Under consideration for publication in Math. Struct. in Comp. Science

Natural Deduction via Graphs: Formal Definition and Computation Rules

HERMAN GEUVERS and IRIS LOEB

Institute for Computing and Information Science,
Radboud University Nijmegen,
The Netherlands
{H.Geuvers, I.Loeb}@cs.ru.nl

Received 13 March 2006

We introduce the formalism of *deduction graphs* as a generalization of both Gentzen-Prawitz style natural deduction and Fitch style flag deduction. The advantage of this formalism is that subproofs can be shared, like in flag deductions (and unlike natural deduction), but also that the linearisation used in flag deductions is avoided. Our deduction graphs have both nodes and *boxes*, which are collections of nodes that also form a node themselves. This is reminiscent of the bigraphs of Milner, where the *link graph* describes the nodes and edges and the *place graph* describes the nesting of nodes. In the paper we give a precise definition of deduction graphs and we give examples to illustrate them. Furthermore we analyse their computational behaviour by studying the process of cut-elimination and by defining translations from deduction graphs to simply typed lambda terms. From a slight variation of this translation we conclude that the process of cut-elimination is strongly normalising. The translation to simple type theory removes quite a lot of structure and we therefore also propose a translation to a context calculus with lets, that faithfully captures the structure of deduction graphs. The proof nets of linear logic also present a graph-like presentation of natural deduction. We point out some similarities of the two formalisms.

Key words: natural deduction, cut-elimination, lambda calculus with let-binding, typed lambda calculus.

1. Introduction

Gentzen-Prawitz (Gentzen 1969; Prawitz 1965) style deduction and Fitch (Fitch 1952) style deduction are two popular ways of doing "natural deduction". In Gentzen-Prawitz style, a deduction has the shape of a tree. This makes the meaning of the logical connectives very perspicuous, but from a practical point of view it is not very efficient, because in order to use one (proved) result in two branches of the tree, one has to prove it twice. In Fitch style, a deduction has a linear format, which makes it possible to reuse an already proved result, by just referring to the line (it has a number) on which it has been proved. However, the order of steps in a Fitch deduction is quite arbitrary: many logical

inferences are independent of another and can henceforth be put in any order. So, there is a lot of bureaucratic detail in Fitch deductions that is irrelevant for the underlying *logical* structure, as a matter of fact, the linearity blurs the logical structure. This logical structure is presented by the dependencies between the logical formulas: how a formula is inferred from other formulas. In this paper we define *deduction graphs*, which are graphs of a specific shape that aim at capturing purely the logical structure of a deduction and omitting bureaucratic details.

In deduction graphs, the nodes have formulas as labels and if node n with label A (this will be denoted as (n, A)) has edges to the nodes $(m_1, B_1), \ldots, (m_k, B_k)$, the idea is that A is derivable from $B_1, \ldots, B_k$ in one logical step. This could also have been formalised with a *multi-edge* from (n, A) to $(m_1, B_1), \ldots, (m_k, B_k)$, but we wanted to keep it simple. An *free node* will be a node with no outgoing edges, which will be a (local) hypothesis. Obviously, not all graphs are deductions graphs: we have to put restrictions. First of all we should avoid cyclicity, because we want to have a notion of *derivability*: a formula A that occurs as a (label of a) node of deduction graph G is meant to be *derivable* from the total collection of *free nodes* of G that A points hereditarily to. Of course, we want this notion of derivability to correspond to the well-known one from natural deduction and we want more: the deduction graphs should also in their fine structure correspond closely to well-known Gentzen-Prawitz style deduction and Fitch style deduction. We have achieved this by making sure that these well-known natural deductions are (almost immediately) an instance of our deduction graphs.

An important aspect of natural deduction is the *scope* of a hypothesis. In Gentzen-Prawitz style we have the concept of *discharging* of hypotheses, which is done e.g. in the $\rightarrow$ introduction rule. The discharged hypotheses are no longer hypotheses on which the conclusion depends. In Fitch style, the notion of scope is even more explicitly present in the *flags* that delimit the scope of a (local) hypothesis: e.g. when the $\rightarrow$-introduction rule is applied, the scope ends. In deduction graphs, we formalise scope using *boxes*. A box is a subgraph (i.e. a collection of nodes and the edges between them) that again forms a node itself. These boxes are typically used for the $\rightarrow$-introduction rule: the hypotheses A (to be "discharged") are put within the box (so they are "hidden" for the rest of the graph) . To make all this a bit more intuitive we give an example of a deduction graph, which can be seen as a derivation of $(B{\rightarrow}C){\rightarrow}(A{\rightarrow}C)$ from $A{\rightarrow}B$.

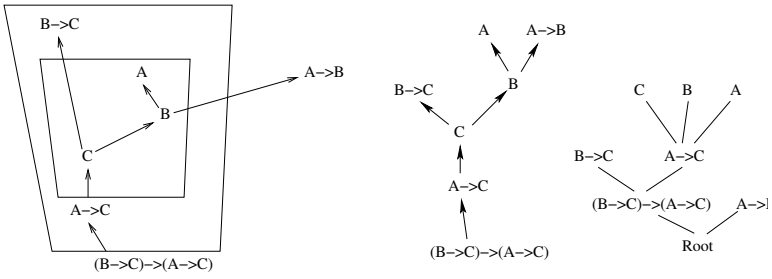

Fig. 1. Deduction graph of $(B{\rightarrow}C){\rightarrow}(A{\rightarrow}C)$ from $A{\rightarrow}B$ and link and place graphs

This is slightly reminiscent of the bigraphs of Milner (Milner 2004). There also collections of nodes can form nodes themselves, giving rise to the view of a bigraph as a combination of a *link graph* and a *place graph*. The first describes the nodes and edges between them and the second describes the nesting structure. This terminology can be applied to our deduction graphs as well. The graph in Figure 1 then gives rise to the two graphs to the right of it. Notwithstanding this superficial correspondence, our deduction graphs have a quite different nature from Milner's bigraphs: in bigraphs edges have no direction and can be multi-linked, but most importantly, nodes in bigraphs have a fixed *arity* which determines the number of edges it connects to. In our setting the number of ingoing edges is unlimited.

Gentzen-Prawitz style deductions have the advantage that they can be brought in a specific form: by repeatedly applying cut-elimination, we eventually get a deduction "without detours". This has led to the important result that Gentzen-Prawitz style deductions have the subformula property (Prawitz 1971). It is well-known, that Gentzen-Prawitz style deductions correspond to simply typed λ-terms and that cut-eliminations can be seen as β-reduction. In (Geuvers and Nederpelt 2004), this has been generalised to Fitch deduction: a mapping from Fitch deductions to typed λ-terms was defined and it was observed that the correspondence is not 1-1. As a matter of fact, Fitch deductions correspond closer to typed terms in a λ-calculus with lets (but that was not made precise in (Geuvers and Nederpelt 2004)).

In the present paper, besides introducing and motivating the notion of deduction graphs (Section 2), the main topic is the study of the computational content of deduction graphs, notably given by the process of cut-elimination (Section 3). For deduction graphs, the basic step of cut-elimination is a reordering of edges. The real 'work' is in the steps that make a cut explicit, where e.g. an unsharing of a part of the deduction graph is required. Cut-elimination is precisely defined and analysed by studying translations to other known calculi.

We first define a mapping $[\![-]\!]$ to simply typed λ-calculus, which takes a deduction graph and a node and computes the simply typed term corresponding to the proof of that node. This follows the well-known Curry-Howard formulas-as-types embedding and has the property that, if $G \twoheadrightarrow_{\mathrm{cut}} G'$ and the cut-formula can be reached from node n, then $[\![G, n]\!] \twoheadrightarrow_{\beta}^{+} [\![G', n]\!]$ (Section 4.1). As this mapping may remove parts of the deduction graph, we can't conclude termination of cut-elimination from this. We therefore define a map $\langle\!|G, n|\!\rangle$ to a slightly modified version of simple type theory in which all subproofs are preserved. From this we obtain strong normalisation of cut-elimination (Section 4.2).

