A general theory of sharing graphs

Stefano Guerrini

Institut de Mathématiques de Luminy - 163, Avenue de Luminy - Case 907 - 13288 Marseille Cedex 9, France - email: `stefanog@iml.univ-mrs.fr`

Sharing graphs are the structures introduced by Lamping for the implementation of 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 implement a wide class of calculi.

We give a semantical characterization of sharing graphs independent from the calculus to be implemented. By means of an algebraic interpretation of sharing graphs, we define a subclass of them, the so-called *proper sharing graphs*, on which the usual notion of graph unfolding gives a lower semi-lattice. The *least-shared-instance* of a proper sharing graph is its maximal proper unfolding, that is, the unique proper *unshared graph* that the unfolding partial order associates to it. Exploiting a simulation property between the reductions of a proper sharing graph and the reductions of its least-shared-instance, we prove that the *read-back* of a proper sharing graph can be computed via an unfolding or *read-back reduction*.

Proper sharing graphs implement in a distributed and local way any graph calculus with a global reduction in the style of the beta-rule of lambda-calculus. In fact, correctness of the sharing implementation requires the so-called *box nesting property* only, or equivalently, it is proved under the only assumption that two redexes never partially overlap. Thus, sharing graphs constitute an abstract machine, say the *sharing graph machine*, that seems to be the most natural low-level computational model for functional languages. Moreover, Lévy's optimal reductions correspond to lazy reductions of that sharing machine.

We stress the proof strategy followed in the paper: it rests 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.

Key words: Linear logic; lambda calculus; proof nets; interaction nets; graph reduction; sharing and optimal reduction.

31 August 1998

1 Introduction

Techniques for the implementation of functional calculi based on the use of pointers (*i.e.*, on graphs) have been known since the beginning of seventies (see [Wad71,PJ86]). In that solutions, sharing is implemented by means of pointers from the nodes accessing a shared subterm to the root of the subterm. That techniques have a main drawback: in order to avoid unwanted side-effects, variable substitution requires a careful implementation (see [Wad71]). In fact, replacing a term T_s for the occurrences of a variable x in a term T_t by substituting a pointer to T_s for any pointer to x would apply the variable substitution in every term T_u that shares the representation of the variable x with T_t. To fix up that problem, the usual solution is to create a new instance of T_t in which to safely replace T_s for x. Soundness is recovered, but at the cost of the duplication of the redexes in T_t.

1.1 A fine decomposition of λ-calculus β-rule

Sharing graphs implement a sharing reduction much finer than the one obtainable keeping multiple pointers to the same subterm. In order to present that sharing reduction, let us try to implement the β-rule of λ-calculus by means of a step-by-step graph reduction system in which the objects rewritten are nodes rather than terms.

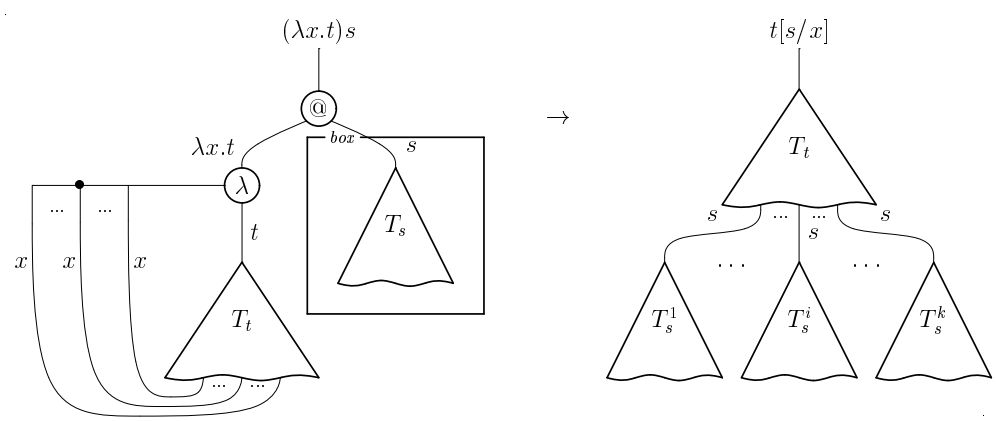

Fig. 1. λ-calculus β-rule.

Let us represent a λ-term t by a graph T derived from its abstract syntax tree: for each λ-abstraction $\lambda x.s$ in t, we merge the occurrences of the variable x in a unique node (the bullet in Figure 1) and, by means of a binding edge, we back-connect the node of x to the node of its binder $\lambda x.s$.

The first step in the decomposition of β-reduction is replacing the β-rule in Figure 1 with the sharing β-rule in Figure 2. The sharing β-rule does not

duplicate the graph of the argument; it rather connects the occurrences of x to the root of T_s by means of a new node, say a multiplexer or *mux* for short.

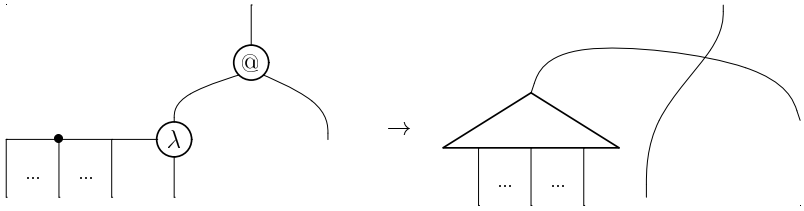

Fig. 2. λ-calculus sharing β rule.

The second step in the decomposition of β-reduction is the definition of a set of rules performing a step-by-step duplication of the redex argument, say a local and distributed duplication of T_s; see Figure 3 (for the sake of a clear design, the muxes in Figure 3 are binary; the generalization to the $k$ary case is straightforward). The meaning of most of the rules in Figure 3 is self-evident, the only ones that deserve some thoughts are λ-up and absorption. In each duplication rule, the mux operates on the node that it points through its *principal port* (the vertex of the triangle), creating a new copy of the duplicating node for each of its *auxiliary ports* (the ports at the base of the triangle).

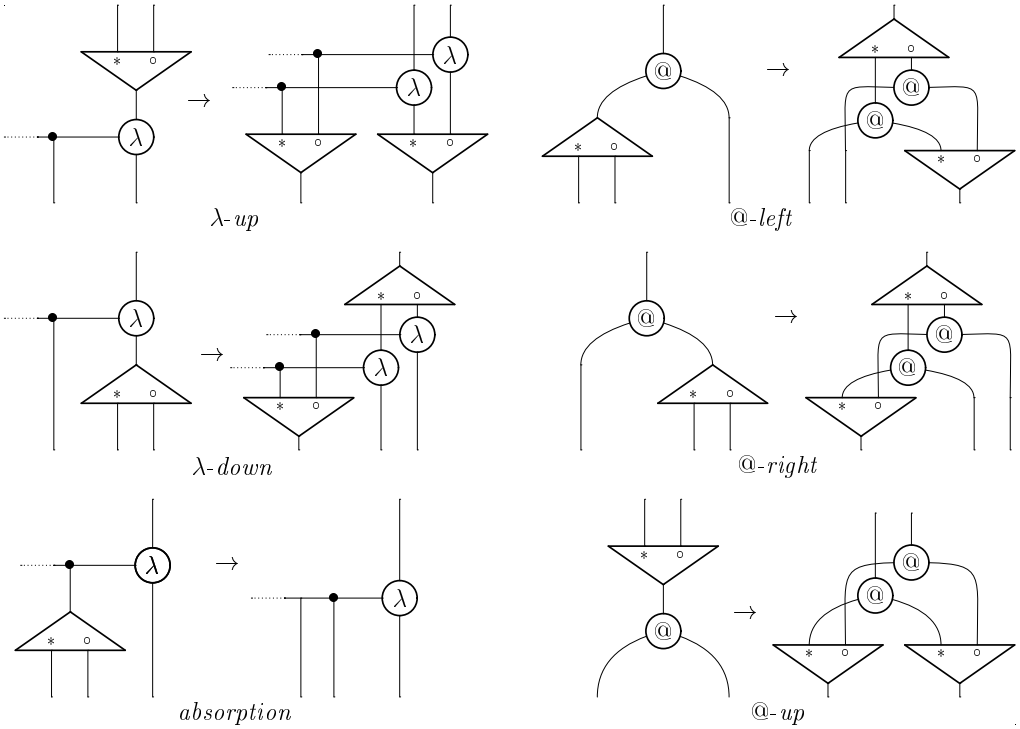

Fig. 3. Duplication rules.

In λ-up, the mux in the left-hand side not only duplicates the λ-node at its principal port, it splits the occurrences of the variable bound by the λ-node also (in Figure 3 we have drawn just one of that occurrences; analog duplications apply simultaneously to the other occurrences). In practice, we

have to create a new instance of the bound variable for each new instance of the λ-node.

The λ-up duplication rule is the first divergence from standard term graph reduction. After the execution of a λ-up, the subgraph pointed by a mux is no longer a shared subterm; in fact, the node inserted to split the bound variable is not a sharing or multiplexing operator, it is rather a sort of *demultiplexing* node. (Since in most of the applications that we shall consider the orientation of graphs is not so clear as in the case of λ-calculus, in the body of the paper, we shall 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 that can be filled in different ways, according to the instance of the shared subterm that we are considering. For instance, in the binary case, if we access a shared subgraph by the left (right) port of a mux, then we must exit the shared subgraph by the left (right) port of a corresponding demux; therefore, in the λ-up rule, the occurrences of the variable connected to the left (right) port of the demux are bound by the λ-node connected to the left (right) port of the mux above the shared subgraph. In the general case, this implies that we distinguish the ports of a mux, *e.g.*, numbering them from left to right (however, in the binary case, we shall continue using the names $*$ and $\circ$).

The second difference w.r.t. term graph rewriting is that demuxes can duplicate nodes; for instance, by an @-left rule. Therefore, since demuxes move bottom-up, the duplication of T_s is not strictly top-down.

The absorption rule applies when a mux reaches a free variable of T_s (w.l.o.g., let us restrict to the case of closed λ-terms); in that case, the mux has terminated its task and can be erased. However, there are cases in which the absorption rule is not correct. Namely, when the λ-node is internal to T_s; in that case, the mux is not on the border of T_s and its erasing is unsound; some thoughts allow us to realize that it is sound to let the mux duplicate the λ-node instead (to simplify the presentation, in Figure 3 we have dropped this kind of λ-rule).

The previous considerations on absorption are a first clue that we must keep some information on the global position of nodes. In the simplified case that we are analyzing (the local implementation of a single β-rule), that information will allow us avoiding unsound applications of absorption.

Reaching a free variable of T_s is not the only way in which a mux terminates its duplication task. For the sake of simplicity, let us assume that T_s does not contain free variables and let us apply rules of type up only. Each mux introduced along the reduction will eventually reach a corresponding demux at the bottom of T_s (*e.g.*, try reducing $(\lambda x.xx)(\lambda y.y)$). This situation corresponds

to the sharing of an empty subgraph; in fact, a mux immediately followed by a dual demux is equivalent to a set of direct connections between each pair of mux/demux ports with the same number (or name). In terms of graph rewriting rules, the previous equivalence gives a mux annihilation rule that replaces a matching mux/demux pair with a set of wires between the matching ports of the pair.

The implementation of our simplified problem is complete. The duplication rules plus the annihilation of muxes give an algorithm for the duplication of T_s. Therefore, the β-rule in Figure 1 can be replaced by a sharing β-rule followed by a sequence of duplication and annihilation rules.

The previous result might not seem particularly surprising. So far, we have got a detailed implementation of β-rule that mimics what we might have got by a direct coding of the duplication of T_s in some programming language external to the graph rewriting system. The important issue of the solution developed so far is that it opens the door to the following question: what does it happen if we freely execute the local rewriting system? 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 completed their task on T_s?

The first consequence of a free interleaving of sharing β-rules and duplications is that the graph may contain muxes belonging to distinct duplication processes. The situation is similar to that of absorption: when a mux/demux pair faces, we are no longer sure that they are matching (*i.e.*, that they delimit the same shared subgraph); therefore, there are cases in which the application of the annihilation rule is incorrect. The correct rewriting rule for the case of a pair of non-matching muxes is the swap rule in Figure 4.

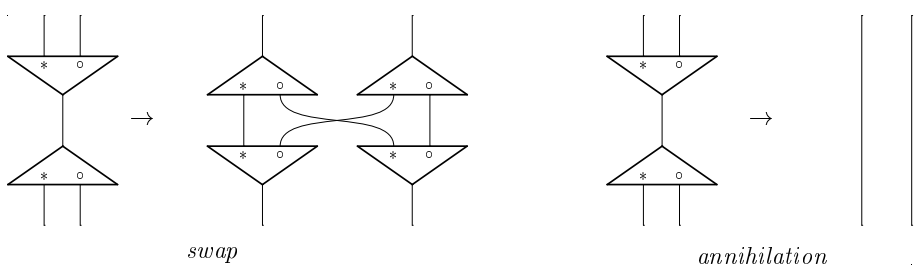

Fig. 4. Mux interactions.

The problem in the choice of the mux interaction rule is the second important clue suggesting the introduction of some additional information into the nodes of the sharing graphs.

A first attempt for matching muxes might be marking each mux introduced by a β-rule by a new label; assuming that the propagation rules preserve mux labeling, we might argue that two muxes should be matching when they face with the same label and non-matching otherwise. Unfortunately, Lamp-

ing (who firstly introduced sharing graphs, see [Lam90]) has shown that this solution does not work: there are cases in which two muxes that have been inserted by the same redex are non-matching. The right solution requires a dynamical labeling of muxes, say an index, that changes along the reduction. This leads us to the next ingredient for the understanding of sharing graphs, the so-called boxes.

1.2 Boxes

The λ-calculus β-rule takes a subgraph as argument and, after its duplication, connects each new instance of the argument to an edge corresponding to an occurrence of the variable in the redex. The subgraphs that are, or can become, the argument of a β-rule are clearly recognizable: they are the subtrees rooted at the right edge of an @-node. In the general case, when the graphs of the calculus are not trees, the argument of a β-like rule is not determined by the topology of the graph only, its subgraph must be explicitly denoted instead; for instance, by means of a frame, say a *box*, surrounding it. Therefore, the syntax of the calculus must include the rules for boxing β-arguments. For instance, in the case of the λ-calculus, there must be a box for each right subtree of an @-node. (As already remarked, the boxing of a λ-terms is implicit and useless if we proceed to β-reduction in a global way; in the sharing implementation of λ-calculus, the boxing of λ-terms turns to be necessary, although the one presented here is not the only solution; see [AL95].)

