure is deadlock-free.*

Proof. Let $\mathsf{U} \prec\!\!\!\prec \mathsf{G}$. If U would contain a deadlock, the constraints of Definition 32 would be unsatisfiable for at least the quiescence internal state. Thus, there is no deadlock in U. It is indeed easy to verify that the presence of a deadlock in G would imply the presence of a deadlock in U. Thus, there is no deadlock in G, and more generally there is no deadlock in any proper $\mathfrak{s}\ell$-structure. $\square$

The read-back reduction system has several critical pairs. They can be classified under two patterns: absorption-propagation and propagation-propagation.

The absorption-propagation critical pairs do not cause any problem: they are confluent for any value of the thresholds, offsets, and port names.

The propagation-propagation pairs cause instead the loss of the locally confluence. Nevertheless, let us note that when the $\mathfrak{s}\ell$-structure is proper:

(i) If the pair is formed of two muxes with the same threshold whose principal doors are doors of the same link, then the two muxes have the same set of port offsets and names, and the critical pair is confluent (for a proper $\mathfrak{s}\ell$-structure is deadlock-free).
(ii) If the muxes have different thresholds, the result depends on the order in which the two propagation redexes r_1 and r_2 of G are contracted. But,

there are two reductions $\mathsf{G} \overset{r_1}{\rightarrow}_\pi \rightarrow_\pi^+ \mathsf{G}_1$ and $\mathsf{G} \overset{r_2}{\rightarrow}_\pi \rightarrow_\pi^+ \mathsf{G}_2$ s.t. G_1 and G_2 coincide modulo the permutation of muxes in Figure 21.

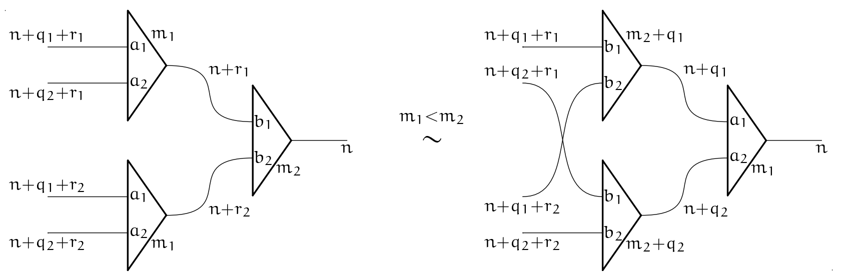

Fig. 21. Mux permutation equivalence.

The *mux permutation equivalence* is the symmetric, transitive and reflexive closure $\sim$ of the rule in Figure 21.

The equivalence relation $\sim$ is sound w.r.t. to properness and unfolding. In fact, if $\mathsf{G}_1 \sim \mathsf{G}_2$, then $\mathsf{U}_1 \sim \mathsf{U}_2$, where $\mathsf{U}_i \prec\!\!\!\prec \mathsf{G}_i$ for $i = 1, 2$.

Lemma 48 *The proper* $s\ell$*-structures modulo* $\sim$ *are locally confluent under* π *reduction. Namely, the diagram*

$$\begin{tikzcd}
& \mathsf{G}_0 \arrow[r, phantom, "\sim"] \arrow[dl, "\pi"'] & \mathsf{H}_0 \arrow[dr, "\pi"] & \\
\mathsf{G}_1 \arrow[dr, "\pi", "*"'] & & & \mathsf{H}_1 \arrow[dl, "\pi"', "*"] \\
& \mathsf{G}_2 \arrow[r, phantom, "\sim"] & \mathsf{H}_2 &
\end{tikzcd}$$

commutes for any pair of proper $s\ell$*-structures* $\mathsf{G}_0, \mathsf{H}_0$*, and for any pair of* π *reductions* $\mathsf{G}_0 \rightarrow_\pi \mathsf{G}_1$*,* $\mathsf{H}_0 \rightarrow_\pi \mathsf{H}_1$*.*

Sketch of the proof. The complete proof is a tedious analysis of all the possible configurations. The key idea is that G_0 and H_0 are equal in all respects apart for some substructures $\mathsf{T}[\mathsf{G}_0] \sim \mathsf{T}[\mathsf{H}_0]$ composed of muxes only, that w.l.o.g. we may assume to be trees. When $r : \mathsf{G}_0 \rightarrow_\pi \mathsf{G}_1$ and $s : \mathsf{H}_0 \rightarrow_\pi \mathsf{H}_1$ does not involve the root mux of any of such trees, the lemma follows from the already remarked property that the diagram commutes when $\mathsf{G}_0 = \mathsf{H}_0$. In the other cases, let $\mathsf{T}[\mathsf{G}_0]$ (or mutatis mutandis $\mathsf{T}[\mathsf{H}_0]$) be the tree whose root mux form the redex r with a link e. There is a reduction $\mathsf{R} : \mathsf{G}_0 \rightarrow_\pi^* \mathsf{G}'$ s.t. all the muxes of $\mathsf{T}[\mathsf{G}_0]$ have interacted with a residual of e (assume that, when a residual of e annihilates interacting with a mux e' of $\mathsf{T}[\mathsf{G}_0]$, the interaction is complete for all the muxes of $\mathsf{T}[\mathsf{G}_0]$ following e' too). Since r and s are the first reductions of the corresponding sequences R and S, the property is proved showing that the diagram commutes replacing R and S for r and s. $\square$

We stress that the previous lemma could be reformulated replacing proper

$s\ell$-structures with deadlock-free $s\ell$-structures, since deadlock-freeness is the only property of the proper $s\ell$-structures that has been used in the proof.

8.3 Existence of the read-back

The next two propositions show that the algebraic semantics is indeed a way to talk about the π rules.

Proposition 49 *Let* G *be a proper* $s\ell$*-structure.*

(i) There is no infinite π *reduction of* G.
(ii) G *has a unique* π *normal form.*

Proof.

(i) Let $\mathscr{Q}$ be the quiescence solution of a proper $u\ell$-structure U. Let us take:
(a) $k_1 = \sum_{e \in L} \sum_{v \in \mathsf{doors}(e)} \|\mathscr{Q}(v)\|$, where $L = \mathsf{lnk}(\mathsf{U}) \setminus \mathsf{lnk}_{\nabla}(\mathsf{U})$;
(b) $k_2 = \sum_{e \in \mathsf{lnk}_{\nabla}(\mathsf{U})} \|\mathscr{Q}(\mathsf{pdoor}(e))\|$.
Any propagation or absorption rule decreases k_1, but may increase k_2. Any mux rule decreases k_2. Therefore, any π rule decreases the pair (k_1, k_2) (w.r.t. the lexicographic order), and because of this there is no infinite π reduction of U. To conclude, let us take $\mathsf{U} \prec\!\!\!\prec \mathsf{G}$, it follows by Lemma 44 that there is no infinite π reduction of G (note that a single π reduction of G is simulated by a non-empty sequence of π reductions of U).

(ii) The uniqueness of the normal form up to $\sim$ follows by the previous item, Lemma 48, and Newman's Lemma. Furthermore, a π normal form N of G cannot contain muxes, for G is deadlock-free. Thus, N is unique. $\square$

Proposition 50 *Any proper* $s\ell$*-structure* G *has a unique least-shared-instance and* $\mathscr{R}(\mathsf{G})$ *is the unique* π *normal form of* G.