To give a better analysis of deduction graphs, we define a mapping $\langle\!\langle - \rangle\!\rangle$ to a typed context calculus with lets (Section 5). This map takes a deduction graph and gives a context (i.e. a term with a hole) in the context calculus with lets. The substitution of a (name of a) node in a context gives the interpretation of the deduction 'in that point'. Moreover, if $G \twoheadrightarrow_{\mathrm{cut}} G'$ then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{\mathrm{let}}^{+} \langle\!\langle G' \rangle\!\rangle$, where the reduction steps code more directly the changes in the deduction graph.

Proof nets (Girard 1987) are another way to give a presentation of deductions (in linear logic) as graphs. As they originate from a sequent calculus, they are constructed quite differently than our deduction graphs. Nevertheless, the reduction rules look very similar.

We will discuss some future work to study the connection between these formalisms (Section 6.1). In Section 6.2 some other ideas for future research are suggested.

2. Deduction Graphs

Definition 2.1. A *closed box directed graph* is a triple $\langle X, G, (\mathcal{B}_i)_{i\in I}\rangle$ where X is a set of *labels*, G is a directed graph where all nodes have a label in X and $(\mathcal{B}_i)_{i\in I}$ is a collection of sets of nodes of G, the *boxes*, such that these boxes themselves (the $\mathcal{B}_i$) are again nodes. Moreover, the boxes $(\mathcal{B}_i)_{i\in I}$ should satisfy the following properties.

1 (Non-overlap) Two boxes are disjoint or one is contained in the other: $\forall i, j \in I(\mathcal{B}_i \cap \mathcal{B}_j = \emptyset \vee \mathcal{B}_i \subset \mathcal{B}_j \vee \mathcal{B}_j \subset \mathcal{B}_i)$,
2 (Box-node edge) There is only one outgoing edge from a box-node and that points into the box itself,
3 (No edges into a box) Apart from the edge from the box-node, there are no edges pointing into a box.

Formally, the labeled directed graph G is a triple $\langle N, E, L\rangle$, where N is a set of nodes, E is a set of edges and L is a (labelling) function from N to X. Furthermore, there is an injection from the set of boxes to the set of nodes, allowing us to identify a box with its *box node*, so we freely write $m \in n$ to denote that m is the box node of some box $\mathcal{B}$ and $n \in \mathcal{B}$. Following Milner's idea, we define the link graph and the place graph of a closed box directed graph.

Definition 2.2. The *link graph* of a closed box directed graph $\langle X, G, B\rangle$ is the labeled directed graph $\langle X, G\rangle$.

Definition 2.3. The *place graph* of a closed box directed graph $\langle X, \langle N, E, L\rangle, B\rangle$ is the labeled graph $\langle X, \langle N, \in, L\rangle\rangle$, where $\in$ is the membership relation between nodes and boxes.

Definition 2.4. A closed box directed graph $\langle X', \langle N', E', L'\rangle, B'\rangle$ is a *subgraph* of a closed box directed graph $\langle X, \langle N, E, L\rangle, B\rangle$, when:

1 $X' \subseteq X$;
2 $\langle N', E'\rangle$ is a subgraph of $\langle N, E\rangle$;
3 $L' = L_{|N'}$;
4 $\langle N', \in\rangle$ is the subgraph of $\langle N, \in\rangle$ induced by N'.

We denote a closed box directed graph by $\langle G, (\mathcal{B}_i)_{i\in I}\rangle$, or by G, leaving the set of labels X, and possibly also the set of boxes, implicit.

So, if $\mathcal{B}$ is a box, $m \in \mathcal{B}$, $n \notin \mathcal{B}$ and $n \longrightarrow\!\triangleright\, m$ then n is the box-node of $\mathcal{B}$. Moreover, if $\mathcal{B}$ is a box with box-node n, then there is exactly one edge from n to some $m \in \mathcal{B}$.

In Figure 2 we give three non-examples of closed box directed graphs and one example of a closed box directed graph.

Definition 2.5. Let G be a closed box directed graph. A *box-topological ordering* of G,

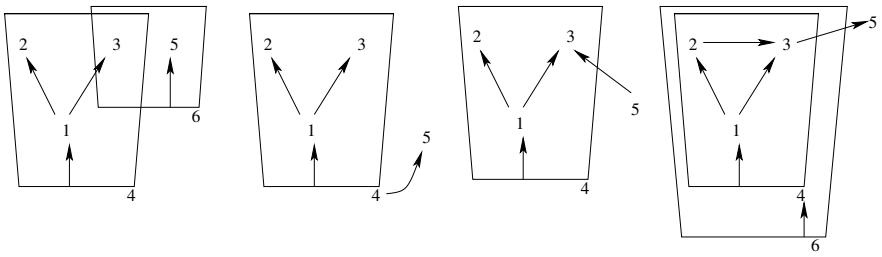

Fig. 2. Three non-examples and one example of a closed box directed graph

is a linear ordering of the nodes of G, that both respects the partial order imposed by the link graph of G and the partial order as imposed by its place graph.

When talking about graphs, we sometimes explicitly write the label with the node, e.g. we write $(n, A) \longrightarrow\!\triangleright (m, B)$ to denote the edge from n with label A to m with label B. We also sometimes will identify a node with its label, if no confusion arises. (Note however that different nodes may have the same label.) So if $n_0 \longrightarrow\!\triangleright n_1$ and the labels on these nodes are A and B, respectively, we sometimes denote this edge by $A \longrightarrow\!\triangleright B$.

We use arrows for three different purposes and for clarity we have used different arrows for each purpose:

— In the text we denote an edge by $\longrightarrow\!\triangleright$,
— We denote the implication by $\rightarrow$,
— We denote a reduction by $\Longrightarrow$.

In a directed graph with boxes, we want to define which nodes are "visible" and can henceforth be used to derive new (logical) conclusions. We therefore define the notion of *scope*, which is not related to the graph structure, but to the boxes.

Definition 2.6. Let $\langle G, (\mathcal{B}_i)_{i\in I}\rangle$, be a directed graph with boxes and let n_0 and n_1 be nodes in this graph.

— Node n_1 is *in scope of* n_0 if n_0 is in all boxes that n_1 is in. In a formula: $\forall i \in I(n_1 \in \mathcal{B}_i \Rightarrow n_0 \in \mathcal{B}_i)$. (So the nodes in scope of n_0 are the nodes that are in 'wider' boxes.)
— Node n_1 is a *top level node* if n_1 is not contained in any box.
— The *free nodes* are the nodes at the top level that have no outgoing edges.

We now define the deduction graphs that correspond to minimal proposition logic. We give an inductive definition, in the style of Gentzen and Prawitz, with the difference that now we are not dealing with a tree structure but with a graph structure, which makes the definition more involved. In the following, Form is the set of formulas from implicational logic.

Definition 2.7. The collection of *deduction graphs for minimal proposition logic* is the set of closed box directed graphs over $\mathbf{N} \times \mathsf{Form}$ inductively defined as follows.

Axiom A single node (n, A) is a deduction graph,
Join If G and G' are disjoint deduction graphs, then $G'' := G \cup G'$ is a deduction graph.

→-E If G is a deduction graph containing two nodes $(n, A{\to}B)$ and (m, A) at the top level, then the graph $G' := G$ with

— a new node (p, B) at the top level

— an edge $(p, B) \longrightarrow\!\!\triangleright (n, A{\to}B)$,

— an edge $(p, B) \longrightarrow\!\!\triangleright (m, A)$,

is a deduction graph. (See Figure 3, the left part.)