Boxes and rewriting rules of the calculus must be compatible, *i.e.*, the correctness of the boxing must be preserved by the β-rule. Therefore, the boxing rules for the λ-calculus cannot be arbitrary. In particular, they must verify the following basic proviso: two boxes cannot overlap in a non-trivial way, that is, two boxes are either disjoint or enclosed one into the other. In the λ-calculus, that proviso is definitely verified; moreover, replacing a term for a variable occurrence, the box of the replacing term is automatically inserted into the boxes surrounding the replaced variable occurrence. In the general case, the β-like rule must specify the rules for properly rearranging boxes after copying and displacing the argument of the redex.

Summarizing, the boxes of the graphs of the calculus must properly nest, say that the calculus must verify the *box nesting property*, and the β-rule must preserve the box nesting property.

The box nesting property gives us the tool for avoiding the introduction of a global constructor for the representation of box frames. Assuming that each node/edge is labeled by the number of boxes surrounding it, say the *box nesting level* of the node/edge, each node with two edges whose levels are respectively

n and $n+k$ (with $k \geq 0$) is the border of k boxes; moreover, the box at level $n+i$, with $0 < i \leq k$, is the maximal connected subgraph containing the node under analysis and s.t. its nodes are at a level greater or equal than $n+i$.

The introduction of the nesting levels solves the problems with the absorption rule and the mux interaction rules. A λ-node external to T_s bounds 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, in the sharing β-rule, let us assume to label the mux inserted by an index equal to the level n of the λ/@ pair in the redex. A box frames T_s (see Figure 1); therefore, its root is at level $n+1$. The index of the mux is thus a sort of *threshold* recognizing when to stop the duplication. Namely, we have the following cases: a mux duplicates the nodes that it meets at a level greater than its threshold; if a mux faces another mux with the same threshold, then the two muxes annihilate; finally, if a mux with threshold m reaches a λ-node whose level is $n \leq m$, then the absorption rule applies.

These are the guidelines for the mux rules in the presence of levels. However, we left to solve an important problem: after the execution of a β-rule, the box nesting of a new instance of the argument T_s depends on the position of the occurrence of the variable that it replaces; thus, duplicating T_s, we must correctly reassign a level to the nodes/edges of every new instance of T_s. In our example from λ-calculus, let n be the level of the β-redex; each instance T_s' of T_s is inserted into k new boxes, where $n+k$ is the level of the occurrence of the variable replaced with T_s'; therefore, since firing the β-rule we have opened the box surrounding T_s (the application in the β-redex disappears), each node/edge of T_s' increases its level by $k-1$. We stress the particular case of $k=0$, it corresponds to a variable occurrence that does not close any box; in that case, the β-rule decreases by 1 all the levels in T_s.

In order to perform the previous reindexing of the β-rule argument, each auxiliary port of a mux brings an offset by which to increase the level of the new instances of the nodes created by a mux propagation rule. The offsets of the ports are determined at mux creation (*i.e.*, by the β-rule): each offset is the difference between the level of the occurrence of the variable corresponding to the port (say $n+k$, if n is the level of the @/λ pair in the redex) and the level of the root of the argument (that is, since the root of T_s has level $n+1$, the offset is $k-1$).

The sharing rewriting system using node/edge levels (see Figure 20) will be detailedly discussed in section 8. For the moment, we only remark that the idea that a mux with threshold m_1 lifts the levels above m_1 applies in the swap rule also. In fact, let $m_1 < m_2$, where m_2 is the threshold of the second mux in the redex; the threshold of each new instance of the mux with threshold m_2 is lifted by the offset of the corresponding port of the mux with threshold m_1

(see Figure 20).

1.3 Soundness

The approach based on levels is clearly sound when each sharing β-rule is followed by a whole duplication of the argument of T_s. In that case, muxes never change their thresholds and the only case of mux interaction is the annihilation rule. When sharing β-rules and duplications are interleaved (hereafter, and when not otherwise specified, we shall include mux interactions into duplications), the analysis becomes greatly involved; muxes thresholds can arbitrarily change and there are cases in which two muxes with the same origin 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 shall prove that, for each *sharing graph*—intuitively, a graph obtained by means of a sequence of sharing β and duplication rules—there is a corresponding graph without muxes, write $G \mapsto \mathscr{R}(G)$ and say $\mathscr{R}(G)$ is the *read-back* of G, s.t. sharing β-rule and duplication rules are sound w.r.t. read-back. Namely, we shall prove that the following diagrams commute:

$$\begin{array}{ccc} G & \xrightarrow[\beta_s]{} & G' \\ \downarrow & & \downarrow \\ \mathscr{R}(G) & \xrightarrow[\beta]{+} & \mathscr{R}(G') \end{array} \qquad\qquad \begin{array}{ccc} G & \xrightarrow[\pi]{} & G' \\ \downarrow & & \downarrow \\ \mathscr{R}(G) & = & \mathscr{R}(G') \end{array} \tag{1}$$

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

According to the diagram on the left, we see that each β_s-rule corresponds to a (finite) non empty sequence of β-reductions: the execution of a β-rule 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-reduction. Furthermore, strong normalization and confluence, that are easily provable in the case in which each β_s-rule is followed by a maximal sequence of π-rules, hold in the general case too. Hence, the syntactical read-back definable via π-reduction, *i.e.*, interpreting a sharing graph G as a shared representation of its π-normal form $\mathsf{NF}_\pi(G)$,

coincides with the semantical one:

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

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 $U \prec\!\!\!\!\prec G$, in which all the muxes are unary (a unary mux will be named *lift*); then, the definition of $\mathscr{R}$ for such a graph U. Assuming the uniqueness of the unfolding U, let us call it the *least-shared-instance* of G, the read-back of G is defined by $\mathscr{R}(G) = \mathscr{R}(U)$.

Intuitively, the least-shared-instance U of G is a representation of $\mathscr{R}(G)$ in which the reindexing of the nodes has not been accomplished yet. At the same time, the unary muxes of U mark the border of a sharable subgraph—let us note that the image in G of a lift is a mux. Nevertheless, associating the correct graph $\mathscr{R}(U)$ to the unshared graph U (a graph is *unshared* when all its muxes are unary) is not an easy task: after replacing the unary muxes of U by direct connections, we must properly reindex the remaining nodes of the graph. In λ-calculus graphs, the tree shape of λ-terms forces the existence of a unique level assignment for each correct unshared graph without muxes; in the general case, such a uniqueness is not guaranteed and determining the right level of each node requires the knowledge of all the unary muxes that would have increased its level.

In the general case, even the definition of least-shared-instance is not easy, for we cannot take any unshared graph for which there is a graph morphism $M : U \to G$. In fact, because of the matching of muxes, not every path of G can appear in an admissible unfolding of G; thus, not every unfolding U of G definable via a graph morphism is correct.

Most of the paper is devoted to defining the correct unfolding partial order $\prec\!\!\!\!\prec$ by which to obtain the least-shared-instance U of a sharing graph G, *i.e.*, s.t. U is the minimal element for which $U \prec\!\!\!\!\prec G$. For that purpose, we have to develop an algebraic semantics of sharing graphs by which to characterize the shape of a proper unfolding morphism $M : U \prec\!\!\!\!\prec G$. In this way, we shall get an algebraic characterization of the proper unshared graphs for which there is a shared representation; at the same time, this will give us a characterization of the (proper) sharing graphs as the ones for which $U \prec\!\!\!\!\prec G$ 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 the β_s-rule is replaced by an unshared version of the β-rule, named 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 the β-rule reflects in a decomposition of the proof of soundness. In fact, given a sharing graph G, we shall prove that any β_s or π-reduction $G \to G'$ can be simulated on its least-shared-instance U. Therefore, diagram (1) is just diagram chasing:

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

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 shall see that the algebraic semantics is indeed an abstract characterization of that property, since we shall see that the proper sharing graphs defined by the algebraic semantics are the ones that π-normalize to a graph without muxes for which the box nesting property holds.

1.4 Optimality and other related works

Sharing graphs have been introduced by Lamping [Lam90] for the implementation of Lévy's optimal reductions of λ-terms [Lév80]. Several refinements of sharing graphs have been 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 can be interpreted inside the so-called Geometry of Interaction (GOI) of Girard [Gir89]; Asperti presented a more categorical justification of Gonthier's technique; Asperti and Laneve gave a generalization of the methodology to the so-called Interaction Systems, the subclass of the Combinatory Reduction Systems [Klo80] for which it is possible to find a Curry-Howard analogy with a suitable intuition-

istic logic. Furthermore, Asperti used sharing graphs for the implementation of an optimal version of an ML-like functional language [AGN96].

The main concern of all these studies have been the implementation of optimal reductions. Hence, the set of rules that they proposed was the maximal one preserving optimality. Here, we revert the point of view: the main concern is getting a sound 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 assuming that the duplication rules are applied following a lazy reduction strategy.

Let us see how this applies in the example developed so far. Executing the rules in 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-node. According to this, the only duplication rules unavoidable are @-left and λ-up. Dropping the other rules, not only has no impact on the computational power, it improves efficiency also, 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.

One of the main consequences of the previously described change of perspective is that the proof techniques differ 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 the optimal rules are sound w.r.t. the paths definable in 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 that approach makes use of the weakest conditions necessary for a proof of soundness; therefore, we might expect that it should 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, neither our algebraic semantics characterizes the proper sharing graphs as the result of a sharing reduction. Indeed, 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, but it is implicit in the definition of least-shared-instance that we shall 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 proved that the two classes coincide. The result of this approach is an 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 that we propose is that it represents a sort of abstract *sharing graph machine*. Differing from optimal implementations, the sharing graph machine does not need any external machinery for reading-back the result of a computation. The duplication rules internalize the read-back into the system; at the same time, a lazy application of them allows to compute via optimal reductions.

Finally, the point of view of the paper has a good proof theoretic motivation also: 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 that we shall study (section 2). The main difference w.r.t. what we have done so far is that we shall present sharing graphs as hypergraphs. Therefore, what in the introduction was a node will become a hyperedge, say a link, and what was an edge will become a vertex. Furthermore, we shall introduce two special kinds of link: box door links, to delimit the border of boxes; contraction links, to merge vertices. We shall define the level of links and vertices and we shall define boxes in terms of levels.

In section 3, we shall 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. We shall give the equivalence between muxes and Gonthier's fan and brackets also.

In section 5, we shall define the sharing morphisms; which are the basic tool for unfolding a sharing graph. In addition, we shall address why the unfolding problem is so difficult and requires the introduction of the algebraic semantics that we shall 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 shall name *arrows*. Links and arrows have a *level*. 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 of the pair (e_t) is the *tail* of e, the second one (e_h) is the *head* of e. Since we shall usually draw hyperarcs according to the top-down orientation—the tail of a hyperarc above its head—we shall 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 said 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, drawing G, a vertex will be represented by an 'arrow' from the link above it 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 shall say that G is connected and acyclic when G^* is connected and acyclic.

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.*

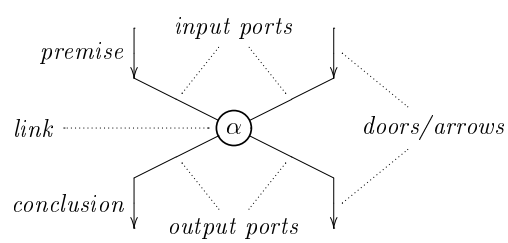

Fig. 5. A link of type α with two input ports and two output ports.

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 without any conclusion is a *target link*. Isolated links without any door 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 arrow can simultaneously be a premise and a conclusion of G. Premises and conclusions are the *doors* of G (and $\mathsf{doors}(G)$ denotes their set). The links above a conclusion or below a premise of G 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 $\mathsf{lnk}(G)$ the set of its links and by $\mathsf{vtx}(G)$ the set of its arrows. Furthermore, if $\alpha \in \Sigma$, we denote by $\mathsf{lnk}_\alpha(G)$ 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. $\mathsf{vtx}(R) \subseteq \mathsf{vtx}(G)$ and $\mathsf{lnk}(R) \subseteq \mathsf{lnk}(G)$. Since a structure does not contain isolated vertices, it is readily seen that the G-substructures coincide with the parts of $\mathsf{lnk}(G)$. In fact, not only any G-substructure R is uniquely determined by the set $\mathsf{lnk}(R) \subseteq \mathsf{lnk}(G)$ but, given a set of links $E \subseteq \mathsf{lnk}(G)$, there exists a (unique) G-substructure s.t. $\mathsf{lnk}(R) = E$. The standard inclusion relation and set operations apply to G-substructures according to their interpretation as set of links. Further, $\mathscr{P}(G)$ will denote the set of the G-substructures.

2.3 Boxes

For the implementation of boxes, we use two reserved link types named *box door links*: the !-link (of-course), or *principal door link*; the ?-link (why-not), or *auxiliary door link*. Both of 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 might have chosen any other pair of names.

Remark 4 According to Figure 6, we shall 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. However, that orientation is not required by the definition of door link; for instance, in the case of the λ-calculus, we shall 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 $\mathsf{pdoor}(B) \in \mathsf{doors}(B)$*—the* principal door *of B—which is external door of an !-link* $\mathsf{pdlink}(B)$*;*
(ii) all the arrows $\mathsf{adoors}(B) = \mathsf{doors}(B) \setminus \{\mathsf{pdoor}(B)\}$*—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 preferred 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 shall 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. Furthermore, in the second case, they cannot have the same principal door link.

Definition 7 (box nesting property) *Let* $\mathsf{box}_G : \mathsf{lnk}_!(G) \to \mathscr{P}(G)$ *be an assignment of boxes to the* !*-links of the structure* G*. We say that* G *with the boxes* $\mathsf{BX}_G = \{\mathsf{box}_G(e) \mid e \in \mathsf{lnk}_!(G)\}$ *has the* box nesting property*, when* $B_1 \cap B_2 \neq \varnothing$ *implies either* $B_1 \subset B_2$ *or* $B_2 \subset B_1$*, for any* $B_1, B_2 \in \mathsf{BX}_G$*.*

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

(i) If $\mathsf{pdlink}(B_1) = \mathsf{pdlink}(B_2)$, then $B_1 = B_2$, for any $B_1, B_2 \in \mathsf{BX}_G$.
(ii) The box nesting property does not forbid $\mathsf{doors}(B_1) \cap \mathsf{doors}(B_2) \neq \varnothing$. 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.