Proof. The unique least-shared-instance of a proper ℓ-structure N is N itself: the quiescence solution of N associates 1 to each arrow of N; thus, $\mathsf{M}(v) = \mathsf{M}(v')$ iff $v = v'$ for any $\mathsf{M} : \mathsf{N} \prec\!\!\!\prec \mathsf{G}$, etc. As a consequence, G has a unique least-shared-instance (by Proposition 45) and $\mathscr{R}(\mathsf{G})$ is defined. By the invariance of the read-back under π reduction (Proposition 45) we can then conclude that $\mathsf{N} = \mathscr{R}(\mathsf{N}) = \mathscr{R}(\mathsf{G})$. $\square$

9 Properness and box nesting property

We will now see that in order to get a proper structure the box nesting property is a necessary requirement. Moreover, we will see that properness is indeed an implicit characterization of the box nesting property.

Proposition 51 *An ℓ-structure is proper iff it has the box nesting property.*

Proof.

(if) Let $\mathscr{S}$ be an internal state of U. Let us take the sequence of internal states $\mathscr{I} = \mathscr{S}_0, \mathscr{S}_1, \ldots, \mathscr{S}_k = \mathscr{S}$ defined in this way: $\mathscr{S}_i(e_!) = \mathscr{S}(e_!)$ if $\ell_\mathsf{U}(e_!) < i$ and $\mathscr{S}_i(e_!) = 1$ otherwise. Let $\mathscr{A}_0$ be the quiescence solution of U. For $i \geq 0$, we inductively define $\mathscr{A}_{i+1}(v) = \mathscr{S}(e_!)\,\mathscr{A}_i(v)$ when $v \in \mathsf{box}^\ell_\mathsf{U}(e_!)$ for some $e_! \in \mathsf{lnk}_!(\mathsf{U})$ s.t. $\ell_\mathsf{U}(e_!) = i$, and $\mathscr{A}_{i+1}(v) = \mathscr{A}_i(v)$ otherwise. The previous definition is sound because of the box nesting property, and gives a sequence of assignments $\mathscr{A}_i$ which are indeed the $\mathscr{S}_i$-solution of U.

(only if) Let $e_1, e_2 \in \mathsf{lnk}_!(\mathsf{U})$ be s.t. $\mathsf{box}^\ell(e_1) \cap \mathsf{box}^\ell(e_2) \neq \varnothing$. W.l.o.g, let $\ell_\mathsf{U}(e_1) \leq \ell_\mathsf{U}(e_2)$. By the definition of box^ℓ, we see that $v \in \mathsf{vtx}(\mathsf{box}^\ell_\mathsf{U}(e_2))$ implies $v \in \mathsf{vtx}(\mathsf{box}^\ell_\mathsf{U}(e_1))$. Hence, $\mathsf{box}^\ell_\mathsf{U}(e_2) \subseteq \mathsf{box}^\ell_\mathsf{U}(e_1)$. When $\ell_\mathsf{U}(e_1) = \ell_\mathsf{U}(e_2)$, let $\mathscr{S}_{12}$ be an internal state of U s.t. $\mathscr{S}_{12}(e) \neq 1$ iff $e \in \{e_1, e_2\}$. We see that $\mathsf{box}^\ell_\mathsf{U}(e_1) = \mathsf{box}^\ell_\mathsf{U}(e_2)$, since $e_2 \in \mathsf{dlinks}(\mathsf{box}^\ell_\mathsf{U}(e_1))$. But in this case, U has an $\mathscr{S}_{12}$-solution iff $\mathscr{S}_{12}(e_1) = \mathscr{S}_{12}(e_2)$. Thus, $e_1 = e_2$. When $\ell_\mathsf{U}(e_1) < \ell_\mathsf{U}(e_2)$, we have that $e_2 \in \mathsf{box}^\ell_\mathsf{U}(e_1) \setminus \mathsf{dlinks}(\mathsf{box}^\ell_\mathsf{U}(e_1))$. Thus, $\mathsf{box}^\ell_\mathsf{U}(e_1) \subset \mathsf{box}^\ell_\mathsf{U}(e_2)$. $\square$

The maps $\mathsf{box}^\ell_\mathsf{U}$ and $\mathsf{BX}^\ell_\mathsf{U}$ can be extended to the case in which U is a proper $u\ell$-structure replacing the actual level function $\ell^{\mathscr{I}}_\mathsf{U}$ for ℓ_U in Definition 11 (this is sound even if U is an ℓ-structure, as in such a case $\ell^{\mathscr{I}}_\mathsf{U} = \ell_\mathsf{U}$). Hence, we have a notion of box nesting property for the $u\ell$-structures too. A relevant consequence of the results on the π reductions is that properness and box nesting property are equivalent also in this case (note that to define $\ell^{\mathscr{I}}_\mathsf{U}$ and then $\mathscr{R}(\mathsf{U})$ we suffice the existence of the quiescence solution of U).

Proposition 52 *A $u\ell$-structure is proper iff it has a quiescence solution and the box nesting property.*

Proof.

(only if) By definition any proper $u\ell$-structure U has a quiescence solution. The rest of the proof is by induction on the length of a normalizing π reduction of U. The base case is proved by Proposition 51. For the induction step, let $\mathsf{U} \to_\pi \mathsf{U}'$. There is a one-to-one correspondence f between the links of U which are not lifts and the links of U' which are not lifts. By inspection of the π rules, we see that $\mathsf{f}(e) \in \mathsf{box}^\ell_{\mathsf{U}'}(\mathsf{f}(e_!))$ iff $e \in \mathsf{box}^\ell_{\mathsf{U}}(e_!)$, for any $e \in \mathsf{lnk}(\mathsf{U}) \setminus \mathsf{lnk}_\nabla(\mathsf{U})$ and any $e_! \in \mathsf{lnk}_!(\mathsf{U})$. Thus, U has the box nesting property iff U' has the box nesting property.

(if) By inspection of the proof of Lemma 49, we see that the existence of a quiescence solution suffices to prove that the π rules are strongly normalizing and that the unique π normal form of U is $\mathscr{R}(\mathsf{U})$. Proving the only if part, we have already seen that the box nesting property is invariant under π reduction. Thus, $\mathscr{R}(\mathsf{U})$ has the box nesting property, and by Proposition 51 it is proper. That U is proper follows by the invariance of properness under π reduction (Lemma 46). $\square$

Proposition 53 *An $s\ell$-structure G is proper iff its π normal form is an ℓ-structure for which the box nesting property holds.*

Proof. By the invariance of properness (Lemma 46), an $s\ell$-structure is proper iff its π normal form N is. Since all the proper $s\ell$-structures are deadlock-free (Fact 47), the proper ℓ-structures are the only proper $s\ell$-structures in π normal form. By Proposition 51, we conclude that G is proper iff N has the box nesting property. $\square$

The latter propositions have two relevant consequences that will be used in the following. The first one is that, for any internal state of a $u\ell$-structure U and any $e_! \in \mathsf{lnk}_!(\mathsf{U})$, the level its solution $\mathscr{A}$ assigns to $\mathsf{idoor}(e_!)$ is lower or equal than the level that $\mathscr{A}$ assigns to any arrow contained in $\mathsf{box}^\ell_{\mathsf{U}}(e_!)$ (as for the actual levels induced by the quiescence solution, we say that $\mathscr{A}$ assigns the level $\ell^{\mathscr{A}}_{\mathsf{U}} = \ell_{\mathsf{U}}(\nu) + \|\mathscr{A}(\nu)\|$ to the arrow ν), and is strictly greater when we assume that the internal state does not force a decreasing of -1.