→-I If G is a deduction graph containing a node (j, B) with no ingoing edges and a finite set of free nodes with label A, $(n_1, A), \ldots, (n_k, A)$, all at the top level, then the graph $G' := G$ with

— A box $\mathcal{B}$ with box-node $(n, A{\to}B)$, containing the nodes (j, B) and (n_1, A), $\ldots, (n_k, A)$ and no other nodes that were free in G,

— An edge from the box node $(n, A{\to}B)$ to (j, B)

is a deduction graph under the proviso that it is a well-formed closed box directed graph. (See Figure 3, the right part.)

Repeat If G is a deduction graph containing a node (n, A) at the top level, the graph $G' := G$ with

— a new node (m, A) at the top level,

— an edge $(m, A) \longrightarrow\!\!\triangleright (n, A)$

is a deduction graph.

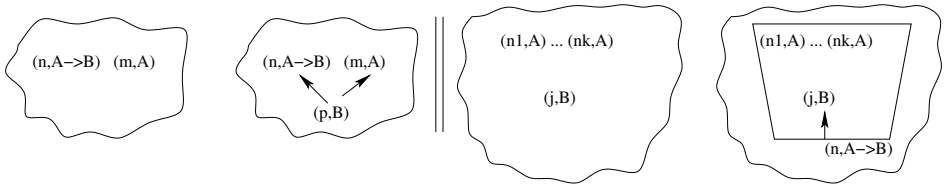

Fig. 3. The →-E and →-I rule of deduction graphs

From now on, we will simply write "deduction graph" instead of "deduction graph for minimal proposition logic".

Example 2.8. We give an example of a deduction graph. In the Figure 4, the part H is some unspecified part of the graph that contains a node $(7, A)$. This deduction graph can be seen as a derivation of B from an assumption $A{\to}A{\to}B$ (assuming a derivation H of A).

Lemma 2.9. Every deduction graph is acyclic.

Proof. Number the nodes in the order in which they have been introduced. Note that Definition 2.7 only allows edges $m \longrightarrow\!\!\triangleright n$ if $m > n$.

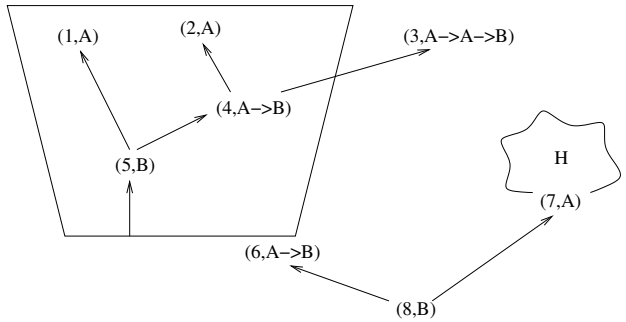

Fig. 4. Example of a deduction graph

Example 2.10. In the deduction graph of Example 2.8 (Figure 4), if we ignore the non-depicted nodes of H, the usual linear ordering of the naturals itself can be used to give a box-topological ordering. More generally, each linear ordering that satisfies the constraints $8 > 7$, $8 > 6 > 5 > 4$, $5 > 1$, $4 > 2$, $4 > 3$ is a box-topological ordering of the deduction graph.

In the rightmost graph of Figure 2, the only possible box-topological ordering of the graph is $6 > 4 > 1 > 2 > 3 > 5$.

The following Lemma enables us to prove that a given graph G is a deduction graph without explicitly supplying a construction. If we (over)simplify the statement of the Lemma, it basically says that we just have to check that G is a non-cyclic closed box directed graph and that each node of G is of one of four possible types (A, E, I or R, to be defined in the Lemma). These types correspond to the Axiom, Join, $\rightarrow$-I and $\rightarrow$-E construction cases of Definition 2.7. The Lemma also makes it possible to define a function inductively on nodes of a deduction graph, independently of how the graph has been created.

Lemma 2.11. *G is a deduction graph if and only if the following hold*

1 *G is a finite closed box directed graph,*
2 *there is a box-topological ordering $>$ of G,*
3 *every node n of G is of one of the following four types:*

- A *n has no outgoing edges.*
- E *n has label B and has exactly two outgoing edges: one to a node $(m, A{\rightarrow}B)$ and one to a node (p, A), both within the scope of n.*
- I *n is a box-node of a box $\mathcal{B}$ with label $A{\rightarrow}B$ and has exactly one outgoing edge, which is to a node (j, B) inside the box $\mathcal{B}$ with no other ingoing edges. All nodes inside the box without outgoing edges have label A.*
- R *n has label A and has exactly one outgoing edge, which is to a node (m, A) that is within the scope of n.*

Proof. $\Rightarrow$: By induction on the construction of deduction graphs (Definition 2.7). For every construction case for deduction graphs we have to check the three properties stated

in the Lemma. Properties (1) and (3) are immediate (note that for the $\to$-I case, property (1) is enforced by the definition). For property (2), we know from the induction hypothesis that there is a box-topological ordering $>$ on smaller graphs. In the construction cases Axiom, Repeat, $\to$-I or $\to$-E, we make the new node that is introduced highest in the $>$-odering, which yields a box-topological ordering on the new graph. In the construction case Join, we have two box topological orderings, $>_1$ on G_1 and $>_2$ on G_2. Then $G_1 \cup G_2$ can be given a box-topological ordering by taking the union of $>_1$ and $>_2$ and in addition putting $n > m$ for every $n \in G_1, m \in G_2$.

$\Leftarrow$: By induction on the number of nodes of G. Let $>$ be the box-topological order that is assumed to exist. Let n be the node that is maximal w.r.t. $>$. Then n must be on the top level. When we remove node n, possibly including its box (if n is of type I), we obtain a graph G' that again satisfies the properties listed in the Lemma. By induction hypothesis we see that G' is a deduction graph. Now we can add node n again, using one of the construction cases for deduction graphs: Join if n is an A node, $\to$-E if n is an E node, $\to$-I if n is an I node Repeat if n is an R node.

Definition 2.12. Let G be a deduction graph.

1 For n a node of G, we define the *subgraph of G generated from n*, G_n, as the subgraph of G consisting of all nodes reachable from n (with boxes inherited from G).
2 For $\mathcal{B}$ a box in G with discharged nodes $(n_1, A), \ldots, (n_k, A)$, and (m, A) be a fresh node (i.e. not in $\mathcal{B}$), we define the *m-closure of $\mathcal{B}$*, $\mathcal{B}^m$, as the graph consisting of all nodes inside $\mathcal{B}$ plus all nodes that can be reached from within $\mathcal{B}$ in one step plus a new node (m, A) and arrows from $(n_1, A), \ldots, (n_k, A)$ to m.

We have the following as a simple corollary of Lemma 2.11.

Corollary 2.13.

1 If G is a deduction graph containing a node n, then G_n is also a deduction graph.
2 If G is a deduction graph containing a box $\mathcal{B}$ and m is a node not in $\mathcal{B}$, then $\mathcal{B}^m$ is also a deduction graph. (Note that in $\mathcal{B}^m$, all nodes that were not in $\mathcal{B}$ have become A-nodes.)

Deduction graphs were meant to generalise both natural deductions (from Gentzen and Prawitz) and flag deductions of Fitch. It is not difficult to see that they do.

Definition 2.14 (Natural Deductions as Deduction Graphs). Given a natural deduction Σ with conclusion B, we define a deduction graph $\bar{\Sigma}$ with a top node with label B as follows.