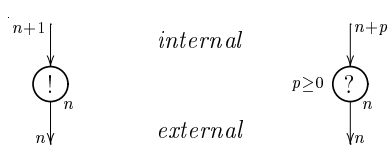

Fig. 6. Exponential links.

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 one box only. The previous difference is equal to $p \geq 0$ in a ?-link—for a ?-link may be auxiliary door of several (even 0) boxes. Note that the levels of the box door 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 !-link or of 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).

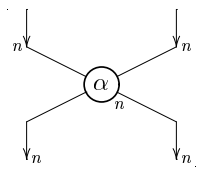

Fig. 7. Levels of a link.

Definition 8 (leveled structure) *An* ℓ-structure G *over the signature* $\Sigma \uplus \{!, ?\}$ *is a structure of links with a* level assignment $\ell_G : (\mathsf{vtx}(G) \cup \mathsf{lnk}\, G) \to \mathbb{N}$ *that associates a non-negative* level *to each vertex and each link of* G *s.t.: (*i*)* $\ell_G(v) = 0$*, for every* $v \in \mathsf{doors}(G)$*; (*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 Actually, labeling both arrows and links with a level is 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, link levels might be recovered from arrow levels, and vice versa. Both that choices correspond to solutions presented in literature. Labeling links is more faithful to the original presentation of optimal reduction algorithms (see [Lam90,GAL92a,GAL92b,AL94,Asp95]). Labeling 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* $e_! \in \mathsf{lnk}_!(G)$*. The* ℓ-box $B = \mathsf{box}^{\ell}_G(e_!)$ *of* $e_!$ *is the* G*-substructure s.t.:*

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

Note that we explicitly exploit connectedness of boxes.

In accord with previous notations, BX_G^ℓ is the set of boxes defined by the function box_G^ℓ, whereas 'the ℓ-structure G has the box nesting property' means that G with the boxes BX_G^ℓ 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 $\mathsf{BX}_G = \mathsf{BX}_G^\ell$.*

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 has the box nesting property, its levels might differ from the ones induced by the box nesting of a proof structure defined on the same structure of links of G. Nevertheless, that possibility has no impact on our study. Independently from the origin of the levels of G, it suffices that G 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 in which we are interested, we introduce a contraction operator. A *contraction* link, or $\curlyvee$-link, has $k+1$ doors: $k > 0$ *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 with the same name (in other words the contracting ports of a contraction are indistinguishable). In the following, we shall use $\mathsf{adoors}(e)$ to denote the set of the contracting (auxiliary) doors of a $\curlyvee$-link e and $\mathsf{pdoor}(e)$ to denote its contracted (principal) door.

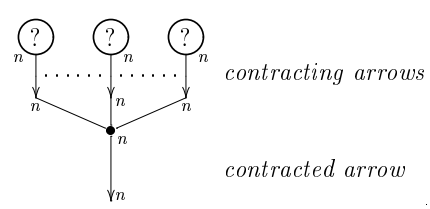

Fig. 8. Contraction link.

Because of the previous assumptions, the orientation of the $\curlyvee$-links must accord with the orientation of the ?-links—therefore, as in the following we shall meet ?-links whose external door is their conclusion only, 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 postulate that a ?-link be always followed by a ⋎-link, that implies inserting a dummy unary contraction link below some ?-links—the advantage is a simplification in the definition of sharing morphism, see Definition 17. A different possibility would have been to merge ? and ⋎-link (as for instance 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 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.

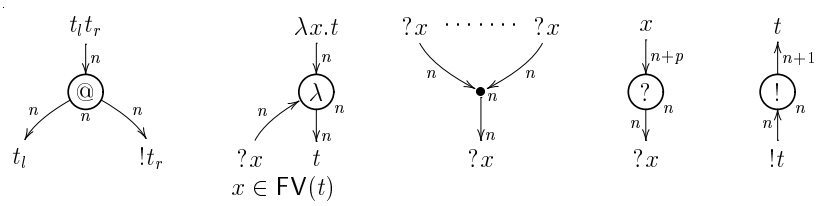

Fig. 9. $\lambda\ell$-terms.

Figure 9 gives the links for building the $\lambda\ell$-structures. The labels at the vertices of the arrows give the formation rules by which to build the correct $\lambda\ell$-structures, say the $\lambda\ell$-nets. The introduction of box door links and contraction implies a small change in the β-rule. In fact, the β-rule does not involve any longer an $@/\lambda$ pair only; the β-redex contains 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 also. Taking into account these considerations, the reformulation of the β-rule is immediate.

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 in our definition of box, weakening cannot be directly introduced (for a treatment of weakening see [GMM97a]). Hence, our MELL is without weakening.

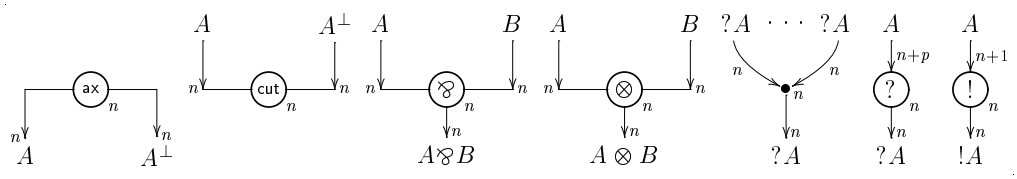

Fig. 10. MELL proof ℓ-nets.

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. However, the only one relevant for our purposes is 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 by means of the isomorphism $!O \multimap O \simeq O$; that, using $\parr$ in the place of $\multimap$, becomes $?I \parr O \simeq O$, where I and O are two constants s.t. $I = O^-$ and $O = I^-$. The links are the same as MELL, but formulas differ. There are four types of formulas only: I and $?I$, named inputs; and O and $!O$, named outputs. Any input formula is the dual of the corresponding output formula, *i.e.*, $I = O^-$, $?I = (!O)^-$, and $X^{--} = X$, for any X. The rules for link composition are given in Figure 11.

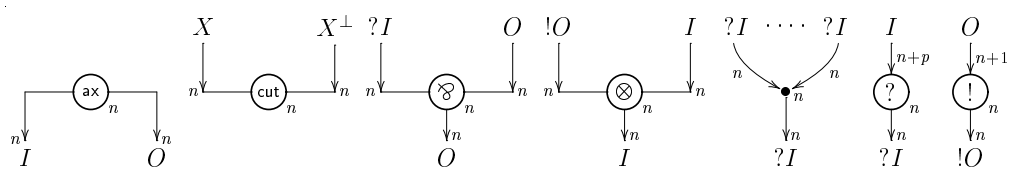

Fig. 11. Pure proof ℓ-nets.

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 $\parr$-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 wider than the one obtainable translating $\lambda\ell$-nets and is defined by the same correctness criteria used defining MELL proof ℓ-nets (see [Reg92]).

Nevertheless, not any pure proof ℓ-net can be interpreted as a MELL proof ℓ-net—for instance, the pure proof ℓ-net obtained by the translation of a $\lambda\ell$-net is a MELL proof net only if the corresponding λ-term is typable.

The rewriting rules are the same as MELL. Again, 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 that 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 those obtained by the reduction of 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 is a reserved name.

Definition 15 (multiplexer) *A kary 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) *a non-negative integer* m*, named the* threshold *of the mux;*
(iv) *a sequence of* k *integers* $q_1, \ldots, q_k$*, named 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 door of the mux—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). We see that:

– each offset q_i is the difference between the level of the ith auxiliary door

and the principal door of the mux;
– the threshold m is lower than the level of the principal door.

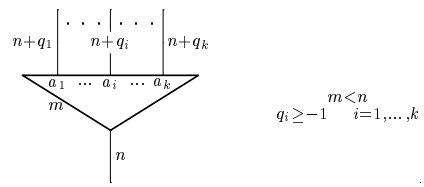

Fig. 12. The mux (multiplexer) link.

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 every $\alpha \in \Sigma$, 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 Gonthier's 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.

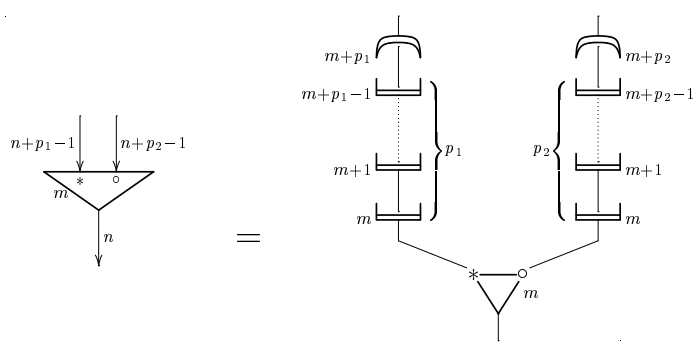

Fig. 13. Equivalence between muxes and brackets.

5 Unfolding the sharing

Definition 17 (sharing morphism) *An* s-morphism *(sharing morphism) is a surjective homomorphism of* $s\ell$*-structures* $M : G_0 \to G_1$ *whose restriction to* $\mathsf{doors}(G_0)$ *is injective. Namely,* M *is a map from* $\mathsf{lnk}(G_0) \uplus \mathsf{vtx}(G_0)$ *to* $\mathsf{lnk}(G_1) \uplus \mathsf{vtx}(G_1)$ *s.t.:*

(i) $M(\mathsf{lnk}(G_0)) = \mathsf{lnk}(G_1)$ *and* $M(\mathsf{vtx}(G_0)) = \mathsf{vtx}(G_1)$*;*
(ii) the links e *and* $M(e)$ *are of the same type;*
(iii) if the arrow v *is connected to a port with name* a *of the link* e*, then* $M(v)$ *is connected to the port with name* a *of the link* $M(e)$*;*
(iv) $\ell_{G_0}(x) = \ell_{G_1}(M(x))$*, for every* $x \in \mathsf{lnk}(G_0) \uplus \mathsf{vtx}(G_0)$*;*
(v) if $v, v' \in \mathsf{doors}(G_0)$ *and* $M(v) = M(v')$*, then* $v = v'$*.*

An s-morphism establishes a bijection between the doors of G_0 and G_1. Contraction and muxes are the unique links that may change cardinality via an s-morphism. Nevertheless, muxes and $\curlyvee$-links present a relevant difference: while it is impossible to equate two auxiliary ports of a mux via an s-morphism, we may have $M(v) = M(v')$, 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.

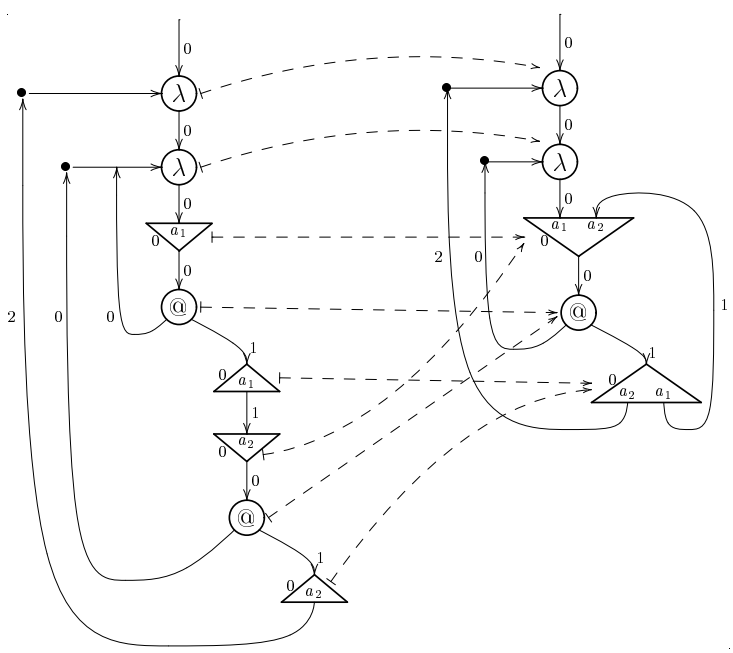

Fig. 14. Sharing morphism.

Proposition 18 *The* $s\ell$*-structures are a partial order w.r.t. the relation* $\preccurlyeq$ *defined by '*$G_1 \preccurlyeq G_2$ *when there exists an* s*-morphism* $M : G_1 \to G_2$*'.*

Proof. We need to show antisymmetry only. Namely, we suffice to prove that $G_1 \preccurlyeq G_2$ and $G_2 \preccurlyeq G_1$ implies $G_1 \simeq G_2$ (*i.e.*, G_1 and G_2 are isomorphic). Let $M_1 : G_1 \to G_2$ and $M_2 : G_2 \to G_1$. These two s-morphisms induce two corresponding pairs of surjective maps between $\mathsf{lnk}(G_1)$ and $\mathsf{lnk}(G_2)$ and between $\mathsf{vtx}(G_1)$ and $\mathsf{vtx}(G_2)$. Thus, $|\mathsf{lnk}(G_1)| = |\mathsf{lnk}(G_2)|$ and $|\mathsf{vtx}(G_1)| = |\mathsf{vtx}(G_2)|$; that is, both M_1 and M_2 are isomorphisms. $\square$

5.1 Unshared structures

A relevant case of $s\ell$-structures are the ones in which there is no sharing at all; that corresponds to the case in which all the muxes are unary. In our intention, that unshared structures are the minimal elements of the equivalence classes induced by $\preccurlyeq$.

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

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

Unfortunately, the relation $\preccurlyeq$ is not the unfolding ordering that we are looking for; the presence of contraction links might imply that a $u\ell$-structure is not minimal. For instance, let us take a $u\ell$-structure U such that all its doors are conclusions of a contraction; there exists a denumerable set of $u\ell$-structures U' s.t. $U' \preccurlyeq U$—the kth of them can be built making k instances of U and merging the $\curlyvee$-links above each k-tuple of 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 $s\ell$-structure G.

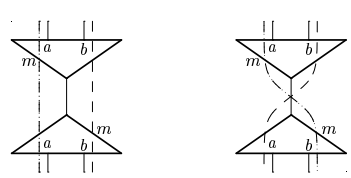

Fig. 15. Matching of ports.