Lemma 54 *Let U be a proper $u\ell$-structure. For any $\nu \in \mathsf{vtx}(\mathsf{U})$ and any $e \in \mathsf{lnk}_!(\mathsf{U})$, we have that:*

(i) $\nu \in \mathsf{box}^\ell_{\mathsf{U}}(e)$ *iff ν is connected to e and* $\ell^{\mathscr{A}}_{\mathsf{U}}(\nu) \geq \ell^{\mathscr{A}}_{\mathsf{U}}(\mathsf{idoor}(e))$*, for any $\mathscr{S}$-solution $\mathscr{A}$;*

(ii) $\nu \in \mathsf{box}^\ell_{\mathsf{U}}(e)$ *iff ν is connected to e and* $\ell^{\mathscr{A}}_{\mathsf{U}}(\nu) > \ell^{\mathscr{A}}_{\mathsf{U}}(\mathsf{idoor}(e))$ *for any $\mathscr{S}$-solution $\mathscr{A}$ s.t. $\|\mathscr{S}(e)\| \geq 0$.*

Proof. By induction on the length of a normalizing π reduction. The statement of the lemma is invariant under π reduction (see the proof of Lemma 43). The if part of the proof of Proposition 51 gives a way to build a generic solution of an ℓ-structure. By inspection of this construction we see that the lemma holds for any ℓ-structure. $\square$

The second consequence is that any $\mathscr{S}$-solution of a $\mathsf{u}\ell$-structure U can be built starting from the quiescence solution of U, following a procedure similar to the one used for the ℓ-structures in the proof of Proposition 51.

Lemma 55 *Let* A *be an* $\mathscr{S}$*-solution of a proper* $\mathsf{u}\ell$*-structure* U*. For any* $\mathsf{e}_0 \in \mathsf{lnk}_!(\mathsf{U})$*, we have that*

$$\mathscr{A}(\nu) = \begin{cases} \mathscr{A}_0(\nu)\,\mathscr{S}(\mathsf{e}_0)^{\uparrow \mathsf{q}_0} & \textit{when } \nu \in \mathsf{vtx}(\mathsf{box}^{\ell}_{\mathsf{U}}(\mathsf{e}_0)) \setminus \mathsf{doors}(\mathsf{box}^{\ell}_{\mathsf{U}}(\mathsf{e}_0)) \\ \mathscr{A}_0(\nu) & \textit{otherwise} \end{cases}$$

where $\mathsf{q}_0 = \|\mathscr{A}_0(\mathsf{edoor}(\mathsf{e}_0))\|$ *and* $\mathscr{A}_0$ *is the* $\mathscr{S}_0$*-solution of* U*, being*

$$\mathscr{S}_0(\mathsf{e}) = \begin{cases} 1 & \textit{when } \mathsf{e} = \mathsf{e}_0 \\ \mathscr{S}(\mathsf{e}) & \textit{otherwise} \end{cases}.$$

Proof. Let $\mathscr{A}_0$ be the $\mathscr{S}_0$-solution. We start proving that the $\mathscr{A}$ defined in the statement is an $\mathscr{S}$-assignment. The lifting assignment constraints hold by hypothesis for $\mathsf{U} \setminus \mathsf{box}^{\ell}_{\mathsf{U}}(\mathsf{e}_0)$. Furthermore, for any $\nu \in \mathsf{vtx}(\mathsf{box}^{\ell}_{\mathsf{U}}(\mathsf{e}_0))$, we have that $\mathscr{A}(\nu) = \mathscr{A}_0(\nu)\,\mathscr{S}(\mathsf{e}_0)^{\uparrow \mathsf{q}_0} \in \mathsf{LSeq}[0, \ell_{\mathsf{U}}(\nu)]$, since $\ell_{\mathsf{U}}(\nu) + \|\mathscr{A}_0(\nu)\| \geq \ell_{\mathsf{U}}(\mathsf{e}_0) + 1 + \mathsf{q}_0$ (by Lemma 54) and $\mathscr{S}(\mathsf{e}_0) \in \mathsf{LSeq}[\ell_{\mathsf{U}}(\mathsf{e}_0), \ell_{\mathsf{U}}(\mathsf{e}_0)+1]$. So, we left to verify if the lifting assignment constraints hold for any $\mathsf{e} \in \mathsf{box}^{\ell}_{\mathsf{U}}(\mathsf{e}_0)$. The only relevant case is $\mathsf{e} \in \mathsf{adoors}(\mathsf{box}^{\ell}_{\mathsf{U}}(\mathsf{e}_0))$. Let $\nu_i = \mathsf{idoor}(\mathsf{e})$ and $\nu_e = \mathsf{edoor}(\mathsf{e})$. By Lemma 54, we know that $\ell_{\mathsf{U}}(\nu_e) + \|\mathscr{A}_0(\nu_e)\| \leq \ell(\mathsf{e}_0) + 1 + \mathsf{q}_0$. Thus, $\mathscr{A}_0(\nu_i)\,\mathscr{S}(\mathsf{e}_0)^{\uparrow \mathsf{q}_0} = \mathcal{H}\,\mathscr{S}(\mathsf{e}_0)^{\uparrow \mathsf{q}_0 + \|\mathscr{A}_0(\nu_e)\|}\,\mathscr{A}_0(\nu_e)$, with $\mathcal{H}\,\mathscr{S}(\mathsf{e}_0)^{\uparrow \mathsf{q}_0 + \|\mathscr{A}_0(\nu_e)\|} \in \mathsf{LSeq}[\ell_{\mathsf{U}}(\nu_e), \ell_{\mathsf{U}}(\nu_i)]$. It is then easy to check that if $\mathscr{A}$ would not be the $\mathscr{S}$-solution of U, then $\mathscr{A}_0$ would not be the $\mathscr{S}_0$-solution of U. □

10 Sharing graph machine

We are now ready to prove that the $\mathsf{s}\ell$-structures plus the π rules are the abstract machine by which to obtain sharing implementations of calculi whose rewriting rules are similar to the β-rule of the λ-calculus or to the exponential cut-elimination rule of linear logic.

10.1 Box reductions

An example of the box reduction rules we want to implement is drawn in Figure 22. The example depicts the (pure) proof net case. But, independently from the calculus under analysis, there is a general pattern characterizing the global or β rewriting rules for which we can give a sharing implementation.