View all formula occurrences as a node and replace every rule of the form

$$\frac{A_1 \ldots A_k}{B}$$

by edges from (m, B) to $(n_1, A_1), \ldots (n_k, A_k)$ (where m is the node associated with this specific occurrence of B and similar for $n_1, \ldots, n_k$ and $A_1, \ldots, A_k$).

In the $\to$ introduction rule, where $A \to B$ is concluded from B, discharging a number of assumptions A, let Σ be the natural deduction with conclusion B. We may assume (by

induction) that we have a deduction graph $\bar{\Sigma}$ with top node (j, B). We now create a box $\mathcal{B}$, consisting of $\bar{\Sigma}$, and for all the top nodes (p, C) of $\bar{\Sigma}$ that correspond to occurrences of hypotheses C that are not discharged at the introduction of $A \to B$ we add a new node (p', C) outside $\mathcal{B}$ and a repeat edge from (p, C) to (p', C). This is indicated in Figure 5.

Fig. 5. From natural deductions to deduction graphs

The following is now immediate.

Lemma 2.15. If Σ is a natural deduction with conclusion B, $\bar{\Sigma}$ is a deduction graph with a top node with label B.

For the translation from Fitch deductions to deduction graphs, we first have to make precise what exactly a Fitch deduction is. We follow the definition that has been given in (Geuvers and Nederpelt 2004), which we do not repeat here. Instead we restrict to a clarifying example, see Figure 6, the left deduction Σ_1. This deduction proves $A \to D$ (on the last line) from the hypotheses $A \to (B \to C) \to D$ and $A \to C$, on lines 1 and 2. The open (i.e. nondischarged) hypotheses are indicated by a 'flag' whose pole extends to the line of the conclusion. Discharging a hypothesis corresponds to ending the flag pole. On line 7, the flag pole of the hypothesis B ends and we conclude $B \to C$ from C, while ending the scope of the B flag. The comments on the right give the *motivation* for the conclusion on the line, referring to a logical operation and previous lines. The lines that can be referred to in a motivation should be *in scope*, a notion that should be intuitively clear. In the Fitch deduction Σ_3 we see one additional rule, which is the *repeat* rule. This rule allows to repeat a formula that has been derived on a previous line, if it is in scope.

Definition 2.16 (Fitch Deductions as Deduction Graphs). Given a Fitch deduction Σ with conclusion B, we define a deduction graph $\hat{\Sigma}$ with a top node with label B as follows.

View a formula occurrence B on line n as a node (n, B) and then add edges as follows: if $\to$-E, p, q is the motivation on line n, add edges from (n, B) to nodes p and q; if $\to$-I, p, q is the motivation on line n, add an edge from (n, B) to node q; if R, p is the motivation on line n, add an edge from (n, B) to node p. Now, if there is a flag that starts at line i and is discharged at line $j + 1$, put a box around nodes $i, \ldots, j$.

This creates a deduction graph, but there is one slight subtlety: in flag deductions, if we introduce $A \to B$, B does not have to be under the 'flag' A, it only has to be 'in scope'.

1	$A\to(B\to C)\to D$	
2	$\quad A\to C$	
3	$\quad\quad A$	
4	$\quad\quad (B\to C)\to D$	$\to$E, 1, 3
5	$\quad\quad\quad B$	
6	$\quad\quad\quad C$	$\to$E, 2, 3
7	$\quad\quad B\to C$	$\to$I, 5, 6
8	$\quad\quad D$	$\to$E, 4, 7
9	$\quad A\to D$	$\to$I, 3, 8

1	B	
2	$(A\to B)\to C$	
3	$\quad A$	
4	$\quad A\to B$	$\to$I, 3, 1
5	C	$\to$E, 2, 4

1	B	
2	$(A\to B)\to C$	
3	$\quad A$	
4	$\quad B$	R, 1
5	$A\to B$	$\to$I, 3, 4
6	C	$\to$E, 2, 5

Fig. 6. Fitch deductions Σ_1, Σ_2 ans Σ_2

(See flag deduction Σ_2 in Figure 6.) In this case, we first add a 'Repeat' step to the flag deduction and then we can perform the translation.

The following Lemma is now immediate, as can be seen from the examples in Figure 7, which are translations of the Fitch deductions of Figure 6.

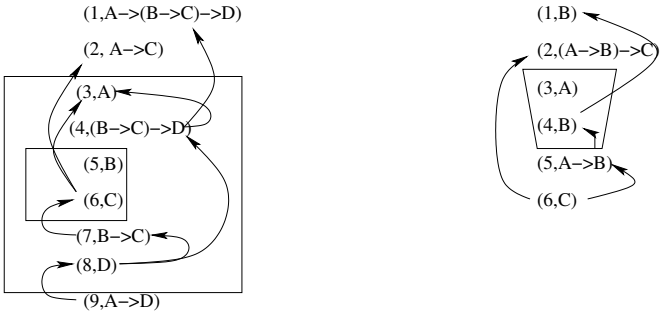

Fig. 7. From flag deductions to deduction graphs

Lemma 2.17. If Σ is a Fitch deduction with conclusion B, $\hat{\Sigma}$ is a deduction graph with a top node with label B.

3. Cut-elimination

We now define the notion of "cut" in deduction graphs. We want to be able to contract cuts in the usual natural-deduction way, thus a cut should be something like an $\to$-I immediately followed by an $\to$-E of the same formula. We want to eliminate such a cut by reordering the edges that are involved in the cut, and removing some. Basically, this amounts to Figure 8. However, it could also happen that the $\to$-I and the $\to$-E are separated by several Repeats.

Definition 3.1. A *cut* in a deduction graph G is a subgraph of G consisting of

— A box-node $(n, A\to B)$,
— A node (p, B),

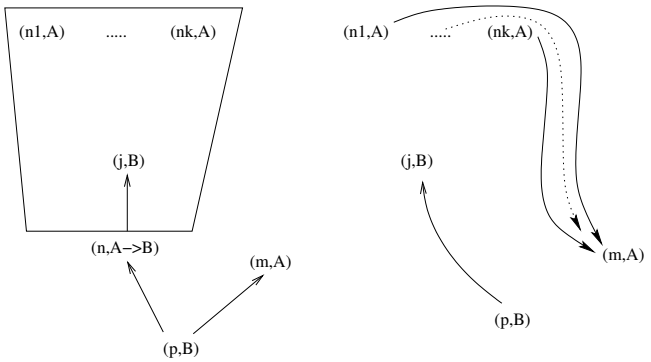

Fig. 8. Cut-elimination in deduction graphs

— A node (m, A),
— A sequence of R-nodes $(s_0, A{\to}B), \ldots, (s_i, A{\to}B)$,
— Edges $(p, B) \longrightarrow\!\triangleright (s_i, A{\to}B) \longrightarrow\!\triangleright \ldots \longrightarrow\!\triangleright (s_0, A{\to}B) \longrightarrow\!\triangleright (n, A{\to}B)$,
— An edge $(p, B) \longrightarrow\!\triangleright (m, A)$.

The operation sketched in Figure 8 is only defined on cuts where the $\to$-I is immediately followed by the $\to$-E on the same formula. Besides that, it may not yield a deduction graph again: when the box-node $(n, A{\to}B)$ has other incoming edges, these incoming edges are now "dangling" and when the node (m, A) is at an equal or greater depth than the nodes $(n_1, A), \ldots, (n_k, A)$, the edges that connect them are pointing into a box (which is forbidden). See Figure 9, where three possible positions of the cut w.r.t. boxes are depicted. In the third case, we have a *depth conflict* and we can not contract the cut. We define a "safety" criterion for when the operation of cut-elimination (as depicted in Figure 8) can be performed without damaging the deduction graph. In the general situation we can also perform cut-elimination, but then we (usually) first have to do some "R-elimination", "unsharing" and "incorporation" steps.