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 must be a perfect matching between

the ports of e_1 and the ports of e_2; moreover, every 'correct' unfolding of G must preserve that matching. 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. $M(e'_1) = e_1$ and $M(e'_2) = e_2$ must match. Referring to Figure 15, each pair e'_1, e'_2 in a correct unfolding of G must be the image of one of the paths drawn on the left-hand side pair of muxes; while, every $G' \preccurlyeq G$ containing an image of one of the paths drawn on the right-hand side pair of muxes is an uncorrect unfolding of G.

6 Overview of the main results

The algebraic semantics that we are going to present in section 7 formalizes the matching problem mentioned in the previous section. That semantics is an algebraic characterization of the *proper $u\ell$-structures*—the only $u\ell$-structures that are correct unfoldings of sharing graphs. At the same time, via a suitable restriction of the s-morphisms, we get a characterization of the *proper $s\ell$-structures*—the $s\ell$-structures for which there exists a correct unfolding (*i.e.*, s.t. there is a proper $u\ell$-structure among their less-shared-instances). In more details, once defined the proper $u\ell$-structures, we shall restrict $\preccurlyeq$ to a partial order $\prec\!\!\prec$ (read 'is a proper unfolding of'), and we shall say that an $s\ell$-structure G is proper when $U \prec\!\!\prec G$ for some proper $u\ell$-structure U.

In section 8 we shall show that the binary relation $\prec\!\!\prec$ is the right unfolding partial order. In particular, we shall prove that:

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

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

The uniqueness of the least-shared-instance of a proper $s\ell$-structure will allow us to semantically associate an ℓ-structure to each proper $s\ell$-structure. Namely, the notion of *solution* of a $u\ell$-structure, that we shall introduce defining properness, will give us a natural way to *semantically read-back* an ℓ-structure $\mathscr{R}(U)$ from a proper $u\ell$-structure U; then, the semantical read-back

of a proper $s\ell$-structure is 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 shall show that the π-normal form of a proper $s\ell$-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

As an introduction to the algebraic part and to the terminology used in it, the next section describes the naive attempts at the origin of our approach.

7.1 Lifting functions

It is natural to interpret a $u\ell$-structure U as a representation of the structure N obtainable by replacing each lift of U with a wire. Nevertheless, it is readily seen that this procedure erases the boxes of U (if any), for in general there is not 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. Since an arrow might be in the scope of several lifts (for the sake of simplicity, 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 that of boxes), we need a general way for reasoning about the displacement of levels that takes place at a given arrow. Let us assume that each arrow yields a function $f_v : \mathbb{Z} \to \mathbb{Z}$ assigning a local name to each level; that is, let $f_v(n)$ be the actual value of the level n at the arrow v. By the way, in order to get a sound renaming at each arrow, we should impose some restrictions on the functions f, as for instance monotonicity. Anyhow, let us go on without too many details, our purpose is to show why such an approach is too weak.

The ℓ-structures are the base case. In them, there is no reindexing at all, for there are no lifts around. Thus, every arrow of an ℓ-structure yields the identity function. In the $u\ell$-structures instead, interpreting a lift $e_\triangledown$ as a reindexing (or shifting) operator for the levels above its threshold leads to associate to $e_\triangledown$ a

functional $\delta_{m,q} : \mathbb{Z}^{\mathbb{Z}} \to \mathbb{Z}^{\mathbb{Z}}$ (where m is the threshold and q is 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

$$f_{v_p} = \delta_{m,q}(f_{v_a})$$

where $v_p = \mathsf{pdoor}(e_\triangledown)$ and $\{v_a\} = \mathsf{adoors}(e_\triangledown)$. The shape of the functions assigned to the arrows v of U is thus

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

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

Provided to succeed assigning a function f_v to each arrow v, we see that $f_{v_p}(\ell(v_p)) = f_{v_a}(\ell(v_a))$ for every lift $e_\triangledown$. Hence, given a $u\ell$-structure U, we may read-back an ℓ-structure N by assigning the level $f_v(\ell_U(v))$ to each arrow of U and removing the lifts contained in it. (For the sake of completeness, some other constraints should be added in order to ensure the constraints on the levels of the ? and !-link doors.)

The latter procedure is at the base of the semantical read-back that we shall give in section 7.9. Nevertheless, the use of the functionals δ is still inadequate for our purposes: 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 is the case, 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, that is the algebraic encoding of the fact that a pair of matching lifts is equivalent to an arrow between its auxiliary ports.

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

equations:

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

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

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

provided that $m_1 < m_2$.

Remark 21 Equation (LO1) encodes the matching problem described in section 5.2. Equation (LO2) corresponds to the idea that, when $m_1 < m_2$, lifting the levels above m_1 by q_1 and the levels above m_2 by q_2 is the same as lifting 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 it is true when $q_i \geq -1$ also; it corresponds to the functional 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 by applying two different operators; that is, $\mathcal{L}[m_1, q_1, a_1](d_1) = d_0 = \mathcal{L}[m_2, q_2, a_2](d_2)$. It aims at forcing the uniqueness of d_0, modulo Equation (LO3); in particular, it ensures that $d_0 = \mathcal{L}[m_1, q_1, a_1]\, \mathcal{L}[m_2, q_2, a_2](d)$, for some d.

We want to find a semantics assigning to each arrow of a $u\ell$-structure a product like the one of equation (4), in which the lifting operator $\mathcal{L}[m, q, a]$ takes the place of the integer functional $\delta_{m,q}$. For this purpose, we need an axiomatization of lifting operators and of their inverse functionals (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 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 every $d \in \mathsf{dom}(\mathcal{F})$).

In more details, for any product of lifting operators, we have that:

$$(\mathcal{L}[m_1, q_1, a_1] \cdots \mathcal{L}[m_k, q_k, a_k])^* = \overline{\mathcal{L}}[m_k, q_k, a_k] \cdots \overline{\mathcal{L}}[m_1, q_1, a_1],$$

where

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

(since $-\notin \mathbb{D}$, $\mathcal{F}(d) =-$ is a denotation for $d \notin \mathsf{dom}(\mathcal{F})$).

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

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

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

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

7.4 The inverse semigroup LSeq^*

First, a remind on 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 every* $\mathcal{F}$*) closed under an involution operation* $(\cdot)^*$ *s.t., for any* $\mathcal{F}, \mathcal{F}_1, \mathcal{F}_2 \in \mathbb{S}$:

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

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

The axioms of an inverse semigroup allow to prove that:

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

$$\begin{aligned} \langle \mathcal{F} \rangle^* &= \langle \mathcal{F} \rangle \\ \langle \mathcal{F} \rangle \, \langle \mathcal{F} \rangle &= \langle \mathcal{F} \rangle \end{aligned}$$

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^* has the shape $\langle \mathcal{F} \rangle$, for some $\mathcal{F}$, and is then invariant under involution, *i.e.*,

$$\begin{aligned} \mathcal{F} \, \mathcal{F} = \mathcal{F} \quad & \textit{implies} \quad \langle \mathcal{F} \rangle = \mathcal{F} \\ \mathcal{F} \, \mathcal{F} = \mathcal{F} \quad & \textit{implies} \quad \mathcal{F}^* = \mathcal{F} \end{aligned}$$

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 $\overline{\mathcal{L}}[m, q, a]$ named 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:*

$$\mathcal{L}[m, q, a]^* = \overline{\mathcal{L}}[m, q, a] \tag{LS0}$$

$$\overline{\mathcal{L}}[m, q, a] \, \mathcal{L}[m, q, a] = 1 \tag{LS1}$$

$$\overline{\mathcal{L}}[m, q_2, a_2] \, \mathcal{L}[m, q_1, a_1] = 0 \qquad \textit{if } q_1 \neq q_2 \textit{ or } a_1 \neq a_2 \tag{LS2}$$

$$\mathcal{L}[m_2, q_2, a_2] \, \mathcal{L}[m_1, q_1, a_1] = \mathcal{L}[m_1, q_1, a_1] \, \mathcal{L}[m_2 + q_1, q_2, a_2] \tag{LS3}$$

$$\overline{\mathcal{L}}[m_2, q_2, a_2] \, \mathcal{L}[m_1, q_1, a_1] = \mathcal{L}[m_1, q_1, a_1] \, \overline{\mathcal{L}}[m_2 + q_1, q_2, a_2] \tag{LS4}$$

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 (LS1) and (LS2) are equivalent to equation (LO1). Axiom (LS3) coincides with equation (LO2). Axiom (LS4) corresponds to the *only if* part of equation (LO3) (the *if* part is subsumed by equation (LO3)).

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 $\mathcal{H} \in \mathsf{LSeq}$, there exists a sequence of lifting operators $\prod_{0<i\leq k} \mathcal{L}[m_i, q_i, a_i] = \mathcal{H}$, with $m_i \leq m_j$ if $i < j$.

By induction on the length $|\mathcal{H}| = k$ of the lifting sequence $\mathcal{H}$, we 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 $n_0 \leq n_1 \leq n_2$.*

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

(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* $\mathcal{H} \in \mathsf{LSeq}[n_0, n_2]$, *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 $\mathcal{H} = \prod_{0<i\leq k} \mathcal{L}[m_i, q_i, a_i]$ be in canonical form. Let h be the first index for which $n_1 + Q_i \geq m_i$ (where $Q_i = \sum_{0\leq j<i} q_j$) 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 $\mathcal{H}_2 = \prod_{h\geq i<k} \mathcal{L}[m_i - Q_h, q_i, a_i]$. By construction, $\mathcal{H}_1 \in \mathsf{LSeq}[n_0, n_1]$ 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 previous construction ensures the uniqueness of $\mathcal{H}_1$ and $\mathcal{H}_2$ also. $\square$

Proposition 27 (canonical form) *For any* $\mathcal{F} \in \mathsf{LSeq}^*$, *with* $\mathcal{F} \neq 0$, *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 $\mathcal{H}_1 \in \mathsf{LSeq}[0, m]$ and $\mathcal{H}_2 \in \mathsf{LSeq}[m, \omega]$ s.t. $\mathcal{H}_+ = \mathcal{H}_2\, \mathcal{H}_1$. If $\mathcal{H}_2 \in \mathsf{LSeq}[m+1, \omega]$, then $\mathcal{F} = \overline{\mathcal{L}}[m, q, a]\, \mathcal{H}_+\, \mathcal{H}_-^* = \mathcal{H}_2^{\uparrow q}\, \overline{\mathcal{L}}[m, q, a]\, \mathcal{H}_1\, \mathcal{H}_-^* = \mathcal{H}_2^{\uparrow q}\, \mathcal{H}_1\, (\mathcal{H}_-\, \mathcal{L}[m, q + ||\mathcal{H}_1||, a])^*$. Otherwise, $\mathcal{H}_2 = \mathcal{L}[m, q, a]\, \mathcal{H}_2'$ with $\mathcal{H}_2' \in \mathsf{LSeq}$ (by the hypothesis $\mathcal{F} \neq 0$); therefore, $\mathcal{F} = \mathcal{H}_2'\, \mathcal{H}_1 \mathcal{H}_-^*$.

(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. $q \neq q'$ or $a \neq a'$, we would get $1 = \overline{\mathcal{L}}[m, q', a']\, \mathcal{H}_1\, \mathcal{H}_2^*\, \mathcal{L}[m, q', a'] = 0$. Hence, $|\mathcal{H}_1| = 0$, 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 $|\mathcal{H}_-| = 0$. Therefore, $\mathcal{H}_1^*\, \mathcal{H}_2 = 1$ and $\mathcal{H}_2 = \langle \mathcal{H}_2 \rangle\, \mathcal{H}_2 = \langle \mathcal{H}_1 \rangle\, \mathcal{H}_2 = \mathcal{H}_1$.

Now, let $\mathcal{H}_+\, \mathcal{H}_-^*$ and $\widehat{\mathcal{H}}_+\, \widehat{\mathcal{H}}_-^*$ be two canonical forms of $\mathcal{F}$. We have $\langle \mathcal{H}_+ \rangle = \langle \mathcal{F} \rangle = \langle \widehat{\mathcal{H}}_+ \rangle$; that implies $\mathcal{H}_+ = \widehat{\mathcal{H}}_+$ and $\mathcal{H}_- = \widehat{\mathcal{H}}_-$. Hence, we suffice to show the uniqueness of the canonical form of any $\mathcal{H} \in \mathsf{LSeq}$. The case $\mathcal{H} = 1$ is a consequence of claim (i). 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 $m_1 < m_2$, let us take a

pair q, a s.t. $q \neq q_1$ or $a \neq a_1$; then, $0 = \overline{\mathcal{L}}[m_1, q, a]\,\mathcal{H} = \widehat{\mathcal{H}}\,\overline{\mathcal{L}}[m_1, q, a]$, where $\widehat{\mathcal{H}} = \mathcal{L}[m_2 + q_1, q_2, a_2]\,\mathcal{H}_2^{\uparrow q_1}$; that leads to the contradiction $1 = \widehat{\mathcal{H}}^*\,\widehat{\mathcal{H}}\,\overline{\mathcal{L}}[m_1, q, a]\,\mathcal{L}[m_1, q, a] = 0$. Hence, we must have $m_1 = m_2$, with $q_1 = q_2$ and $a_1 = a_2$ also (otherwise, we would get the contradiction $1 = \mathcal{H}^*\,\mathcal{H} = 0$ again). Concluding, from the initial canonical forms, we have got two shorter canonical forms $\mathcal{H}_1 = \mathcal{H}_2$; therefore, by the induction hypothesis, we conclude. $\square$

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 In the proof of Proposition 27, $1 \neq 0$ is a key assumption on LSeq^*.