(i) In a β rule, the principal door link $e_!$ of a box B_1 interacts with several ? box door links $e_?^1, e_?^2, \ldots$ which may either be auxiliary doors of a box B_2 or not, respectively when $\ell(\mathsf{idoor}(e_?)) > \ell(\mathsf{edoor}(e_?))$ or $\ell(\mathsf{idoor}(e_?)) = \ell(\mathsf{edoor}(e_?))$.
(ii) None of the ? links $e_?^i$ is auxiliary door of $\mathsf{box}^\ell(e_!)$.
(iii) The result of the interaction is a new structure in which:
  (a) there is an instance B_1^i of B_1 for each $e_?^i$;
  (b) the box around each B_1^i is removed (*i.e.*, its ! link is erased) and B_1^i is pushed inside the boxes of which $e_?^i$ is auxiliary door;
  (c) each $e_?^i$ is erased and its internal door is connected to $\mathsf{pdoor}(e_!)^i$ (*i.e.*, the copy of $\mathsf{pdoor}(e_!)$ in B_1^i);
  (d) each $\curlyvee$ link $e_\curlyvee$ which was below a door ν of B_1 is transformed into a $\curlyvee$ link $\bar{e}_\curlyvee$ s.t. $\mathsf{pdoor}(\bar{e}_\curlyvee) = \mathsf{pdoor}(e_\curlyvee)$, and $\mathsf{adoors}(\bar{e}_\curlyvee) = (\mathsf{adoors}(e_\curlyvee) \setminus \mathsf{adoors}(B_1)) \cup \{\nu^i \mid \nu \in \mathsf{adoors}(e_\curlyvee) \cap \mathsf{adoors}(B_1)\}$.

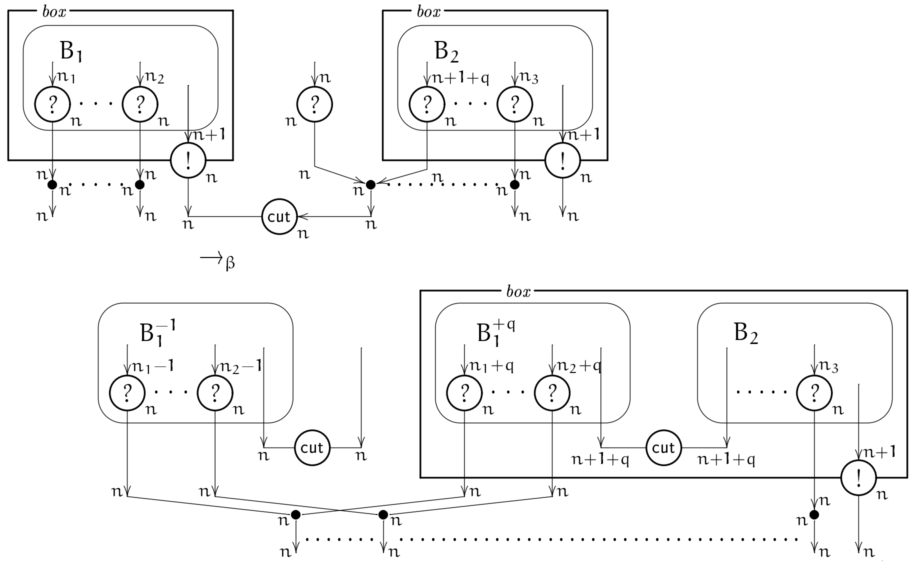

Fig. 22. Interacting boxes: global (β) rule.

Remark 56 The extent of the previous pattern is not completely defined. For instance, to faithfully reproduce the schema of Figure 22 in the λ-calculus structures, we should split the usual beta rule in two interactions (*i.e.*, we should switch from λ-terms to pure proof nets): a first one merging the top arrow of the redex to the body arrow of the λ link; a second one that, apart for the cut link, would be an instance of the rule of Figure 22. We could instead prefer to maintain the usual formulation of beta rule assuming that it is a valid instance of the pattern in Figure 22. In fact, all the theorems we are going to give in the next sections are still valid under this assumption. Besides, in order to not be worthlessly abstract, we will refer to the schema of Figure 22 when it will help in exposing the proofs of the forthcoming theorems.

Remark 57 In terms of levels, to push B_1^i inside the boxes of $e_?^i$ means to increase all the levels in B_1^i by the offset $q_i = \ell(\mathsf{idoor}(e_?^i)) - \ell(\mathsf{edoor}(e_?^i)) - 1$ (operation denoted by the superscript $+q_i$ in Figure 22). By the restriction on the levels of a structures, we see that this effectively means to enclose B_1^i inside $q_i+1 > 0$ new boxes when $e_?^i$ is auxiliary door of some boxes (see B_1^{+q} in Figure 22); otherwise, when $e_?^i$ did not close any box, it just means to remove the box around B_1^i (see B_1^{-1} in Figure 22).

10.2 Sharing β rule

Our aim is to show that, just assuming that the ℓ-nets of the calculus to be implemented have the box nesting property, the global version of the β rule in Figure 22 may be replaced by the sharing or β_s rule of Figure 23.

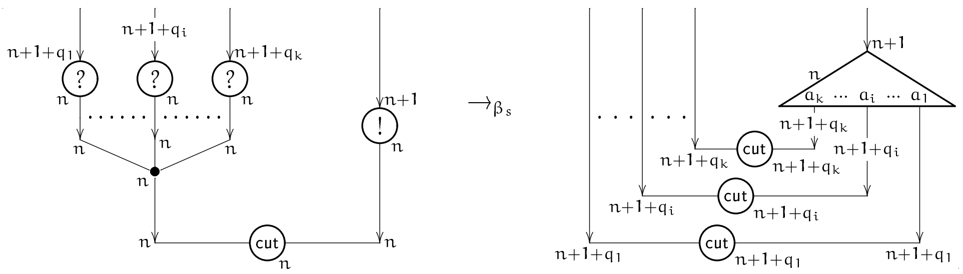

Fig. 23. Interacting boxes: shared (β_s) rule.

In more details, we want to prove that, applying with no restriction the $\beta_s \cup \pi$ rules to an ℓ-net N, we obtain a proper $s\ell$-structure G whose read-back $\mathscr{R}(G)$ is a β reduct of N.

Before to go on with this plan, let us note that the β rule is sound w.r.t. the box nesting property.

Fact 58 *The box nesting property is stable under* β *reduction.*

10.3 Unshared reductions

The last step towards the achievement of our goal is the proof of a simulation property for β_s similar to the one already proved for π (Lemma 44). Unfortunately, in the case of β_s the procedure is not so direct as in the case of the π rules, since the β_s-reduct of a $u\ell$-structure is a $u\ell$-structure only when the contracted redex is unary (*i.e.*, it involves only one ? link).

There is however a way to transform a k-ary β redex into a set of unary ones. In fact, let us split the reduction of a β redex $r = (e_!, \{e_?^1, \dots, e_?^k\})$ in two steps (assume to represent a β redex by the pair formed of the box door links involved in it): firstly, the box B_1 of the ! link $e_!$ is duplicated, creating an instance B_1^i for each $e_?^i$ (see Figure 24); secondly, each of the k redexes $(e_!, \{e_!^i\})$ is reduced applying the β rule. The first step of this decomposition defines a *box duplication* rule transforming a k-ary β redex into k unary redexes.