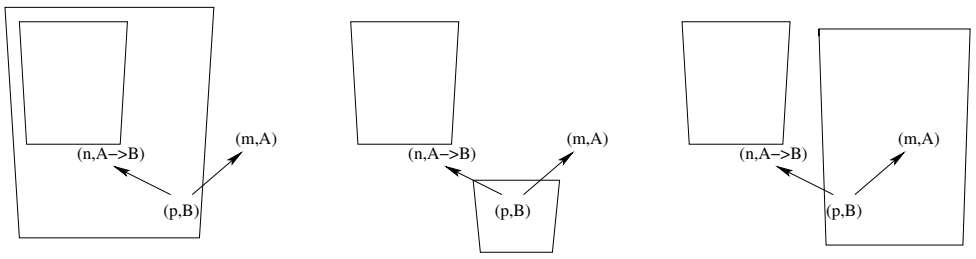

Fig. 9. Three possible positions of the cut w.r.t. boxes. The third presents a depth-conflict.

Definition 3.2. A cut in a graph G is *safe* if the following requirements hold:
— there is an edge from (p, B) to $(n, A{\to}B)$ and that is the only edge to $(n, A{\to}B)$;

— the node (m, A) is in scope of $(n, A{\to}B)$.

The process of *eliminating a safe cut* is the following operation on a deduction graph.

— remove the edges to and from n,
— remove the edge from (p, B) to (m, A),
— remove the box node n
— add an edge from (p, B) to (j, B) (the B-node inside the box that n pointed to),
— add edges from $n_1, \ldots, n_k$ (the free A-nodes inside the box) to (m, A).

Example 3.3. In the deduction graph of Example 2.8, there is a cut that we can eliminate. If we do so we obtain the following deduction graph.

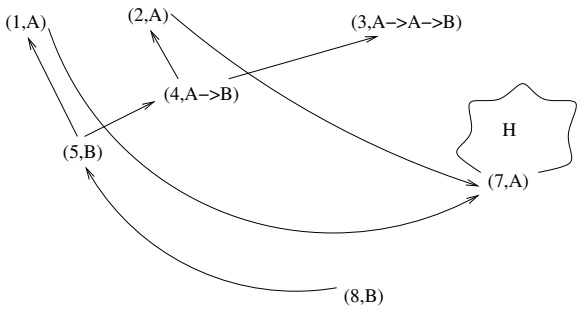

Fig. 10. Deduction graph of Figure 4 after a cut-elimination step

Lemma 3.4. If G is a deduction graph with cut c and G' is obtained from G by eliminating c, then G' is also a deduction graph.

Proof. We use Lemma 2.11. All nodes in G' are of the right form: A, E, I or R. To see that G' is a closed box directed graph, we verify the property that there are no edges pointing into a box: (m, A) is in scope of $(n, A{\to}B)$, so (also after removing the box node), (m, A) is in scope of $n_1, \ldots, n_k$ and the edges from $n_1, \ldots, n_k$ to m do not point into a box. Finally, to find a topological order on G', note that there is no edge from m to n and no edge from m to any node inside the eliminated box $\mathcal{B}$. So, without loss of generality, we may assume that $l > m$ for all $l \in \mathcal{B}$.

If a deduction graph is not safe, we first want to make it safe before performing a cut-elimination step. A first problem with non-safe deduction graphs is that the $\to$-I step and the $\to$-E step could be separated by Repeats. See Figure 11 for an example. To eliminate the cut that arises from $A{\to}B$ and A, we first have to eliminate the Repeats.

Definition 3.5 (Cut hidden by repeats). Let G be a deduction graph that contains a node $(n_0, A{\to}B)$, an R-node $(n_1, A{\to}B)$, a node (n_2, B), a node (n_3, A) and edges $n_1 \longrightarrow\!\triangleright\, n_0$, $n_2 \longrightarrow\!\triangleright\, n_1$ and $n_2 \longrightarrow\!\triangleright\, n_3$. The repeat-elimination at n_0, n_1, n_2 is obtained by:

— When an edge points to n_1, redirect it to n_0;
— Remove n_1.

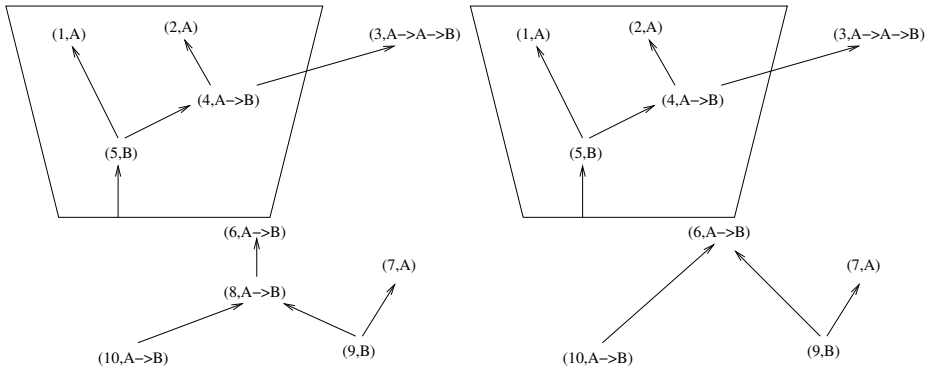

Fig. 11. Ded. graph with a cut hidden by a repeat and the same graph with the cut made explicit

Lemma 3.6. For G a deduction graph, the repeat-elimination of G is also a deduction graph.

Another problem arises when a boxed part is "shared", see Figure 12.

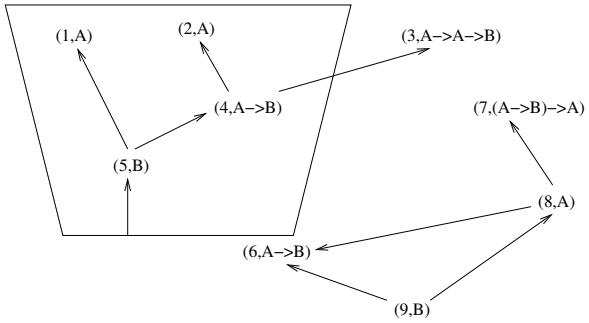

Fig. 12. Deduction graph with a cut hidden by sharing

Definition 3.7 (Cut hidden by sharing). Let G be a deduction graph that contains a box $\mathcal{B}$ with box-node $(n, A{\rightarrow}C)$ and $k \geq 2$ ingoing edges, from $p_1, \ldots, p_k$. Then the *unsharing of G at nodes $n, p_1, \ldots, p_k$* is obtained by (see Figure 13)

— making a box $\mathcal{B}'$, that contains a copy of all nodes and edges of $\mathcal{B}$,
— copy all the outgoing edges of $\mathcal{B}$ to $\mathcal{B}'$ (thus if we had $q \longrightarrow\!\triangleright\, m$ with $q \in \mathcal{B}$, $q' \in \mathcal{B}'$ and $m \notin \mathcal{B}$, then we add $q' \longrightarrow\!\triangleright\, m$, where q' is the copy of q in $\mathcal{B}'$),
— letting $p_2, \ldots p_k$ point to n' (the box-node of $\mathcal{B}'$) instead of n.

Figure 14 shows the unsharing of the deduction graph of Figure 12.

Lemma 3.8. Let G be a deduction graph that contains a box $\mathcal{B}$ with box-node n with label $A{\rightarrow}B$ and $k \geq 2$ ingoing edges, from $p_1, \ldots p_k$. Then the unsharing of G at nodes $n, p_1, \ldots, p_k$ is a deduction graph.