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

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, $\mathcal{H}_2 = \mathcal{H}_1\,\mathcal{H}_1^*\,\mathcal{H}_2 = \mathcal{H}_1\,\mathcal{H}_+\,\mathcal{H}_-^*$, which implies $\mathcal{H}_-^* = 0$. 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 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 $i = 1,2$; $\mathcal{H} \sqsubseteq \mathcal{H}_i$ 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.*

The operator $\sqcap$ 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 every* $v \in \mathsf{vtx}(U)$*;*
(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.*, of a ? or of 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.*

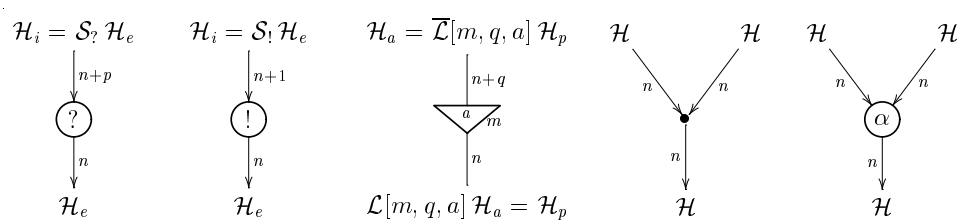

Fig. 16. Lifting assignment ($\mathcal{S}_? \in \mathsf{LSeq}[n, n+p]$ and $\mathcal{S}_! \in \mathsf{LSeq}[n, n+1]$).

The second and third items of the previous definition are compatible with the first one (by Fact 26); since $\mathsf{LSeq}[n,n] = \{1\}$, the fourth item is a special case of the second one. Further, for any box door link e, the parameter $\mathcal{S}_e$ for which

$\mathscr{A}(v_i) = \mathcal{S}_e\,\mathscr{A}(v_e)$ is uniquely determined by the lifting sequences assigned to the internal and external doors v_i and v_e (it is indeed determined by $\mathscr{A}(v_i)$ only, see Fact 26).

Remark 33 The constraints on the assignments at the box door links encode that box door links are global boundaries for the scope of the reindexing operators associated to lifts. Moreover, the introduction of the parameters $\mathcal{S}_e$ 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 becomes 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 $n_0 \leq n_1 + ||\mathcal{H}||$, by Fact 26. Hence, interpreting $||\mathcal{H}||$ as the global quantity by which we increase the levels above $n_1 - 1$, no level inside a box is lower than the levels of the box doors.) Anyhow, there is 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 for the interior of $\mathsf{box}^\ell(e_!)$. Whereas, the reindexing parameter of a ?-link is not independent. In fact, a ?-link may only erase the reindexing operators introduced either by the principal doors of the boxes of which the link is auxiliary door or by reductions executed inside such boxes. We shall farther see that this has a direct correspondence in the dynamics of boxes that we shall discuss in section 10 and in the read-back reduction system that we shall introduce in section 8.

7.8 Solutions of a $u\ell$-structure

In the relevant cases of λ-calculus and MELL, the topology of the corresponding structures forces the uniqueness of the lifting assignment (if any), once fixed the parameters $\mathcal{S}_!$. This is false in the general case, in which we must use the $\sqcap$ operator for pruning the parts of a lifting assignment that do not correspond to lifts of the $u\ell$-structure or to factors of any $\mathcal{S}_!$.

A map $\mathscr{S}$ from the !-links of a $u\ell$-structure U to LSeq is an *internal state* of U when $\mathscr{S}(e_!) \in \mathsf{LSeq}[\ell(\mathsf{pdoor}(e_!)), \ell(\mathsf{pdoor}(e_!)) + 1]$. By what previously said on the parameters $\mathcal{S}_e$, any lifting assignment determines a unique internal state s.t. $\mathscr{S}(e) = \mathcal{S}_e$.

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

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

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

and to the internal states of U (in the previous equations, replace the vertices of U with the !-links of U).

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

Proof. Let e be an !-link. We have $\mathscr{A}_1(v_e) \sqcap \mathscr{A}_2(v_e) = (\mathscr{S}_1(e) \sqcap \mathscr{S}_2(e))\,(\mathscr{A}_1(v_e) \sqcap \mathscr{A}_2(v_e))$ (by Fact 31); and so on for the other links. $\square$

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 $U\}$ is closed under the meet operation and has a minimum.

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

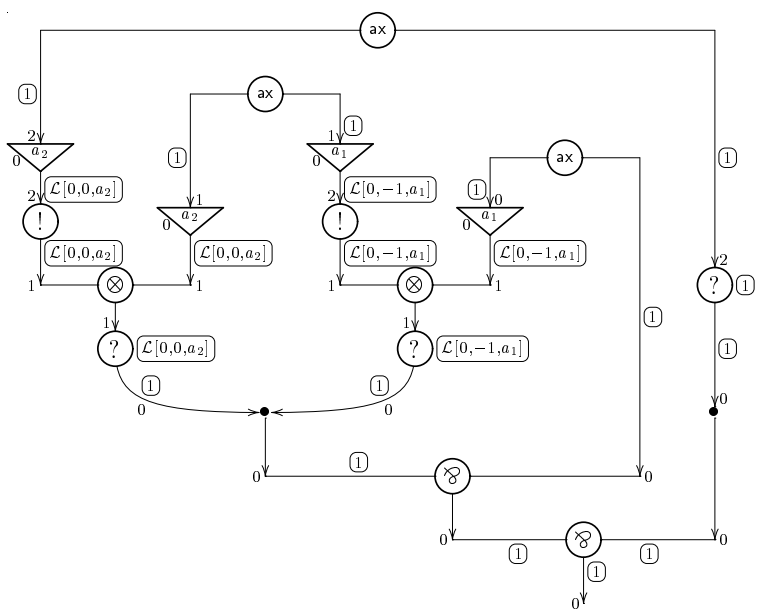

Fig. 17. An example of quiescence solution.

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

7.9 Semantical read-back

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

(i) $\ell_U(v) + ||\mathscr{A}(v)|| \geq 0$, for every $v \in \mathsf{vtx}(U)$;
(ii) $\ell_U(v_e) + ||\mathscr{A}(v_e)|| \leq \ell_U(v_i) + ||\mathscr{A}(v_i)||$, when v_i and v_e are the internal and the external door of a box door link e, respectively;
(iii) $\ell_U(v_p) + ||\mathscr{A}(v_p)|| = \ell_U(v_a) + ||\mathscr{A}(v_a)||$, when $v_p = \mathsf{pdoor}(e_\triangledown)$ and $\{v_a\} = \mathsf{adoors}(e_\triangledown)$ for some lift $e_\triangledown$.

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

(i) associating to each arrow v its actual level *$\ell^{\mathscr{I}}_U(v) = \ell_U(v) + ||\mathscr{Q}(v)||$;*
(ii) replacing each lift $e_\triangledown$ and its doors v_p and v_a by a unique arrow with level $\ell^{\mathscr{I}}_U(v_p) = \ell^{\mathscr{I}}_U(v_a)$.

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

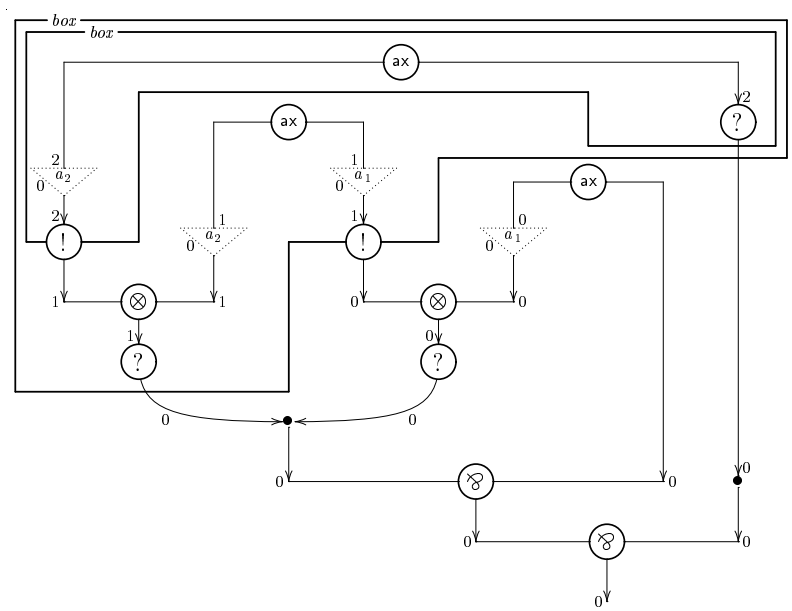

Fig. 18. Read-back.

7.10 Proper structures

We are now able to characterize the 'proper' unfoldings of an $s\ell$-structure.

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

(i) $M : U \preccurlyeq G$;
(ii) *U has an $\mathscr{S}$-solution for every internal state $\mathscr{S}$;*
(iii) *if $\mathscr{A}$ is a solution of U, then $M(v) = M(v')$ and $\mathscr{A}(v) = \mathscr{A}(v')$ implies $v = v'$, for every $v, v' \in \mathsf{vtx}(U) \setminus \bigcup_{e_{\Upsilon} \in \mathsf{lnk}_{\Upsilon}(U)} \mathsf{adoors}(e_{\Upsilon})$.*

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

Remark 39 Requiring the existence of a solution for every internal state accords with the idea that the reindexing parameters of the !-links are independent parameters.

Definition 40 (proper $s\ell$-structure) *An $s\ell$-structure is* proper *when it has a complete unfolding.*

Remark 41 According to Definition 37, the partial order $\prec\!\!\!\prec$ is flat: a proper ℓ-structure is related only with its least-shared-instance (later, we shall prove that the least-shared-instance of a proper $s\ell$-structure is unique). However, the relation $\prec\!\!\!\prec$ could be extended assuming that $G_1 \prec\!\!\!\prec G_2$ (read G_1 is a *proper unfolding* of G_2) when G_1 and G_2 are proper $s\ell$-structures with the same least-shared-instance and $G_1 \preccurlyeq G_2$. Besides, in our study, we shall meet the case in which G_1 is a $u\ell$-structure only. For this reason, in the following we shall use $U \prec\!\!\!\prec G$ to denote that U is the least-shared-instance of G.

Any $u\ell$-structure with an $\mathscr{S}$-solution for every internal state $\mathscr{S}$ is proper, *e.g.*, the $u\ell$-structure in Figure 17 is proper and the identity s-morphism is its complete unfolding. Another example of proper $s\ell$-structure is given in Figure 19; its least-shared-instance is the $u\ell$-structure in Figure 17.

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 that we are going to introduce in the next section.

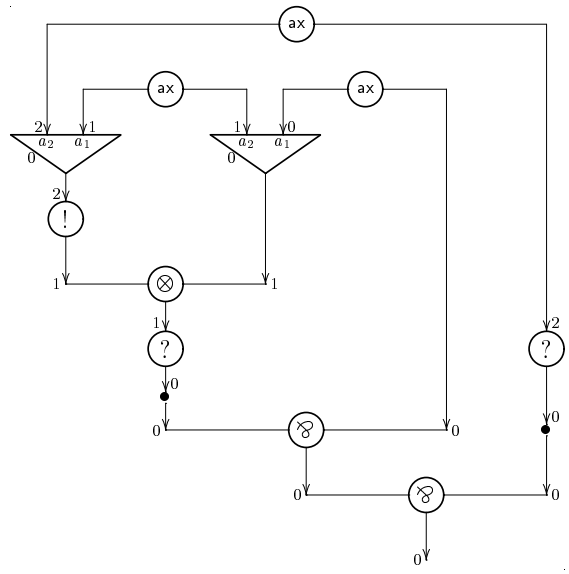

Fig. 19. A proper $s\ell$-structure.

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 $U \prec\!\!\!\!\prec G$. The* read-back *of G is the ℓ-structure $\mathscr{R}(G) = \mathscr{R}(U)$.*

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 (for the sake of clarity, all the muxes in the picture are binary; the extension to the $k$ary case is direct). The π-rules implement the interpretation of lifts that guided us so far, taking into account the additional duplicating task of k-muxes. 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 schemata valid independently from the orientation of the arrows (*e.g.*, in the !/?-propagation the principal door of the mux may be the internal door of the

box door link).

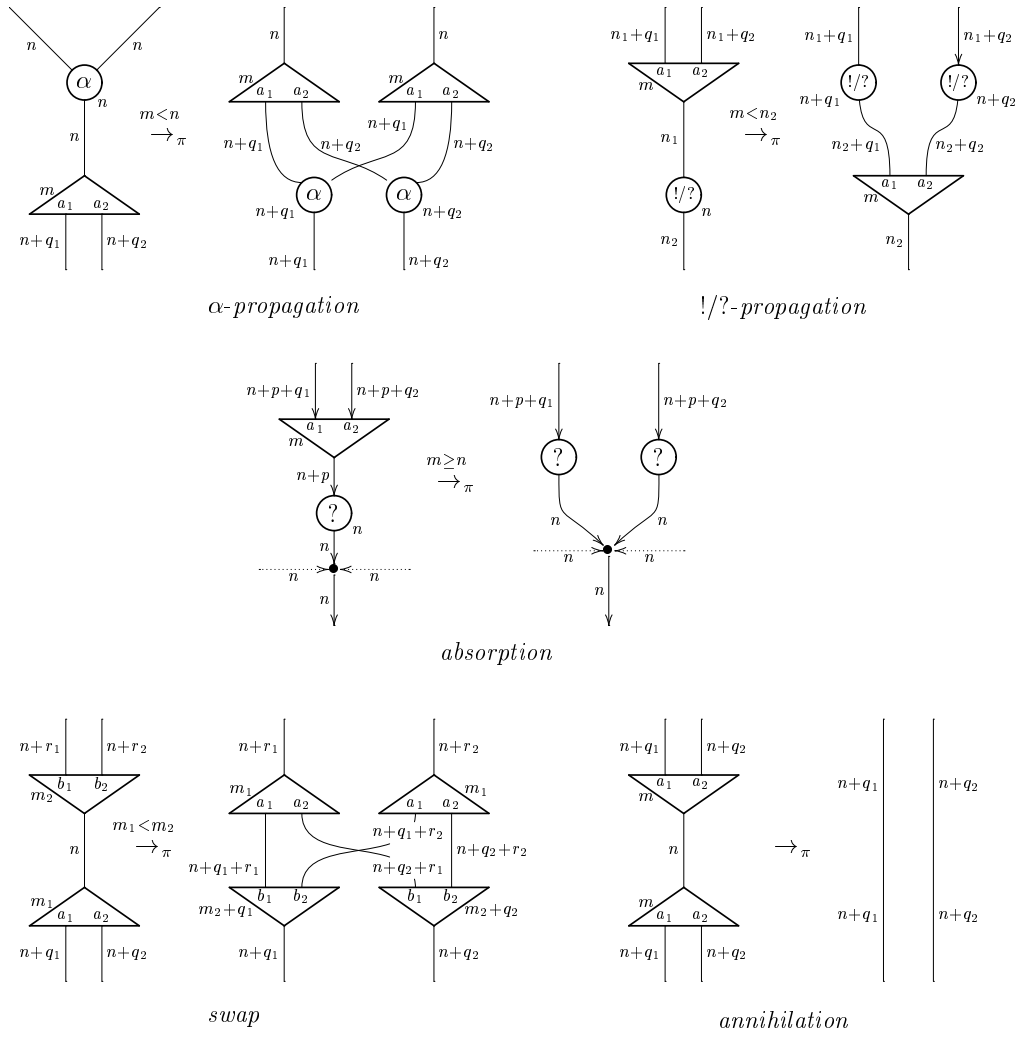

Fig. 20. Read-back rewriting system (π-rules).