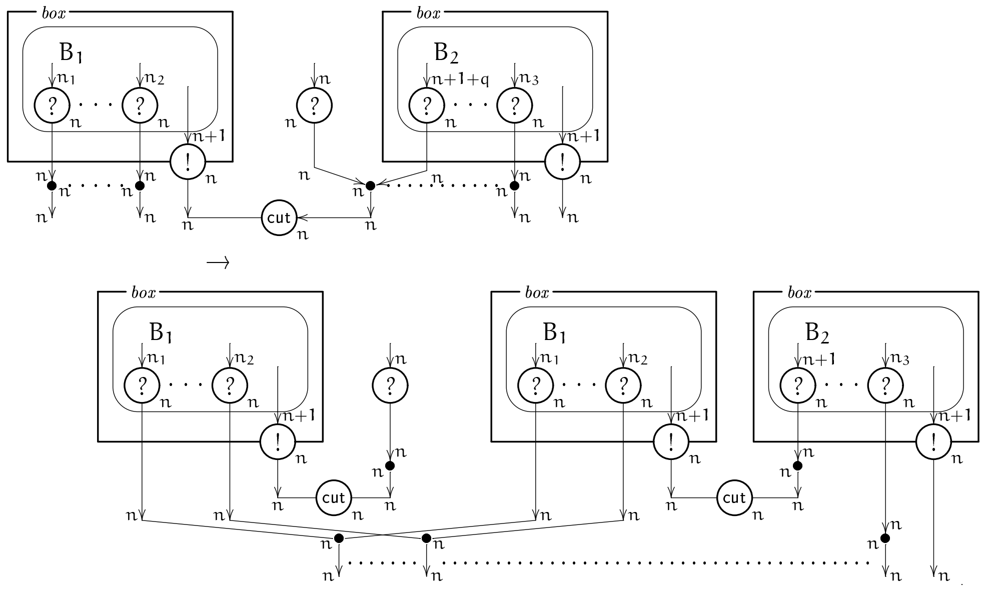

Fig. 24. Interacting boxes: box duplication.

The previous decomposition can be exploited to define the unshared version of the β rule we need, the β_u rule, just replacing β_s for β in the second step.

The β rule can be applied to ℓ-structures only. Exploiting the definition of boxes we gave for the proper $u\ell$-structures (see section 9), the β_u rule can be instead used to reduce proper $u\ell$-structures—this is not possible for β as we would be in trouble reindexing the thresholds of the lifts. This approach is sound. In fact, the β_u-reduct of a proper $u\ell$-structure U is a proper $u\ell$-structure and its read-back is the correct β-reduct of $\mathscr{R}(U)$.

Lemma 59 *Let* U *be a proper* $u\ell$*-structure. If* $U \to_{\beta_u} U'$*, then* U' *is proper and* $\mathscr{R}(U) \to_\beta \mathscr{R}(U)'$.

Proof. Let $U \to U''$ be a box duplication (see Figure 24). Each arrow/link of U'' is a residual of a unique arrow/link of U. More formally, the residual relation $\mapsto \subset \mathsf{vtx}(U) \times \mathsf{vtx}(U'')$ is defined by: *(i)* $v \mapsto v^i$, if $v \in \mathsf{vtx}(B_1)$ and v^i is the i-th instance of v; *(ii)* $\mathsf{pdoor}(e_Y) \mapsto \mathsf{pdoor}(e_Y)^i$, if e_Y is the contraction link of the redex; *(iii)* $v \mapsto v$, if v is not one of the previous cases. And analogously for $\mapsto \subset \mathsf{lnk}(U) \times \mathsf{lnk}(U'')$.

Let $r : U \to_{\beta_u} U'$, with $r = (e_!, \{e_?^1, \ldots, e_?^k\})$. For any internal state $\mathscr{S}$ of U'', let us take $\mathscr{S}^i(e) = \mathscr{S}(e^i)$, when $e \in B_1$ and $e \mapsto e^i$, and $\mathscr{S}^i(e) = \mathscr{S}(e)$ otherwise, for $i = 1, \ldots, k$. Let $\mathscr{A}^i$ be the $\mathscr{S}^i$-solution of U. By Lemma 55, all the $\mathscr{A}^i$ coincide outside B_1, and in particular, $\mathscr{A}^1(v) = \mathscr{A}^2(v) = \cdots \overset{def}{=} \mathscr{A}(v)$, when $v \mapsto v$. The previous map $\mathscr{A}(v)$ can then be extended to the whole $\mathsf{vtx}(U'')$ defining $\mathscr{A}(v) = \mathscr{A}^i(v^i)$, when $v \in B_1$ and $v \mapsto v^i$. The map $\mathscr{A}$ is the $\mathscr{S}$-solution of U''.

Let $q_i = \ell_U(\mathsf{idoor}(e^i_?)) - \ell_U(e_!) - 1$ and let a_i be the name chosen for the lift inserted reducing the i-th β redex of U''. For any internal state $\mathscr{S}'$ of U'', let us take the internal states $\mathscr{S}_0(e)$ and $\mathscr{S}(e)$ s.t.: (*i*) $\mathscr{S}_0(e) = \mathscr{S}(e) = \mathscr{S}'(e)$, when $e \neq e^i_!$; (*ii*) $\mathscr{S}_0(e^i_!) = 1$; (*iii*) $\mathscr{S}(e^i_!) = \mathcal{L}[\ell(e_!), q_i, a_i]\,\mathcal{H}^i$, being $\mathscr{A}_0$ the $\mathscr{S}_0$-solution of U (actually $\mathscr{S}_0$ is an internal state of U'', but we use the same name for the internal state of U obtained replacing $\mathscr{S}_0(e_!) = 1$ for $\mathscr{S}_0(e^i_!) = 1$) and $\mathcal{H}^i\,\mathscr{A}_0(\mathsf{edoor}(e^i_?)) = \mathscr{A}_0(\mathsf{idoor}(e^i_?))$. By Lemma 55 and by the result of the previous paragraph, the $\mathscr{S}$-solution of U'' is s.t. $\mathscr{A}(v) = \mathscr{A}_0(v)$ when $v \notin B_1$, and $\mathscr{A}(v^i) = \mathscr{A}_0(v)\,\mathscr{S}(e^i_!)^{\uparrow r}$ when $v \in B_1$ and $v \mapsto v^i$, where $r = \|\mathscr{A}_0(\mathsf{pdoor}(e_!))\|$.

The map $\mathscr{A}'$ obtained restricting $\mathscr{A}$ to $\mathsf{vtx}(U'') \cap \mathsf{vtx}(U')$ (since all the β reduced redexes are unary each arrow of U'' has at most one residual) is the $\mathscr{S}'$-solution of U'. Thus, U' is proper, for the previous procedure applies to any $\mathscr{S}'$.

The way in which $\mathscr{A}'$ is constructed shows that (take $\mathscr{S}'$ equal to the quiescence internal state) $\ell^{\mathscr{I}}_U(v) = \ell^{\mathscr{I}}_{U'}(v)$ when $v \notin B_1$, and $\ell^{\mathscr{I}}_U(v) + q_i = \ell^{\mathscr{I}}_{U'}(v^i)$ when $v \in B_1$ and $v \mapsto v^i$. That is, $\mathscr{R}(U) \to_\beta \mathscr{R}(U')$. $\square$

Proposition 60 *Let* G_0 *be a proper* $s\ell$*-structure and let* $U_0 \lll G_0$*. For any* $G_0 \to_{\beta_s} G_1$*, there exists* $U_0 \to^+_{\beta_u} U_1$ *s.t.* $U_1 \lll G_1$*.*

Proof. Let $r : G_0 \to_{\beta_s} G_1$. The corresponding reduction $\rho : U_0 \to^+_{\beta_u} U_1$ we are looking for is a development of the redexes $R = M_0^{-1}(r)$, with $M_0 : U_0 \lll G_0$. The complete unfolding $M_1 : U_1 \lll G_1$ is instead the natural one induced by the residual relation.