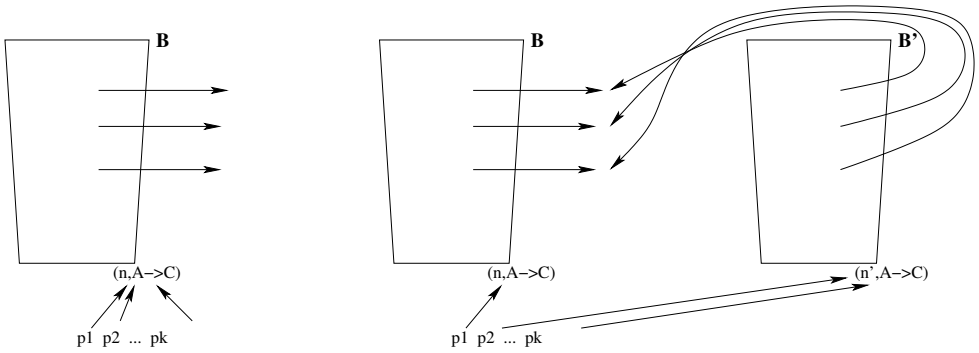

Fig. 13. Unsharing a deduction graph

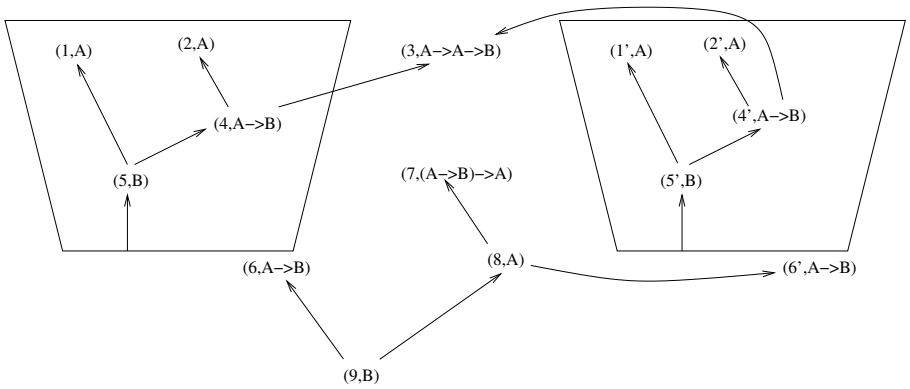

Fig. 14. The deduction graph of 12, with the cut made explicit

Proof. We use Lemma 2.11. The graph G' is a finite closed box directed graph. We make a box-topological ordering of G' by putting all nodes of $\mathcal{B}'$ immediately after the last node of $\mathcal{B}$ in the box-topological order of G. Finally, each node of G' satisfies one of the cases of Lemma 2.11.

Finally, the problem of cuts being hidden as a consequence of a depth conflict, is solved by *incorporating* a box within another. See Figure 15 for an example of a depth conflict.

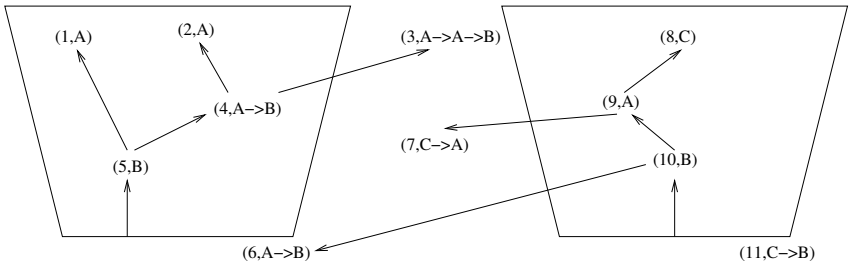

Fig. 15. Deduction graph with hidden cut due to a depth conflict

Definition 3.9 (Cut hidden by a depth conflict; incorporation). We have a *depth conflict* in the deduction graph G if G contains a box $\mathcal{B}$ with box-node $(n, A{\rightarrow}B)$ that has exactly one ingoing edge, from (p, B), and (p, B) is at a greater depth than $(n, A{\rightarrow}B)$. In that case the *incorporation of G at n, p* is obtained by moving box $\mathcal{B}$ into one deeper box that includes (p, B).

Figure 16 shows the incorporation of the deduction graph of Figure 15.

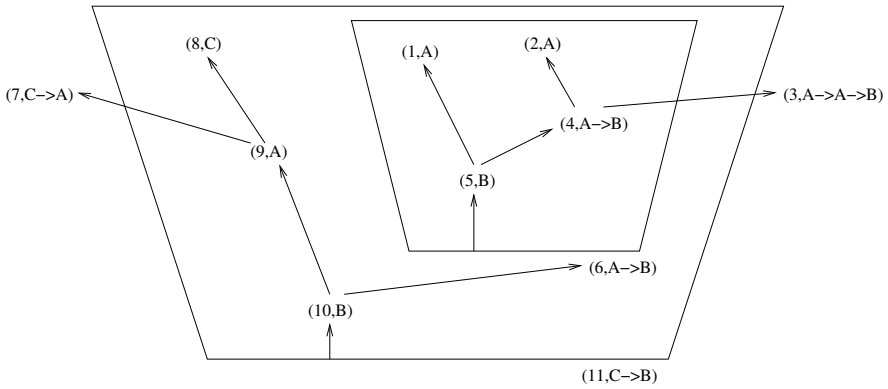

Fig. 16. Deduction graph of Figure 15 with the cut made explicit via an incorporation.

Lemma 3.10. Let G be a deduction graph that contains a depth conflict at nodes n, p. The incorporation of G at n, p is a deduction graph.

Proof. Because no other edges point to n, the resulting graph is a finite closed box directed graph. The nodes in the resulting graph are all of the same type as they were in G. Also the topological order can be taken the same as for G.

Definition 3.11. Given a deduction graph G with a cut c, *the process of eliminating the cut c* is the following.

1 (Making explicit cuts hidden by repeat) As long as there is no edge from (p, B) to $(n, A{\rightarrow}B)$, perform the appropriate repeat-elimination step as described in Definition 3.5;
2 (Making explicit cuts hidden by sharing) If there is an edge from (p, B) to $(n, A{\rightarrow}B)$ and this is not the only edge to $(n, A{\rightarrow}B)$, perform the appropriate unsharing step, as defined in Definition 3.7;
3 (Making explicit cuts hidden by a depth conflict) As long as (m, A) is at a greater depth then $(n, A{\rightarrow}B)$ perform an incorporation step, as described in Definition 3.9;
4 If c is safe, perform the cut-elimination step as defined in Definition 3.2.

4. Computational content: from deduction graphs to λ-terms

4.1. *Translation to simply typed λ-terms*

We now map deduction graphs to λ-terms. Every node in a deduction graph can be mapped to a term, the term corresponding to the proof of the formula in that node.

Definition 4.1. Given a deduction graph G and a node n in G, we define the λ-term $[\![G, n]\!]$ as follows (by induction on the number of nodes of G).

A If (n, A) has no outgoing edges, $[\![G, n]\!] := x^A_n$,
E If $(n, B) \longrightarrow\!\triangleright (m, A{\to}B)$, and $(n, B) \longrightarrow\!\triangleright (p, A)$, define $[\![G, n]\!] := [\![G_m, m]\!]\,[\![G_p, p]\!]$.
I If $(n, A{\to}B)$ is a box-node with $(n, A{\to}B) \longrightarrow\!\triangleright (j, B)$ and the free nodes of the box are $(n_1, A), \ldots, (n_k, A)$, define
$[\![G, n]\!] := \lambda x{:}A.([\![G_j, j]\!][x_{n_1} := x, \ldots, x_{n_k} := x])$.
R If $(n, A) \longrightarrow\!\triangleright (m, A)$, define $[\![G, n]\!] := [\![G_m, m]\!]$

Remark that $[\![G, n]\!] = [\![G_n, n]\!]$ for all n. This mapping ignores quite a lot of structure of the graph. The following Lemma is an immediate consequence of the definitions.