The *propagation* rules correspond to the idea that a mux has to reindex and duplicate every link that 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 $e_\triangledown$. The proviso stress that in an α-propagation the mux is acting on a link whose level is above its threshold.)

The proviso on the propagation rules forbids their application when 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 Remark 33 also). Hence, the missing !-link case must 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 the mux thresholds are a machinery for matching 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 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.

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 $r : U \to_\pi U'$. 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. $\mathscr{S}'(e) = \mathscr{S}(e)^{\uparrow q}$, when $e = e_!$, and $\mathscr{S}'(e) = \mathscr{S}(e)$, otherwise. $\square$

Lemma 44 *Let G_0 be a proper $s\ell$-structure and let $U_0 \prec\!\!\!\prec G_0$. For every $G_0 \to_\pi G_1$, there exists $U_0 \to_\pi^+ U_1$ s.t. $U_1 \prec\!\!\!\prec G_1$.*

Proof. Let $M_0 : U_0 \prec\!\!\!\prec G_0$ and let r be a redex of G_0. The counterimage of r is a set of redexes $M_0^{-1}(r)$ that may contain a case of critical pair only: two (or more) lifts whose principal doors are contracting doors of the same $\curlyvee$-link. When r is an absorption, any of that 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 also, all the critical pairs are confluent. Hence, let us execute in any order the redexes in the set $M_0^{-1}(r)$, closing the critical pairs present in it. The result is the proper $u\ell$-structure U_1

(by Lemma 43, U_1 is proper). The s-morphism between $M_1 : U_1 \prec\!\!\!\prec G_1$ is the one induced by the function mapping every residual of $e \in \mathsf{Ink}(U_0)$ into the corresponding residual of $M_0(e)$. $\square$

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 $G \to_\pi G'$, we have that:*

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

Proof.

(i) It is an immediate consequence of Lemma 43 and Lemma 44.
(ii) (*if*) Let $U_1 \prec\!\!\!\prec G \succ\!\!\!\succ U_2$. By Lemma 44, $U_i \to_\pi^+ U'$ by a sequence of reductions corresponding to $G \to_\pi G'$. By inspection of the rules, we see that this is possible only if $U_1 = U_2$. (*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 $\mathscr{R}(U) = \mathscr{R}(U')$. $\square$

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 $G \to_\pi G'$, 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 $u\ell$-structure. To lift the result to the $s\ell$-structures, let us note that the simulation property can be reversed. Namely, if $G \to_\pi G'$ and $U' \prec\!\!\!\prec G'$, then there is $U \prec\!\!\!\prec G$ s.t. $U \to_\pi^+ U'$. $\square$

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 $s\ell$-structures (see Proposition 49). To prove such a result, let us start proving that local

confluence holds, modulo some mux permutations, for the $s\ell$-structures in which each mux eventually interacts with some other link (Lemma 48).

Let $e_\triangledown$ be a mux whose principal door is not an auxiliary door of another mux. We say that $e_\triangledown$ forms a *deadlock* when no π-rule can be applied to it.

Fact 47 (deadlock-freeness) *Let us say that an $s\ell$-structure G is* deadlock-free *when there is no $G \to_\pi^* G'$ s.t. G' contains a deadlock. Any proper $s\ell$-structure is deadlock-free.*

Proof. Let $U \prec\!\!\!\prec G$. 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. Now, it is readily seen 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 $s\ell$-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 locally confluence. Nevertheless, let us note that, when the $s\ell$-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 $s\ell$-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. Anyhow, there are two reductions $G \xrightarrow{r_1}_\pi \to_\pi^+ G_1$ and $G \xrightarrow{r_2}_\pi \to_\pi^+ G_2$ s.t. G_1 and G_2 coincide modulo the permutation of muxes in Figure 21.

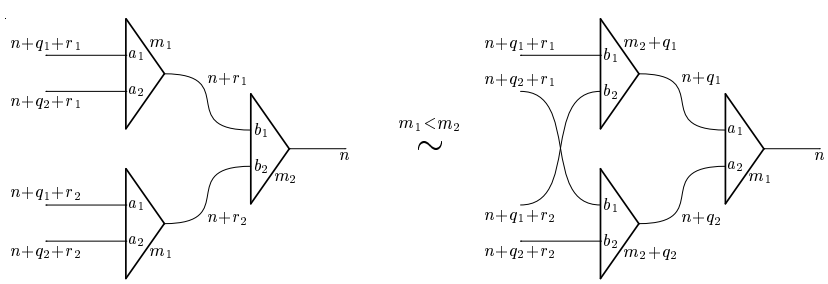

Fig. 21. Mux permutation equivalence.

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.

properness and unfolding. In fact, if $G_1 \sim G_2$, then $U_1 \sim U_2$, where $U_1 \prec\!\!\prec G_1$ and $U_2 \prec\!\!\prec G_2$.

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

$$\begin{tikzcd}
 & G_0 \sim H_0 \arrow[dl, "\pi"'] \arrow[dr, "\pi"] & \\
G_1 \arrow[dr, "\pi"', "*"] & & H_1 \arrow[dl, "\pi"', "*"] \\
 & G_2 \sim H_2 &
\end{tikzcd}$$

commutes for any pair of proper $s\ell$-structures G_0, H_0 and for any pair of π-reductions $G_0 \to_\pi G_1$, $H_0 \to_\pi H_1$.

Sketch of the proof. The complete proof of the lemma requires 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 of muxes $T[G_0] \sim T[H_0]$ that, w.l.o.g., we may assume to be trees of muxes. When $r : G_0 \to_\pi G_1$ and $s : H_0 \to_\pi H_1$ does not involve the root mux of any of that trees, the lemma follows from the already remarked property that, when $G_0 = H_0$, the diagram commutes. In the other cases, let $T[G_0]$ (or, mutatis mutandis, $T[H_0]$) be the tree whose root mux form the redex r with a link e. There is a reduction $R : G_0 \to_\pi^* G'$ s.t. all the muxes of $T[G_0]$ interacts with a residual of e (assume that, when a residual of e annihilates interacting with a mux e' of $T[G_0]$, the interaction is done for all the muxes of $T[G_0]$ following e'). Since r and s are the first reductions of the corresponding sequences R and S, the proof concludes showing that the diagram still commutes after replacing R and S for r and s. $\square$

We stress that the previous lemma could be reformulated replacing deadlock-free $s\ell$-structures for proper $s\ell$-structures, since deadlock-freeness is the only property of the proper $s\ell$-structures required by 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 $L = \mathsf{lnk}(U) \setminus \mathsf{lnk}_{\triangledown}(U)$;
(b) $k_2 = \sum_{e \in \mathsf{lnk}_{\triangledown}(U)} |\mathscr{Q}(\mathsf{pdoor}(e))|$.
Any propagation or absorption rule decreases k_1, but may increase k_2; any mux rule decreases k_2. Summing up, any π-rule decreases the pair (k_1, k_2) (w.r.t. the lexicographic ordering). Therefore, there is no infinite π-reduction of U. To conclude, let us take $U \prec\!\!\!\!\prec G$; by Lemma 44, it follows 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. Therefore, N is unique. $\square$

Proposition 50 *Any proper $s\ell$-structure G has a unique least-shared-instance and $\mathscr{R}(G)$ 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, $M(v) = M(v')$ iff $v = v'$ for any $M : N \prec\!\!\!\!\prec G$, etc. As a consequence, by the fact that the π-normal form of G is a proper ℓ-structure N and Proposition 45, G has a unique least-shared-instance U and $\mathscr{R}(G) = \mathscr{R}(U)$; moreover, N is the π-normal form of U also. By the invariance of the read-back under π-reduction (Proposition 45 again), we conclude that $N = \mathscr{R}(N) = \mathscr{R}(U) = \mathscr{R}(G)$. $\square$

9 Properness and box nesting property

In this section, we shall see that, in order to get a proper structure, the box nesting property is a necessary requirement. Moreover, we shall see that properness is 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 $\mathscr{S}_0, \mathscr{S}_1, \ldots, \mathscr{S}_k$ defined in the following way: $\mathscr{S}_i(e_!) = \mathscr{S}(e_!)$, when $\ell_U(e_!) < i$, and $\mathscr{S}_i(e_!) = 1$, otherwise. By construction, $\mathscr{I} = \mathscr{S}_0$ and, if k is greater or equal than the maximal level in the ℓ-structure, $\mathscr{S} = \mathscr{S}_k$. 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 $v \in \mathsf{box}^\ell_U(e_!)$ for some $e_! \in \mathsf{lnk}_!(U)$ s.t. $\ell_U(e_!) = i$, and $\mathscr{A}_{i+1}(v) = \mathscr{A}_i(v)$, otherwise. This construction is sound because of the box nesting property; moreover, each $\mathscr{A}_i$ is the $\mathscr{S}_i$-solution of U.

(only if) Let $e_1, e_2 \in \mathsf{lnk}_!(U)$ be s.t. $\mathsf{box}^\ell_U(e_1) \cap \mathsf{box}^\ell_U(e_2) \neq \varnothing$. W.l.o.g., let $\ell_U(e_1) \leq \ell_U(e_2)$. By the definition of box^ℓ_U, we see that $v \in \mathsf{vtx}(\mathsf{box}^\ell_U(e_2))$ implies $v \in \mathsf{vtx}(\mathsf{box}^\ell_U(e_1))$. Hence, $\mathsf{box}^\ell_U(e_2) \subseteq \mathsf{box}^\ell_U(e_1)$. When $\ell_U(e_1) = \ell_U(e_2)$, let $\mathscr{S}$ be an internal state of U s.t. $\mathscr{S}(e) \neq 1$ iff $e \in \{e_1, e_2\}$. We see that $\mathsf{box}^\ell_U(e_1) = \mathsf{box}^\ell_U(e_2)$, since $e_2 \in \mathsf{dlinks}(\mathsf{box}^\ell_U(e_1))$. In this case, U has an $\mathscr{S}$-solution iff $\mathscr{S}(e_1) = \mathcal{F} = \mathscr{S}(e_2)$ (this solution assigns $\mathcal{F}$ to each arrow internal to $\mathsf{box}^\ell_U(e_1) = \mathsf{box}^\ell_U(e_2)$ and 1 to the other arrows of U). Therefore, $e_1 = e_2$. When $\ell_U(e_1) < \ell_U(e_2)$, then $e_2 \in \mathsf{box}^\ell_U(e_1) \setminus \mathsf{dlinks}(\mathsf{box}^\ell_U(e_1))$. Therefore, $\mathsf{box}^\ell_U(e_1) \neq \mathsf{box}^\ell_U(e_2)$. $\square$

The maps box^ℓ_U and BX^ℓ_U extends to the case in which U is a proper $u\ell$-structure replacing the actual level function $\ell^{\mathscr{I}}_U$ for ℓ_U in Definition 11—we remind that, when U is an ℓ-structure, $\ell^{\mathscr{I}}_U = \ell_U$. According to this, we shall say that a $u\ell$-structure U has the box nesting property when its boxes BX^ℓ_U verify the constraints in Definition 7. Anyhow, the boxing functions, the box nesting property and the read-back $\mathscr{R}(U)$ of a $u\ell$-structure U (we remind that, for the computation of $\mathscr{R}(U)$, we suffice the existence of the quiescence solution of U) are defined only if U has a quiescence solution.

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 $U \to_\pi U'$. There is a one-to-one correspondence f between the links of U that are not lifts and the links of U' that are not lifts. By inspection of the π-rules, we see that, for any $e \in \mathsf{lnk}(U) \setminus \mathsf{lnk}_\triangledown(U)$ and any $e_! \in \mathsf{lnk}_!(U)$, $f(e) \in \mathsf{box}^\ell_{U'}(f(e_!))$ iff $e \in \mathsf{box}^\ell_U(e_!)$. Therefore, 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 $\mathscr{R}(U)$. Proving the only if part, we have already seen that the box nesting property is invariant under π-reduction. Thus, $\mathscr{R}(U)$ has the box nesting property and, by Proposition 51, it is proper. Finally, the properness of U 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 with the box nesting property.*

Proof. By the invariance of properness (Lemma 46), an $s\ell$-structure is proper iff its π-normal form N is proper. Since 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. Therefore, by Proposition 51, we conclude. $\square$

The latter propositions have two important technical consequences. The first one relates the level that any $\mathscr{S}$-solution $\mathscr{A}$ assigns to the internal arrows of a box to the level that $\mathscr{A}$ assigns to the internal door of the principal door link of the box (as for the actual levels induced by the quiescence solution, we say that $\mathscr{A}$ assigns the level $\ell_U^{\mathscr{A}} = \ell_U(v) + ||\mathscr{A}(v)||$ to the arrow v). In particular, it ensures that, for the computation of the box of an !-link $e_!$, we can take the solution of any state $\mathscr{S}$ s.t. $||\mathscr{S}(e_!)|| \geq 0$.