Before to give all the details of the proof let us note that the residual relation defined in the proof of Lemma 59 can be defined for any rule and extended to redexes in the natural way. We already remarked that $\mapsto$ is a partial one-to-many relation in the case of box duplication (*i.e.*, $y \mapsto x$ and $z \mapsto x$ only if $y = z$). In the case of β_s it is indeed an injective partial function.

Let $r : U_0 \to U'$ by a box duplication. For any $v' \in \mathsf{vtx}(U')$, let us define $M'(v') = M(v)$ if $v \mapsto v'$, and analogously for the links. Though $M' : U' \preccurlyeq G_0$, the proper $u\ell$-structure U' is not a least-shared-instance of G_0. In fact, for any internal state $\mathscr{S}$ s.t. $\mathscr{S}(e^1_!) = \mathscr{S}(e^2_!)$ with $e_1 \mapsto e^i_!$ and $e_1 \neq e_2$, if $v \mapsto v^i$ and $v^i \in \mathsf{box}^{\ell}_{U'}(e^i_!)$, then $M'(v^1) = M'(v^2)$ and $\mathscr{A}(v^1) = \mathscr{A}(v^2)$, where $\mathscr{A}$

is the $\mathscr{S}$-solution of U'. Nevertheless, we stress that such internal states are the only ones for which we can simultaneously have $M'(\nu^1) = M'(\nu^2)$ and $\mathscr{A}(\nu^1) = \mathscr{A}(\nu^2)$, for some $\nu^1 \neq \nu^2$ that are not contracting arrows of a $\curlyvee$ link.

A finite development of box duplications relative to the set R is a sequence of box duplication rules $\rho' : U_0 \to^* U'$ s.t. U' does not contain any residual of $r \in R$ and, for any $r' \in \rho'$, there is $r \in R$ s.t. $r \mapsto r'$. Finite developments of box duplications definitely exist (actually all the developments are finite); for instance, reduce the redexes of R using an innermost-outermost strategy. We have already seen that $M' : U' \preccurlyeq G_0$. Furthermore, any redex r' of U' s.t. $M'(r') = r$ is unary. Hence, let a_i be the name of the lift port to which $\mathsf{idoor}(e^i_?)$ is connected in G_1; and let $\rho'' : U' \to_{\beta_s} U_1$ be obtained reducing all the (unary) redexes $R' = \{(e'_!, \{e'_?\}) \mid M'(e'_!) = e_!, M'(e'_?) = e^i_?\}$, and inserting a lift with port name a_i for each redex $(e'_!, \{e'_?\})$ s.t. $M'(e'_?) = e^i_?$, for $i = 1, 2, \ldots$. By construction, there is $M_1 : U_1 \preccurlyeq G_1$ (the natural one induced by the restriction of M' to the arrows of U_1). Moreover, the way in which names are assigned to lifts, plus the concluding remark of the previous paragraph, allows to state that $M_1 : U_1 \prec\!\!\!\preccurlyeq G_1$.

To conclude the proof, just note that $U_0 \xrightarrow{\rho'} U' \xrightarrow{\rho''} U_1$ can be easily rearranged in a sequence of β_u reductions. $\square$

11 Sharing reductions of nets

We now have all the ingredients we need to state our main results:

(i) We know how to semantically and syntactically read-back an ℓ-structure from a proper $s\ell$-structure—remind that $G \to^*_\pi \mathscr{R}(G)$ (see Proposition 50).
(ii) We know that properness and box nesting property are tightly related (see section 9).
(iii) We know that β_s and π reductions of proper $s\ell$-structures can be simulated by corresponding reductions of proper $u\ell$-structures (see Lemma 44 and Proposition 60).
(iv) We know that β_s and π are sound w.r.t. read-back.

In other words, the rules β_s and π define an abstract computational system, say a *sharing graph machine*, in which the implementation of the global rule β is decomposed in a set of atomic steps.

In order to formalize the latter claim by a theorem, let us introduce some final notations.

11.1 Nets

Let N be an ℓ-structure over the signature $\Sigma \uplus \{!, ?, \curlyvee\}$. We say that N is *correct*, or that it is a *$\Sigma\ell$-net*, when it satisfies some given correctness criterion that subsumes the box nesting property.

A proper $s\ell$-structure over $\Sigma \uplus \{!, ?, \curlyvee\}$ is *correct*, say a *$\Sigma s\ell$-net*, when its read-back is a $\Sigma\ell$-net. Further, given a set Nets of correct ℓ-nets, we will say that a set SNets of proper $s\ell$-structures over Σ is the set of the correct $s\ell$-nets corresponding to Nets, when $\mathsf{G} \in \mathsf{SNets}$ iff $\mathscr{R}(\mathsf{G}) \in \mathsf{Nets}$.

11.2 Net rewriting systems

A *net rewriting system* over the signature Σ (let us implicitly assume that $\{!, ?, \curlyvee\} \subset \Sigma$) is a triple $\sigma = (\Sigma, \mathsf{Nets}(\sigma), \rightarrow_\sigma)$, where $\mathsf{Nets}(\sigma)$ is some set of $\Sigma\ell$-nets, and $\rightarrow_\sigma \subset \mathsf{Nets}(\sigma) \times \mathsf{Nets}(\sigma)$ is the congruence induced by some set of graph rewriting rules. Moreover, $\beta \in \sigma$ is the only rule involving box door links and contractions.

Definition 61 (sharing implementation) *The* sharing implementation *of a net rewriting system $\sigma = (\Sigma, \mathsf{Nets}(\sigma), \rightarrow_\beta \cup \rightarrow_\gamma)$ is the (sharing) net rewriting system $\sigma_s = (\Sigma \uplus \{\nabla\}, \mathsf{SNets}(\sigma), \rightarrow_{\beta_s} \cup \rightarrow_\pi \cup \rightarrow_\gamma)$, where $\mathsf{SNets}(\sigma)$ is the set of $\Sigma s\ell$-nets corresponding to $\mathsf{Nets}(\sigma)$.*

Theorem 62 *Let σ_s be the sharing implementation of the net rewriting system $\sigma = (\Sigma, \mathsf{Nets}(\sigma), \rightarrow_\beta \cup \rightarrow_\gamma)$. We have that:*

(i) for any σ_s reduction $\mathsf{G} \rightarrow^*_{\sigma_s} \mathsf{G}'$, there is a corresponding σ reduction $\mathscr{R}(\mathsf{G}) \rightarrow^*_\sigma \mathscr{R}(\mathsf{G}')$;
(ii) for any σ reduction $\mathsf{N} \rightarrow^*_\sigma \mathsf{N}'$, there is a corresponding σ_s reduction $\mathsf{N} \rightarrow^*_{\sigma_s} \mathsf{N}'$;
(iii) σ_s is strongly normalizing iff σ is;
(iv) σ_s is confluent iff σ is;
(v) any σ_s normal form of $\mathsf{G} \in \mathsf{SNets}(\sigma)$ is a σ normal form of $\mathscr{R}(\mathsf{G})$.

Proof.

(i) Since it is trivial to see that $\mathsf{G} \rightarrow_\gamma \mathsf{G}'$ implies $\mathscr{R}(\mathsf{G}) \rightarrow_\gamma \mathscr{R}(\mathsf{G}')$, it follows immediately from Lemma 44 and Proposition 60.
(ii) A *standard strategy* for σ_s is a reduction strategy s.t. each β_s is followed by a normalizing π reduction. By Proposition 50 and Proposition 60, we see that $r : \mathsf{N} \rightarrow_\beta \mathsf{N}''$ implies $\rho : \mathsf{N} \overset{r}{\rightarrow}_{\beta_s} \mathsf{G} \rightarrow^*_\pi \mathscr{R}(\mathsf{G}) = \mathsf{N}''$, where ρ is

standard. Thus, for any $N \rightarrow^*_{\sigma} N'$ there is a standard strategy reduction s.t. $N \rightarrow^*_{\sigma_s} N'$.