Lemma 4.2. Let G, G' be a deduction graph and let n be a node of G and G'. Then:
— If G' is obtained from G by an unsharing step, then $[\![G, n]\!] = [\![G', n]\!]$.
— If G' is obtained from G by an incorporation step, then $[\![G, n]\!] = [\![G', n]\!]$.
— If G' is obtained from G by an R-elimination step, then $[\![G, n]\!] = [\![G', n]\!]$.

We now connect the process of cut-elimination on deduction graphs with β-reduction on simply typed λ-terms. We first relate substitution in λ-terms to a notion of substitution in deduction graphs. In deduction graphs, substitution is performed by turning an A-node into an R-node, by linking it to a node with the same label.

Definition 4.3. Let G and G' be two deduction graphs and suppose that $m_0, \ldots, m_k$ are the free nodes of G. The function f from the nodes of G to the nodes of G' is a *deduction graph substitution* (or *dg-substitution*) if the following hold for all nodes m, n_0, n_1 of G.

1 the label of m is the label of $f(m)$;
2 if $n_0 \longrightarrow\!\triangleright n_1$, then $f(n_0) \longrightarrow\!\triangleright f(n_1)$;
3 $m \in \mathcal{B}$ iff $f(m) \in f(\mathcal{B})$;
4 If $m \neq m_0, \ldots m_k$ is an A/E/I/R node, then $f(m)$ is an A/E/I/R node too.

In the third clause of the Definition, we use $f(\mathcal{B})$. This clause should obviously be understood as '$m \in \mathcal{B}$ where $\mathcal{B}$ has box node n iff $f(m) \in \mathcal{B}'$ where $\mathcal{B}'$ has box node $f(n)$'.

A dg-substitution cannot change much of the graph: G' may contain parts that G doesn't have and G' may have some more sharing than G. But most importantly, a top level A node of G may be 'replaced by' some other node. The correspondence with substitution is stated in the following Lemma.

Lemma 4.4. Let G and G' be two deduction graphs with $m_0, \ldots, m_k$ the free nodes of G and n a top-level node of G. Let f be a dg-substitution from G to G'. Then the following holds.

$$[\![G', f(n)]\!] = [\![G, n]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]]$$

where $[\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]]$ is an abbreviation for $[x_{m_0} := [\![G', f(m_0)]\!], \ldots, x_{m_k} := [\![G', f(m_k)]\!]]$.

Proof. The proof is by induction on the number of reachable nodes from n. By $\overset{\text{IH}}{=}$ we indicate an application of the induction hypothesis.

A If (n, A) has no outgoing edges, then there is an i such that $n = m_i$.

$$\begin{array}{rcl} [\![G', f(m_i)]\!] & = & x_{m_i}[x_{m_i} := [\![G', f(m_i)]\!]] \\ & = & [\![G, m_i]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]]. \end{array}$$

E If $(n, B) \longrightarrow\!\triangleright (q, A{\to}B)$, and $(n, B) \longrightarrow\!\triangleright (p, A)$, then

$$\begin{array}{rcl} [\![G', f(n)]\!] & = & [\![G', f(q)]\!][\![G', f(p)]\!] \\ & \overset{\text{IH}}{=} & ([\![G', q]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]])([\![G, p]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]]) \\ & = & ([\![G, q]\!][\![G, p]\!])[\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]] \\ & = & [\![G, n]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]] \end{array}$$

I If $(n, A{\to}B)$ is a box-node with $(n, A{\to}B) \longrightarrow\!\triangleright (j, B)$ and the free nodes of the box are $(n_1, A), \ldots, (n_l, A)$, then $f(n_1), \ldots, f(n_l)$ are A-nodes inside the box with box-node $f(n)$. We use $[\overrightarrow{x_n} := x]$ as abbreviation for $[x_{n_1} := x, \ldots, x_{n_l} := x]$, we use $[\overrightarrow{x_{f(n)}} := x]$ as abbreviation for $[x_{f(n_1)} := x, \ldots, x_{f(n_l)} := x]$ and we use $[\overrightarrow{x_n} := [\![G', \overrightarrow{f(n)}]\!]]$ as abbreviation for $[x_{n_1} := [\![G', f(n_1)]\!], \ldots, x_{n_l} := [\![G', f(n_l)]\!]]$.

$$\begin{array}{rcl} [\![G', f(n)]\!] & = & \lambda x{:}A.([\![G, f(j)]\!][\overrightarrow{x_{f(n)}} := x]) \\ & \overset{\text{IH}}{=} & \lambda x{:}A.(([\![G, j]\!][\overrightarrow{x_n} := [\![G', \overrightarrow{f(n)}]\!]][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]])[\overrightarrow{x_{f(n)}} := x]) \\ & \overset{*}{=} & \lambda x{:}A.(([\![G, j]\!][\overrightarrow{x_n} := x])[\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]]) \\ & = & (\lambda x{:}A.([\![G, j]\!][\overrightarrow{x_n} := x]))[\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]] \\ & = & [\![G, n]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]] \end{array}$$

The equality $\overset{*}{=}$ uses the fact that $\overrightarrow{m}$ and $\overrightarrow{n}$ are disjoint and that $[\![G', f(n_i)]\!] = x_{f(n_i)}$ (because n_i is an A node inside a box and so $f(n_i)$ is an A node too).

R If $(n, A) \longrightarrow\!\triangleright (q, A)$, then

$$\begin{array}{rcl} [\![G', f(n)]\!] & = & [\![G', f(q)]\!] \\ & \overset{\text{IH}}{=} & [\![G, q]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]] \\ & = & [\![G, n]\!][\overrightarrow{x_m} := [\![G', \overrightarrow{f(m)}]\!]]. \end{array}$$

Lemma 4.4 is used in the proof of the following lemma:

Lemma 4.5 (Cut-elimination is β-reduction). If $G \twoheadrightarrow_{\text{cut}} G'$, then $[\![G, n]\!] \twoheadrightarrow_{\beta} [\![G', n]\!]$.

The lemma can be made more precise: if there is a path from n to the cut-formula, then we know that the β-reduction is not empty. If the cut-formula is not reachable from n, the reduction is a zero-step reduction and the right-hand side and left-hand side are the same.

4.2. *Strong normalisation of cut-elimination*

As the reduction in Lemma 4.5 may be empty, we cannot conclude strong normalisation for cut-elimination from strong normalisation for β. A seemingly simple way to solve this is to consider the set $[\![G, n_1]\!], \ldots, [\![G, n_p]\!]$ for all top level nodes without incoming edges $n_1, \ldots, n_p$, and to prove that if $G \twoheadrightarrow_{\mathrm{cut}} G'$, then one of the terms $[\![G, n_i]\!]$ does a real reduction step. (So then the sum of the lengths of the maximal reduction sequences from $[\![G, n_1]\!], \ldots, [\![G, n_p]\!]$ may be used as a measure.) The problem with this reasoning is twofold:

1 If we do a cut-elimination where the $\to$-I is empty (there are no $n_1, \ldots, n_k$), we create a new top node (the (m, A) that we 'cut' with) without incoming edges.
2 A box may contain nodes without ingoing edges. These become (new) top nodes after a cut-elimination step.

Problem (1) cannot just be solved by taking $[\![G, n]\!]$ for all nodes n, because in the unsharing phase, nodes may get copied. Problem (2) cannot be solved by taking $[\![G, n]\!]$ for all nodes without incoming edges (including the ones inside boxes), because if the box gets removed – due to a cut-elimination step – we have to substitute inside $[\![G, n]\!]$.

The solution is to collect in one "term" all the λ-terms that arise from a node without incoming edges. To do that we extend the simply typed λ-calculus with a new term construction $\langle -, \ldots \rangle$ with the following rule.