Lemma 54 *Let $e_! \in \mathsf{lnk}_!(U)$ and U be a proper $u\ell$-structure. If $\mathscr{S}$ is an internal state of U s.t. $||\mathscr{S}(e_!)|| \geq 0$ and $\mathscr{A}$ is the $\mathscr{S}$-solution of U, then $v_0 \in \mathsf{vtx}(\mathsf{box}_U^\ell(e_!)) \setminus \mathsf{doors}(\mathsf{box}_U^\ell(e_!))$ iff v_0 is connected to $v_k = \mathsf{idoor}(e_!)$ by a path $\phi = v_0 e_1 v_1 \ldots v_{k-1} e_k v_k$ s.t. $\ell_U^{\mathscr{A}}(v_i) \geq \ell_U^{\mathscr{A}}(\mathsf{idoor}(e_!))$, for $i = 0, \ldots, k$.*

Proof. The statement of the lemma is invariant under π-reduction (see the proof of Lemma 43); therefore, it suffices to prove it for any ℓ-structure N. The if part of the proof of Proposition 51 gives a way to build a generic solution of N; by inspection of that construction, we conclude. $\square$

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

Lemma 55 *Let $\mathscr{A}$ be the $\mathscr{S}$-solution of a proper $u\ell$-structure U. For every $B_0 = \mathsf{box}_U^\ell(e_0)$, we have that*

$$\mathscr{A}(v) = \begin{cases} \mathscr{A}_0(v)\,\mathscr{S}(e_0)^{\uparrow q_0} & \textit{when } v \in \mathsf{vtx}(B_0) \setminus \mathsf{doors}(B_0) \\ \mathscr{A}_0(v) & \textit{otherwise} \end{cases}$$

where $q_0 = ||\mathscr{A}_0(\mathsf{edoor}(e_0))||$ and $\mathscr{A}_0$ is the solution of U for the internal state

$$\mathscr{S}_0(e_!) = \begin{cases} 1 & \textit{when } e_! = e_0 \\ \mathscr{S}(e_!) & \textit{otherwise} \end{cases}.$$

Proof. Let $\mathscr{A}_0$ be the $\mathscr{S}_0$-solution of U. First of all, we prove that the $\mathscr{A}$ constructed in the statement is an $\mathscr{S}$-assignment. By construction and by the hypothesis on $\mathscr{A}_0$, the lifting assignment constraints hold for each link in $U \setminus B_0$. Let $\mathscr{S}(e_0) = \mathcal{L}[m_1, q_1, q_1] \cdots \mathcal{L}[m_k, q_k, q_k]$. For every $v \in \mathsf{vtx}(B_0) \setminus \mathsf{doors}(B_0)$, we have that $\ell_U(v) + ||\mathscr{A}_0(v)|| \geq \ell_U(e_0) + 1 + q_0$ (by Lemma 54). Therefore, if $Q_i = q_1 + \cdots + q_{i-1}$, we have $m_i + q_0 < \ell_U(e_0) + 1 + q_0 + Q_i \leq \ell_U(v) + ||\mathscr{A}_0(v)|| + Q_i$ (by $\mathscr{S}(e_0) \in \mathsf{LSeq}[\ell_U(e_0), \ell_U(e_0) + 1]$); that is, $\mathscr{A}(v) = \mathscr{A}_0(v)\,\mathscr{S}(e_0)^{\uparrow q_0} \in \mathsf{LSeq}[0, \ell_U(v)]$. Therefore, we left to verify that the lifting assignment constraints hold for every $e \in \mathsf{lnk}(B_0)$. The only relevant case is $e \in \mathsf{adoors}(B_0)$. Let $v_i = \mathsf{idoor}(e)$ and $v_e = \mathsf{edoor}(e)$. By Lemma 54, $\ell_U(v_e) < \ell_U(e_0) + 1 + Q$, where $Q = q_0 - ||\mathscr{A}_0(v_e)||$ (since $v_e \in \mathsf{doors}(B_0)$). Therefore, $\mathscr{A}(v_i) = \mathscr{A}_0(v_i)\,\mathscr{S}(e_0)^{\uparrow q_0} = \mathcal{H}\,\mathscr{A}_0(v_e)\,\mathscr{S}(e_0)^{\uparrow q_0} = \mathcal{H}\,\mathscr{S}(e_0)^{\uparrow Q}\,\mathscr{A}_0(v_e)$. Now, since $\ell_U(e_0) + 1 + q_0 \leq \ell_U(v_i) + ||\mathscr{A}_0(v_i)|| = \ell_U(v_i) + ||\mathcal{H}|| + ||\mathscr{A}_0(v_e)||$, we have $\ell_U(v_e) \leq \ell_U(e_0) + Q \leq m_i + Q < \ell_U(e_0) + 1 + Q + Q_i \leq \ell_U(v_i) + ||\mathcal{H}|| + Q_i$; that is, $\mathcal{H}\,\mathscr{S}(e_0)^{\uparrow Q} \in \mathsf{LSeq}[\ell_U(v_e), \ell_U(v_i)]$. Therefore, $\mathscr{A}$ is an $\mathscr{S}$-assignment for U. Moreover, it is readily seen that, if $\mathscr{A}$ is not an $\mathscr{S}$-solution of U, then $\mathscr{A}_0$ is not an $\mathscr{S}_0$-solution of U. $\square$

10 Sharing graph machine

The $s\ell$-structures, equipped with the π-rules and the sharing β-rule, give the abstract machine for the implementation of the 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 that we want to implement in our sharing machine 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 rewriting rules, or β-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 $k \geq 0$?-links $e^1_?, e^2_?, \ldots, e^k_?$. If $\ell(\mathsf{idoor}(e^i_?)) > \ell(\mathsf{edoor}(e^i_?))$, the ?-link $e^i_?$ is auxiliary door of some box B_2; whereas, when $\ell(\mathsf{idoor}(e^i_?)) = \ell(\mathsf{edoor}(e^i_?))$, the ?-link $e^i_?$ does not close any box. Namely, each $e^i_?$ closes $\ell(\mathsf{idoor}(e^i_?)) - \ell(\mathsf{edoor}(e^i_?))$ boxes.

(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^i_1 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.*, to the copy of $\mathsf{pdoor}(e_!)$ in B_1^i);
(d) each $\curlyvee$-link $e_\curlyvee$ that is below a door v of B_1 is transformed into a $\curlyvee$-link $\bar{e}_\curlyvee$ s.t. $\mathsf{pdoor}(\bar{e}_\curlyvee) = \mathsf{pdoor}(e_\curlyvee)$ and $\mathsf{adoors}(\bar{e}_\curlyvee) = (\mathsf{adoors}(e_\curlyvee) \setminus \mathsf{adoors}(B_1)) \cup \{v^i \mid v \in \mathsf{adoors}(e_\curlyvee) \cap \mathsf{adoors}(B_1)\}$ (where v^i denotes the copy of v in B_1^i).

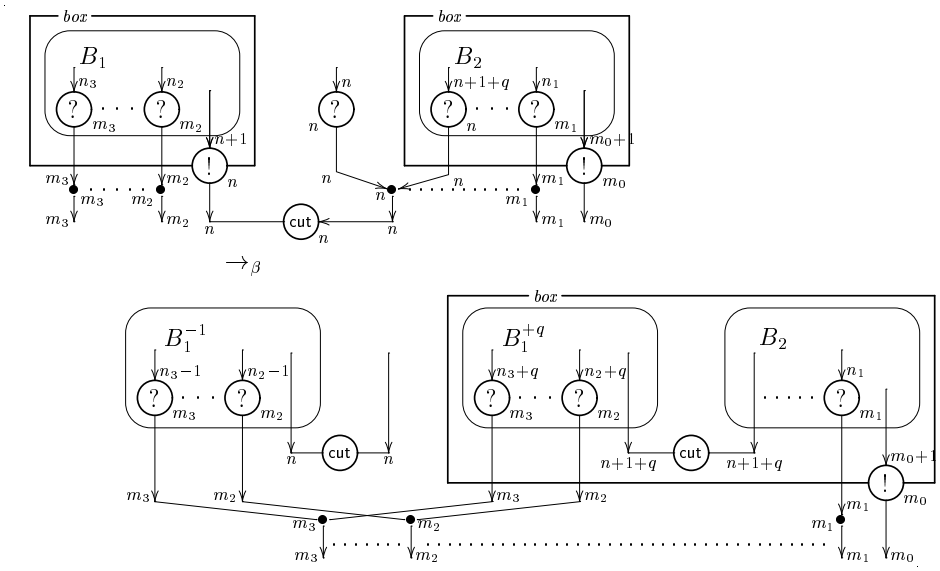

Fig. 22. Interacting boxes: global β-rule.

Remark 56 The extent of the previous pattern is not precisely defined. For instance, in order to faithfully reproduce the schema in Figure 22 in the λ-calculus structures, we should split the usual β-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 in Figure 22. On the other hand, in some weak sense and apart for some minor technicalities, the leveled reformulation of the β-rule in Figure 1 fits into the pattern of Figure 22 and all the theorems that we are going to give are valid for the sharing version of that rule also. Besides, in order to not be worthlessly abstract, in the proofs of the forthcoming theorems, we shall always refer to the schema in Figure 22.

Remark 57 In terms of levels, pushing B_1^i inside the boxes of $e_?^i$ means to increase the levels in B_1^i by the offset $q_i = \ell(\mathsf{idoor}(e_?^i)) - \ell(\mathsf{edoor}(e_?^i)) - 1$ (denoted by the superscript $+q_i$ in Figure 22). In fact, by the restriction on the levels of a structure, this corresponds to enclosing 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); whereas, when $e_?^i$ does not close any box, it corresponds to removing the box around B_1^i (see B_1^{-1} in Figure 22).

10.2 Sharing β-rule

We aim at showing that, under the only assumption that the ℓ-nets of the calculus have the box nesting property, the global version of the β-rule in Figure 22 can be replaced by the sharing or β_s-rule in Figure 23.

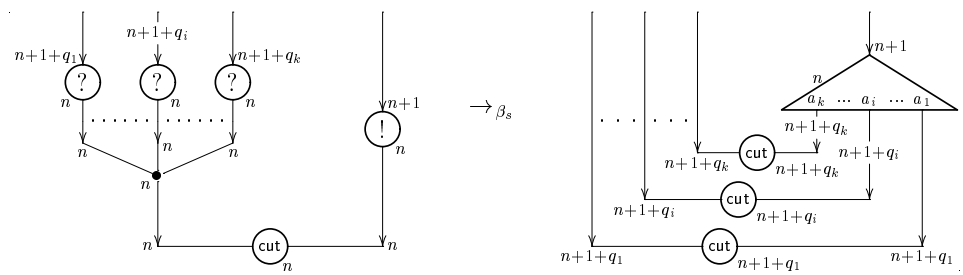

Fig. 23. Interacting boxes: (sharing) β_s-rule.

In more detail, we want to prove that the result of the application (without any restriction) of a sequence of $(\beta_s \cup \pi)$-rules to an ℓ-net N leads to a proper $s\ell$-structure G whose read-back $\mathscr{R}(G)$ is a β-reduct of N.

We stress that the box nesting property is our key assumption. Therefore, as a preliminary fact, let us notice 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 preliminary step towards the achievement of our goal is the proof of a simulation property for the β_s-rule similar to the one already proved for the π-rules (Lemma 44). Unfortunately, in the case of the β_s-rule the proof technique is not so direct as in the case of the π-rules, since a β_s-reduct of a $u\ell$-structure is a $u\ell$-structure only if the contracted redex is unary (*i.e.*, only if the β-redex involves only one ?-link). Luckily, there is 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 $r = (e_!, \{e_?^1, \ldots, e_?^k\})$ in two steps (assume to represent a β-redex by the pair formed of the box door links involved in it):

(i) the box B_1 of the !-link $e_!$ is duplicated, creating an instance B_1^i for each $e_?^i$ (see Figure 24);
(ii) a β-rule is executed for each of the k redexes $(e_!, \{e_!^i\})$.

The first step of this decomposition defines a *box duplication* rule transforming a $k$ary β-redex into k unary redexes.

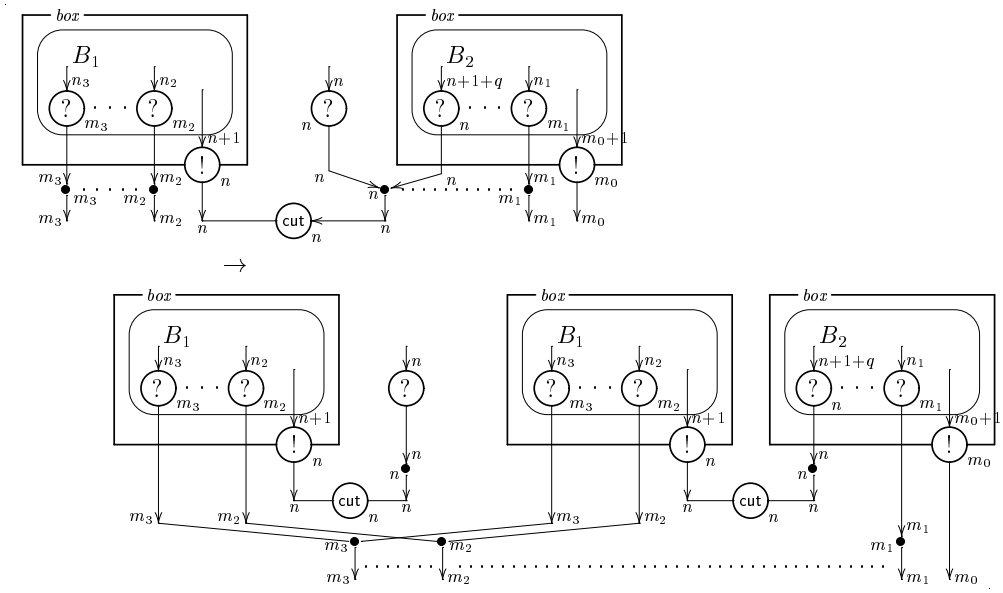

Fig. 24. Interacting boxes: box duplication.