(iii) The π rules are strongly normalizing (Proposition 49). Hence, a σ_s reduction ρ_s contains an infinite number of them iff it contains an infinite number of rules β_s or γ. As a consequence, any infinite σ_s reduction ρ_s contains an infinite number of rules β_s or γ. But this is possible iff the σ reduction ρ corresponding to ρ_s is infinite.

(iv) Trivial.

(v) Remind that the π normal form of a proper $s\ell$-structure G is equal to $\mathscr{R}(G)$ (see Proposition 49). □

11.3 Optimality

Sharing implementations are tightly related to λ-calculus optimal reductions [Lév80] and to their generalization to Interaction Systems [AL94]. Anyhow, because of the generality of the rewriting system σ, such a correspondence is restricted to the implementation of the β rule only.

Let us assume the hypotheses and notations of Theorem 62. A σ_s reduction $\rho_s : G \rightarrow_{\sigma_s} G'$ is β-*optimal* when the number of β_s rules executed by ρ_s is lower or equal than the number of β_s rules applied by any other equivalent reduction $\rho'_s : G \rightarrow_{\sigma_s} G'$ (*i.e.*, ρ_s can be obtained from ρ'_s applying the rules of Lévy's permutation equivalence, see [Lév80]).

A β-*duplicating* rule is a π rule in which the redex (e_∇, e) is formed of a mux e_∇ whose principal door is the internal door of a box door link e. Let σ_o (π_o) be the set of all the σ_s (π) rules but the β-duplicating ones.

Theorem 63 *Any σ_o reduction is β-optimal.*

Proof. Let $\rho_o : G \rightarrow^*_{\sigma_o} G'$. By hypothesis, there are no erasing β redexes (in each β redex there is at least a ? link) and the γ rules do not erase β redexes. Hence, at least a residual of each $r \in \rho_o$ must be reduced in any reduction $\rho : G \rightarrow^*_{\sigma_s} G'$. Moreover, any redex r of G_1 s.t. $G \rightarrow^*_{\sigma_o} G_1$ has at most a residual in any G_2 s.t. $G_1 \rightarrow^*_{\sigma_o} G_2$. □

By the way, the β-optimal reductions correspond to the usual λ-calculus or linear logic optimal reductions ([Lam90,GAL92b,GAL92a,AG97]) when σ_s is the corresponding sharing implementation (*cf.* section 4.1). In such cases, to delay or to not execute a β-duplicating rule does not hide any β redex. In fact, for any β redex r of $\mathscr{R}(G)$, there is a reduction $G \rightarrow^*_{\pi_o} G'$ s.t. the image of r in G' is a β redex (note that $\mathscr{R}(G') = \mathscr{R}(G)$). As a consequence, it is not difficult to prove that for any λ or MELL $s\ell$-net G, if N is the σ normal form of $\mathscr{R}(\mathsf{G})$, then the σ_o normal form of G is an $s\ell$-net G' s.t. $\mathscr{R}(\mathsf{G}') = \mathsf{N}$.

Besides, an analog property does not old in the general case. In fact, let us assume that σ is not confluent. This means that γ is not confluent. Hence, let ρ_1 and ρ_2 be two γ reductions internal to a box B of an ℓ-net N. Let us assume that in N there are at least two instances of B, say B' and B''. A possible reduction of N is $\rho_i'\rho_j'' : \mathsf{N} \rightarrow_\gamma \mathsf{N}_{ij}$, in which ρ_i' is internal to B' and ρ_j'' is internal to B'', with $i, j \in \{1, 2\}$. For the sake of simplicity let us also assume that each N_{ij} is in normal form. If G is a proper $s\ell$-structure in which B' and B'' are shared, there is no β-optimal reduction $\mathsf{G} \rightarrow_{\sigma_o} \mathsf{G}_{12}$ s.t. $\mathscr{R}(\mathsf{G}_{12}) = \mathsf{N}_{12}$, and analogously for N_{21}.

The latter point shows that the π rules are not just a nice way to present Lévy's optimality. They give indeed a low-level implementation interesting per se, that applies even to systems for which Lévy's optimality cannot be defined.

11.4 Box erasing and garbage collection

As already pointed out in Remark 14, our definition of box forbids a direct implementation of MELL with weakening. It might seem that this also imply the impossibility to erase boxes. Luckily, the latter point is instead false. In fact, a box may be erased, provided that Σ contain an erasing link suitably shaped for this purpose, say an ϵ link. (In Definition 61 we asked that the only rewriting rules involving box door links are β and π. As we need that an ϵ link erase box door links too, the use of an erasing link would imply a slight reformulation of the definition of sharing implementation. Besides, this would not hurt Theorem 62, the only consequence would be a more involved statement.)

For instance, let us take the case that the ϵ link be a sort of mux with 0 auxiliary ports. According to this, an ϵ link at level m erases any link to which it is connected but box door links of level n, when $n \leq m$, and muxes. (Instantiating the α-propagation rule to the case of ϵ, we see indeed that such an ϵ-propagation rule erases the mux involved in it. So, the meaning of the previous restriction is that a mux $e_\triangledown$ is not erased by an ϵ link e_ϵ when $\mathsf{doors}(e_\epsilon) \in \mathsf{adoors}(e_\triangledown)$.) According to the restriction on levels, an ϵ link stops its erasing at the internal port of a box door link whose level is lower than its threshold. Then, let $r = (e_!, \{e_?\})$ be a β redex of an ℓ-structure G (for the sake of simplicity let us start with the case without muxes) s.t. above $e_?$ there is an ϵ link e_ϵ with $\ell_\mathsf{G}(e_\epsilon) = \ell_\mathsf{G}(e_?) + 1$. It is readily seen that the β_s reduction of r might be followed by an erasing reduction sequence ending with the complete erasing of $\mathsf{box}^{\ell}(e_!)$. Namely, $G \to^* G'$, for some G' in which in the place of $\mathsf{box}^{\ell}_{G}(e_!)$ there is a set of ? links—the auxiliary doors of $\mathsf{box}^{\ell}_{G}(e_!)$—each one below an instance of e_{ϵ}. Besides, in the general case, an ϵ link might stop at the auxiliary door of a mux, and consequently a subgraph that should be erased might rather remain alive. In order to erase such a garbage, we should lost most of the sharing contained in the net.

Independently from the formulation of the ϵ link, the previous example opens a problem of garbage collection completely independent from the one of soundness, which was indeed our main concern in this paper. Anyhow, because of the relevance of such a problem in any practical use of sharing graphs, a detailed study of garbage collection is one of our future goals.

12 Conclusions and further work

The box nesting property is the minimal (and natural) requirement under which the sharing reductions can be used to get a local and distributed implementation of β-like rules. It is our aim to study how the class of the calculi having the box nesting property relates with the classes already studied by the term graph rewriting community. As shown by the relevant examples of λ-calculus and MELL, a wide set of calculi fits in this class. Asperti and Laneve [AL94] proved that sharing graphs can be used for the so-called Interaction Systems, a subset of the Combinatory Reduction Systems for which there is a corresponding intuitionistic logic. Our requirement seems weaker and more general, so we expect to find a much wider class.