Exploiting the previous decomposition, we define an unshared version of the β-rule, the β_u-rule, by replacing β_s for β in the second step. Moreover, while the β-rule applies to ℓ-structures, the β_u-rule applies to proper $u\ell$-structures (we remind that in section 9 we have given the definition of the boxes of a proper $u\ell$-structure). The soundness of this approach is ensured by the fact that the β_u-reduct of a proper $u\ell$-structure U is a proper $u\ell$-structure whose 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 ith instance of v; (ii) $\mathsf{pdoor}(e_\Upsilon) \mapsto \mathsf{pdoor}(e_\Upsilon)^i$, if e_Υ is the contraction link of the redex; (iii) $v \mapsto v$, otherwise. Analogously, we define the relation $\mapsto \subset \mathsf{lnk}(U) \times \mathsf{lnk}(U'')$.
Let $r : U \to_{\beta_u} U'$, with $r = (e_!, \{e_?^1, \ldots, e_?^k\})$. For any internal state $\mathscr{S}$ of U'' and $i = 1, \ldots, k$, let us take $\mathscr{S}^i(e) = \mathscr{S}(e^i)$, when $e \in \mathsf{lnk}(B_1)$ and $e \mapsto e^i$, and $\mathscr{S}^i(e) = \mathscr{S}(e)$, otherwise. Let $\mathscr{A}^i$ be the $\mathscr{S}^i$-solution of U. By Lemma 55, all the $\mathscr{A}^i$ coincide outside B_1; therefore, let $\mathscr{A}^1(v) = \mathscr{A}^2(v) = \cdots \stackrel{def}{=} \mathscr{A}(v)$, for $v \mapsto v$. The map $\mathscr{A}(v)$ extends to the whole $\mathsf{vtx}(U'')$ by $\mathscr{A}(v) = \mathscr{A}^i(v^i)$, when $v \in \mathsf{vtx}(B_1)$ and $v \mapsto v^i$; moreover, $\mathscr{A}$ 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 port name of the lift inserted by the ith β-redex in the β_u-rule. 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) if $\mathscr{A}_0$ is the $\mathscr{S}_0$-solution of U (actually, $\mathscr{S}_0$ is an internal state of U''; nevertheless, by $\mathscr{S}_0$ we also denote the internal state of U obtained by replacing $\mathscr{S}_0(e_!) = 1$ for $\mathscr{S}_0(e^i_!) = 1$) and $\mathcal{H}^i$ is the unique lifting sequence s.t. $\mathcal{H}^i\,\mathscr{A}_0(\mathsf{edoor}(e^i_?)) = \mathscr{A}_0(\mathsf{idoor}(e^i_?))$, then $\mathscr{S}(e^i_!) = \mathcal{L}[\ell(e_!), q_i, a_i]\,\mathcal{H}^i$. By Lemma 55 and by the construction of the $\mathscr{S}$-solution $\mathscr{A}$ of U'' in the previous paragraph, we have that: (i) $\mathscr{A}(v) = \mathscr{A}_0(v)$, when $v \notin \mathsf{vtx}(B_1)$; (ii) $\mathscr{A}(v) = \mathscr{A}_0(v^i)$, when $v \in \mathsf{doors}(B_1)$ and $v \mapsto v^i$; (iii) $\mathscr{A}(v^i) = \mathscr{A}_0(v)\,\mathscr{S}(e^i_!)^{\uparrow r}$, when $v \in \mathsf{vtx}(B_1) \setminus \mathsf{doors}(B_1)$ and $v \mapsto v^i$, with $r = ||\mathscr{A}_0(\mathsf{pdoor}(e_!))||$.
Since the $u\ell$-structure U' is obtained from U'' by the simultaneous sharing reduction of a set of unary β-redexes, each arrow of U'' has at most one residual in U'; namely, the arrows of U' are a subset of the arrows of U''. Let $\mathscr{S}'$ be the restriction of the internal state $\mathscr{S}$ of U'' to $\mathsf{lnk}_!(U'') \cap \mathsf{lnk}_!(U')$; the restriction of $\mathscr{A}$ to $\mathsf{vtx}(U'') \cap \mathsf{vtx}(U')$ gives the $\mathscr{S}'$-solution $\mathscr{A}'$ of U'. Since $\mathscr{S}'$ is any internal state $\mathscr{S}'$ of U', we conclude that U' is proper.
The construction of $\mathscr{A}'$ shows that (take $\mathscr{S}'$ equal to the quiescence internal state) $\ell^{\mathscr{I}}_U(v) = \ell^{\mathscr{I}}_{U'}(v)$, when $v \notin \mathsf{vtx}(B_1)$, and $\ell^{\mathscr{I}}_U(v) + q_i = \ell^{\mathscr{I}}_{U'}(v^i)$ when $v \in \mathsf{vtx}(B_1) \setminus \mathsf{doors}(B_1)$ 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 $U_0 \prec\!\!\!\!\prec G_0$. For any $G_0 \to_{\beta_s} G_1$, there exists $U_0 \to^+_{\beta_u} U_1$ s.t. $U_1 \prec\!\!\!\!\prec G_1$.*

Proof. Let $r : G_0 \to_{\beta_s} G_1$. The corresponding reduction $\rho : U_0 \to^+_{\beta_u} U_1$ that we are looking for is a development of the set of redexes $R = M_0^{-1}(r)$, where $M_0 : U_0 \prec\!\!\!\!\prec G_0$; whereas, the complete unfolding $M_1 : U_1 \prec\!\!\!\!\prec G_1$ is the natural one induced by the residual relation.
Before to enter into the detail 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 the β_s-rule, $\mapsto$ is an injective partial function.
Let $r : U_0 \to U'$ be a box duplication. For $v' \in \mathsf{vtx}(U')$, 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, these are the only internal states for which we can simultaneously have $M'(v^1) = M'(v^2)$ and $\mathscr{A}(v^1) = \mathscr{A}(v^2)$, for some $v^1 \neq v^2$ 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.: (i) U' does not contain any residual of $r \in R$; (ii) 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, we can 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. Let $\rho'' : U' \to_{\beta_s} U_1$ be obtained reducing all the (unary) redexes in the set $R' = \{(e'_!, \{e'_?\}) \mid M'(e'_!) = e_!, M'(e'_?) = e^i_?\}$ and, for $i = 1, 2, \ldots$, inserting a lift with port name a_i for each redex $(e'_!, \{e'_?\})$ s.t. $M'(e'_?) = e^i_?$. 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 $M_1 : U_1 \prec\!\!\prec G_1$.
To conclude, let us note that $U_0 \xrightarrow{\rho'} U' \xrightarrow{\rho''} U_1$ can be rearranged in a sequence of β_u-reductions. $\square$

11 Sharing reductions of nets

Finally, we have all the ingredients for stating 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-rules are sound w.r.t. read-back.

Summing up, the rules β_s and π define an abstract computational system, say a *sharing graph machine*, in which the implementation of the global β-rule is decomposed in a set of atomic steps. In order to formalize this latter claim, 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 it is a $\Sigma s\ell$*-net*, when its read-back is a $\Sigma\ell$-net. Further, given a set Nets of correct ℓ-nets, we shall 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 $G \in \mathsf{SNets}$ iff $\mathscr{R}(G) \in \mathsf{Nets}$.

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

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

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

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

Proof.

(i) Since it is trivial to see that $G \to_\gamma G'$ implies $\mathscr{R}(G) \to_\gamma \mathscr{R}(G')$, 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-rule is followed by a normalizing π-reduction. By Proposition 50 and Proposition 60, we see that $r : N \to_\beta N''$ implies $\rho : N \xrightarrow{r}_{\beta_s} G \to^*_\pi \mathscr{R}(G) = N''$, where ρ is standard. Thus, for any $N \to^*_\sigma N'$ there is a standard strategy reduction s.t. $N \to^*_{\sigma_s} N'$.
(iii) Since the π-rules are strongly normalizing (Proposition 49), a σ_s-reduction ρ_s contains an infinite number of π-rules iff it contains an infinite number of rules β_s or γ also. 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). $\square$

The 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 (*i.e.*, we cannot say anything on how to optimize the number of γ-rules).

Let us assume the hypotheses and notations of Theorem 62. A σ_s-reduction $\rho_s : G \to_{\sigma_s} G'$ 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 $\rho'_s : G \to_{\sigma_s} G'$ (*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 $(e_\triangledown, e)$ is formed of a mux $e_\triangledown$ 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 *Every σ_o-reduction is β-optimal.*

Proof. Let $\rho_o : G \to^*_{\sigma_o} G'$. 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 $\rho : G \to^*_{\sigma_s} G'$. Moreover, if $G \to^*_{\sigma_o} G_1$, any redex r of G_1 has at most a residual in any G_2 s.t. $G_1 \to^*_{\sigma_o} G_2$. $\square$

The β-optimal reductions correspond to the usual λ-calculus or linear logic optimal reductions ([Lam90,GAL92b,GAL92a,AG98]) when σ_s is the corresponding sharing implementation (*cf.* section 4.1). In these cases, to delay or to not execute a β-duplicating rule does not hide any β-redex. In fact, for every β-redex r of $\mathscr{R}(G)$, there is a reduction $G \to^*_{\pi_o} G'$ 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 every λ-calculus or MELL $s\ell$-net G, if N is the σ-normal form of $\mathscr{R}(G)$, then the σ_o-normal form of G is an $s\ell$-net G' s.t. $\mathscr{R}(G') = N$. 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 $\rho'_i\rho''_j : N \to_\gamma N_{ij}$, with $i, j \in \{1, 2\}$; in which, ρ'_i is a reduction isomorphic to ρ_i and internal to B', whereas ρ''_j is a reduction isomorphic to ρ_j and internal to B''. For the sake of simplicity, let us assume that each N_{ij} is in normal form also. If G is a proper $s\ell$-structure in which B' and B'' are shared, there is no β-optimal reduction $G \to_{\sigma_o} G_{12}$ s.t. $\mathscr{R}(G_{12}) = N_{12}$, and analogously for N_{21}.

The latter point shows that the π-rules are not a nice way to present Lévy's optimality only. They give indeed a low-level implementation interesting per se, that can be applied also when 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 implies the impossibility of erasing boxes. The latter point is instead false: 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. An ϵ-link erases box door links also. Therefore, the introduction of an erasing link implies a slight reformulation of the definition of sharing implementation. Besides, this does not hurt Theorem 62, the only consequence is a more involved statement.)

For instance, let us assume 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 $n \leq m$ 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 $\mathsf{doors}(e_\epsilon) \in \mathsf{adoors}(e_\triangledown)$.) Because of previous restrictions, 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 $\ell_G(e_\epsilon) = \ell_G(e_?) + 1$. It is readily seen that the β_s-reduction of r enables 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 $\mathsf{box}^\ell_G(e_!)$, s.t. each one of them is 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 remain alive. The only way for erasing that garbage is loosing 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 that is our main concern in this paper.

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 these 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 [Laf97] 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 constraints that we have imposed in our analysis is that in a β-redex $(e_!, \{e_?^1, \ldots, e_?^k\})$ no $e_?^i$ is an auxiliary door link of $\mathsf{box}^\ell(e_!)$, that is, we have forbidden boxes interacting with themselves. All the results of the paper should hold dropping this constraint. Nevertheless, we chose to eliminate these cyclic situations, for their presence would have involved the description of the β-rules. Furthermore, the analog of a box interacting with itself is the μ-rule $\mu x.T \to_\mu T[\mu x.T/x]$; that is, a calculus with an explicit recursion operator. Hence, we have preferred to maintain the distinction between β and μ-rule in sharing graphs also, reserving a separate study for the μ-like rules.

We have already stated that β_s plus π give an abstract machine by which to get sharing implementations. Such a sharing graph machine seems the natural low-level model of λ-calculus. We hope that this might finally lead to a satisfactory measure for the cost of λ-calculus reductions, and to a way of comparing 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], Lawall and Mairson [LM96], and Asperti and Mairson [AM98]

showing that Lévy families cannot be the cost model of λ-calculus reductions.

Acknowledgments

I wish to 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

[AG98] Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages.* Cambridge University Press, 1998.

[AGN96] A. Asperti, C. Giovannetti, and A. Naletto. The Bologna Optimal Higher-Order Machine. *Journal of Functional Programming*, 6(6), 1996.

[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.

[AL95] Andrea Asperti and Cosimo Laneve. Comparing λ-calculus translations in sharing graphs. In M. Dezani-Ciancaglini and Gordon Plotkin, editors, *Typed Lambda Calculi and Applications, TLCA '95*, volume 902 of *LNCS*, pages 1–15. Springer-Verlag, 1995.

[AM98] Andrea Asperti and Harry Mairson. Parallel beta reduction is not elementary recursive. In *Conference Record of POPL '98: The 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, San Diego, California, 1998.

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

[Asp96] Andrea Asperti. On the complexity of beta-reduction. In *Conference Record of POPL '96: The* 23rd *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, St. Petersburg Beach, Florida, 1996.

[DR93] Vincent Danos and Laurent Regnier. Local and asyncrhonous beta-reduction. In *Proceedings of the 8th Annual IEEE Symposium on Logic in Computer Science*, (LICS'93), pages 296–306, Montreal, Canada, 1993.

[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 LNCS, pages 319–333. Springer-Verlag, 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* (POPL'92), Albuquerque, New Mexico, 1992, pages 15–26, 1992.

[GAL92b] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science* (LICS'92), pages 223–234, Santa Cruz, California, 1992.

[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, 1989. Elsevier (North-Holland).

[GMM97a] Stefano Guerrini, Simone Martini, and Andrea Masini. Coherence for sharing proof nets. IRCS Report 97-03, Institute for Reasearch in Cognitive Science, University of Pennsylvania, March 1997. Extended abstract in: Harald Ganzinger, editor, *Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA-96)*, volume 1103 in *LNCS*, pages 215–229, New Brunswick, NJ, 1996. Springer-Verlag.

[GMM97b] Stefano Guerrini, Simone Martini, and Andrea Masini. Proof nets, garbage, and computations. In *TLCA 97*, volume 1210 in LNCS, 1997. Springer-Verlag.

[Gue97] Stefano Guerrini. Sharing-graphs, sharing-morphisms and (optimal) λ-graph reductions. In J. Ginzburg, Z. Khasidashvili, C. Vogel, J.-J. Lévy, and E. Vallduví, editors, *The Tbilisi Symposyum on Logic, Language and Computation*, Gudauri, Georgia, October 1995. In *Studies in Logic, Language and Information*. CSLI Publications, 1997.

[Klo80] J. W. Klop. *Combinatory Reduction Systems*. PhD Thesis, Matematisch Centrum, Amsterdam, The Netherlands, 1980. Mathematical Centre Tracts 127.

[Laf90] Yves G. A. Lafont. Interaction nets. In *Conference Record of the 17th Annual ACM Symposium on Principles of Programming Languages*, (POPL'90), pages 95–108, San Francisco, California, 1990.

[Laf97] Yves Lafont. Interaction Combinators. *Information and Computation*, 137(1):69–101, 1997.

[Lam90] John Lamping. An algorithm for optimal lambda calculus reduction. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages*, (POPL'90), pages 16–30, San Francisco, California, 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, 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, 1993.

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

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

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