Lafont's Interaction Nets are the natural system to compare with sharing graphs [Laf90]. Sharing graphs are interaction nets whose interactions are no more restricted to the principal ports of the interacting elements and with the additional information of levels. We think that the relation between interaction nets and sharing graphs is similar to the relation between Combinatory Logic and λ-calculus. Both such systems are Turing complete and based on an applicative principle, but to simulate the reduction of a λ-term inside Combinatory Logic may greatly increase the length of the reduction. In spite of this, Lafont's interaction combinators [Laf95] and the work of Fernández and Mackie [FM96b,FM96a] on how to encode term rewriting systems into interaction nets might give useful insights for the determination of the class of systems implementable by sharing graphs.

One of the restriction we imposed to β rule is that in a β redex $(e_!, \{e^1_?, \ldots, e^k_?\})$ no $e^i_?$ can be an auxiliary door link of $\mathsf{box}^{\ell}(e_!)$, that is, we forbade a box to interact with itself. All the results of the paper should hold even dropping such a constraint. Nevertheless, we chose to eliminate such cases since their presence would have involved the description of β. Furthermore, the analog of a box interacting with itself is the μ rule $\mu x.T \rightarrow_{\mu} T[\mu x.T/x]$, that is, a calculus with an explicit recursion operator. Hence, we prefered to maintain the distinction between β and μ also in sharing graphs and to reserve a separate study for the μ-like rules in which an object interacts with itself.

We already stated that β_s plus π give an abstract machine by which to get sharing implementations. Further, such a sharing graph machine seems the natural low-level model of λ-calculus. We hope that this would finally lead to a satisfactory measure for the cost of λ-calculus reductions, and that this would finally lead to a way to compare λ-calculus with the computational models used in complexity theory. We think that the interest in the definition of a suitable notion of cost and of complexity classes for sharing computations is even more appealing after the results of Asperti [Asp96] and Lawall and Mairson [LM96] showing that Lévy families are unlike to be the cost model of λ-calculus reductions.

Acknowledgments

I thank Simone Martini who supervised my thesis and always encouraged me persisting, even when things seemed to get everyday more involved; Andrea Asperti who introduced me to the intriguing world of optimal reductions and sharing graphs; Andrea Masini for fruitful discussions; and Corrado Böhm, if I had not met him during my undergraduate studies in electronic engineering, maybe I would have never worked on these topics.

References

[AG97] Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages.* Cambridge University Press, 1997. To appear.

[AGN95] A. Asperti, C. Giovannetti, and A. Naletto. The Bologna Optimal Higher-Order Machine. Technical Report UBLCS-95-9, University of Bologna, Department of Computer Science, March 1995.

[AL94] Andrea Asperti and Cosimo Laneve. Interaction systems I: The theory of optimal reductions. *Mathematical Structures in Computer Science*, 4(4):457–504, December 1994.

[Asp95] Andrea Asperti. Linear logic, comonads and optimal reductions. *Fundamentae Informaticae*, 22:3–22, 1995.

[Asp96] Andrea Asperti. On the complexity of beta-reduction. In *Proceedings of Twentythird Annual ACM Symposyum on Principles of Programming Languages*, St. Petersburg Beach, Florida, January 1996. ACM.

[DR93] Vincent Danos and Laurent Regnier. Local and asyncrhonous beta-reduction. In *Proceedings of 8th Annual Symposium on Logic in Computer Science*, pages 296–306, Montreal, Canada, June 1993. IEEE.

[FM96a] Maribel Fernández and Ian Mackie. From term rewriting to generalised interaction nets. In Herbert Kuchen and S. Doaitse Swierstra, editors, *Programming Languages: Implementations, Logics, and Programs. 8th International Symposium, PLILP'96, Aachen, Germany*, number 1140 in Lecture Notes in Computer Science, pages 319–333. Springer-Verlag, September 1996.

[FM96b] Maribel Fernández and Ian Mackie. Interaction nets and term rewriting systems (extended abstract). In H. Kirchner, editor, *Trees in Algebra and Programming – CAAP'96*, number 1059 in LNCS, pages 149–164. Springer-Verlag, 1996.

[GAL92a] G. Gonthier, M. Abadi, and J.-J. Levy. The geometry of optimal lambda reduction. In *Conference record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages: papers presented at the symposium, Albuquerque, New Mexico, January 19–22, 1992*, pages 15–26, New York, NY, USA, 1992. ACM Press.

[GAL92b] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 223–234, Santa Cruz, California, 22–25 June 1992. IEEE Computer Society Press.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[Gir89] Jean-Yves Girard. Geometry of interaction 1: Interpretation of system F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Logic Colloqium '88*, pages 221–260, Amsterdam, The Netherlands, 1989. Elsevier (North-Holland).

[GMM97a] Stefano Guerrini, Simone Martini, and Andrea Masini. Coherence for sharing proof nets, 1997. Submitted. Extended abstract in Harald Ganzinger, editor, *Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA-96)*, volume 1103 of *LNCS*, pages 215–229, New Brunswick, NJ, USA, July 27–30 1996. Springer-Verlag.

[GMM97b] Stefano Guerrini, Simone Martini, and Andrea Masini. Proof nets, garbage, and computations. In *TLCA 97*, Lecture Notes in Computer Science, Berlin, 1997. Springer-Verlag.

[Gue97] Stefano Guerrini. Sharing-graphs, sharing-morphisms and (optimal) λ-graph reductions. In J. Ginzburg, Z. Khasidashvili, J.-J. Lévy, and E. Vallduví, editors, *The Tbilisi Symposyum on Logic, Language and Computation, Tbilisi, Georgia, October '95.* CSLI Publications, 1997.

[Laf90] Yves G. A. Lafont. Interaction nets. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 95–108. Association for Computing Machinery, 1990. Papers presented at the Symposium, San Francisco, California, January 17-19, 1990.

[Laf95] Yves Lafont. Interaction combinators. FTP, July 1995.

[Lam90] John Lamping. An algorithm for optimal lambda calculus reduction. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 16–30, San Francisco, California, January 1990.

[Lév80] Jean-Jacques Lévy. Optimal reductions in the lambda-calculus. In Jonathan P. Seldin and J. Roger Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 159–191. Academic Press, 1980.

[LM96] Julia L. Lawall and Harry G. Mairson. Optimality and inefficiency: What isn't a cost model of the lambda calculus? In *Proceedings of the 1996 ACM SIGPLAN International Conference on Functional Programming*, pages 92–101, Philadelphia, Pennsylvania, 24–26 May 1996.

[MM95] S. Martini and A. Masini. On the fine structure of the exponential rule. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 197–210. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

[PJ86] Simon L. Peyton Jones. *The Implementation of Functional Programming Languages.* Series in Computer Science. Prentice-Hall International, Englewood Cliffs, NJ, 1986.

[Reg92] Laurent Regnier. *Lambda-Calcul et Reseaux.* Phd Thesis, Université Paris 7, Paris, January 1992.

[Wad71] C. P. Wadsworth. *Semantics and pragmatics of the lambda-calculus.* Phd Thesis, Oxford, England, 1971. Chapter 4.