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N_1 : \tau_1 \quad \ldots \quad \Gamma \vdash N_k : \tau_k}{\Gamma \vdash \langle M, N_1, \ldots, N_k \rangle : \sigma}$$

Definition 4.6. We define $\lambda{\to}\langle\rangle$ as the λ-calculus with the tupling constructor $\langle -, \ldots \rangle$ and the above derivation rule. The reduction rules for $\lambda{\to}\langle\rangle$ are as follows.

$$\begin{aligned}
(\lambda x{:}\sigma.M)N &\twoheadrightarrow_{\bar\beta} M[x := N] \text{ if } x \in \mathrm{FV}(M) \\
(\lambda x{:}\sigma.M)N &\twoheadrightarrow_{\bar\beta} \langle M, N \rangle \text{ if } x \notin \mathrm{FV}(M) \\
\langle M, P_1, \ldots, P_k \rangle N &\twoheadrightarrow_{\bar\beta} \langle MN, P_1, \ldots, P_k \rangle \\
\langle \ldots, \langle M, P_1, \ldots, P_k \rangle, N_1, \ldots, N_p \rangle &\twoheadrightarrow_{\bar\beta} \langle \ldots, M, P_1, \ldots, P_k, N_1, \ldots, N_p \rangle
\end{aligned}$$

As can be observed from the typing and the reduction rules, the $\vec{N}$ in $\langle M, \vec{N} \rangle$ act as a kind of 'garbage'. The order of the terms in $\vec{N}$ is irrelevant and we therefore consider terms *modulo* permutation of these vectors, which we will write as $\equiv_p$.

We now give an interpretation $\langle\!|-|\!\rangle$ of deduction graphs as $\lambda{\to}\langle\rangle$-terms and we show that a cut-elimination step for graph deductions is mimicked by a (non-empty) $\bar\beta$-reduction. Then we show that $\bar\beta$-reduction is strongly normalising and we conclude that cut-elimination for deduction graphs is strongly normalising.

Definition 4.7. Given a deduction graph G and a node n in G, we define the λ-term $\langle\!|G, n|\!\rangle$ as follows (by induction on the number of nodes of G).

A If (n, A) has no outgoing edges, $\langle\!|G, n|\!\rangle := x_n^A$,
E If $(n, B) \longrightarrow\!\!\triangleright (m, A{\to}B)$, and $(n, B) \longrightarrow\!\!\triangleright (p, A)$, define $\langle\!|G, n|\!\rangle := \langle\!|G, m|\!\rangle\, \langle\!|G, p|\!\rangle$.
R If $(n, A) \longrightarrow\!\!\triangleright (m, A)$, define $\langle\!|G, n|\!\rangle := \langle\!|G, m|\!\rangle$
I If $(n, A{\to}B)$ is a box-node with $(n, A{\to}B) \longrightarrow\!\!\triangleright (j, B)$, the free nodes of the box are $n_1, \ldots, n_k$ and the nodes without incoming edges inside the box are $m_1, \ldots, m_p$, then

$$\langle\!|G, n|\!\rangle := \lambda x{:}A.\langle\langle\!|G, j|\!\rangle, \langle\!|G, m_1|\!\rangle, \ldots, \langle\!|G, m_p|\!\rangle\rangle[x_{n_1} := x, \ldots, x_{n_k} := x].$$

The interpretation of the deduction graph G, $\langle\!|G|\!\rangle$, is defined as $\langle\langle\!|G, r_1|\!\rangle, \ldots, \langle\!|G, r_l|\!\rangle\rangle$, where $r_1, \ldots, r_l$ are the topnodes without incoming edges in the deduction graph G.

Lemma 4.8 (Cut-elimination is $\bar{\beta}$-reduction in $\lambda{\to}\langle\rangle$). If $G \twoheadrightarrow_{\mathrm{cut}} G'$, then $\langle\!|G|\!\rangle \twoheadrightarrow \twoheadrightarrow_{\bar{\beta}}^{+} \langle\!|G'|\!\rangle$.

Proof. By induction on the structure of G. Note that, if G' is obtained from G via an repeat-elimination step, an unsharing step or an incorporation step, then $\langle\!|G|\!\rangle \equiv_p \langle\!|G'|\!\rangle$. If G' is obtained from G by contracting a safe cut, say with (p, B) being the conclusion of the $\to$-E, then $\langle\!|G, p|\!\rangle$ is a subterm of $\langle\!|G|\!\rangle$, possibly several times (causing the $\twoheadrightarrow_{\bar{\beta}}^{+}$ instead of just $\twoheadrightarrow_{\bar{\beta}}$). If we look at the subterm $\langle\!|G, p|\!\rangle$ of $\langle\!|G'|\!\rangle$ itself, we first note that it occurs as a subterm of a $\langle \ldots, \ldots \rangle$ expression: $\langle \ldots, \langle\!|G, p|\!\rangle, \ldots \rangle$. We have

$$\begin{array}{rcl}
\langle\!|G, p|\!\rangle & \equiv & \langle\!|G, n|\!\rangle \langle\!|G, m|\!\rangle \\
& \equiv & (\lambda x{:}A.\langle\langle\!|G, j|\!\rangle, \langle\!|G, m_1|\!\rangle, \ldots, \langle\!|G, m_p|\!\rangle\rangle[x_{n_1} := x, \ldots, x_{n_k} := x])\langle\!|G, m|\!\rangle \\
& \twoheadrightarrow_{\bar{\beta}} & \langle\langle\!|G, j|\!\rangle, \langle\!|G, m_1|\!\rangle, \ldots, \langle\!|G, m_p|\!\rangle\rangle[x_{n_1} := \langle\!|G, m|\!\rangle, \ldots, x_{n_k} := \langle\!|G, m|\!\rangle] \\
& \equiv & \langle\langle\!|G', p|\!\rangle, \langle\!|G, m_1|\!\rangle^*, \ldots, \langle\!|G, m_p|\!\rangle^*\rangle
\end{array}$$

where the superscript * denotes the substitution $[x_{n_1} := \langle\!|G, m|\!\rangle, \ldots, x_{n_k} := \langle\!|G, m|\!\rangle]$ As a consequence of the cut-elimination, the box has been removed and the nodes $m_1, \ldots, m_p$ have moved one level up (in terms of depth), thus

$$\begin{array}{rcl}
\langle\!|G|\!\rangle & \equiv & \ldots\langle \ldots, \langle\!|G, p|\!\rangle, \ldots\rangle \ldots \\
& \twoheadrightarrow_{\bar{\beta}} & \ldots\langle \ldots, \langle\langle\!|G', p|\!\rangle, \langle\!|G, m_1|\!\rangle^*, \ldots, \langle\!|G, m_p|\!\rangle^*\rangle, \ldots\rangle \ldots \\
& \twoheadrightarrow_{\bar{\beta}} & \ldots\langle \ldots, \langle\!|G', p|\!\rangle, \langle\!|G, m_1|\!\rangle^*, \ldots, \langle\!|G, m_p|\!\rangle^*, \ldots\rangle \ldots \\
& \equiv_p & \langle\!|G'|\!\rangle
\end{array}$$

We now prove strong normalisation of $\twoheadrightarrow_{\bar{\beta}}$. The proof uses an adaptation of the standard method of saturated sets. We interpret types as sets of (strongly normalising, not necessarily well-typed) terms and we show that this interpretation is sound. All types will be interpreted as a so called *saturated set*.

Definition 4.9. A set X of $\lambda{\to}\langle\rangle$-terms is called *saturated* if

— $X \subset \mathrm{SN}_{\bar{\beta}}$,
— $\langle x\vec{P}, \vec{N}\rangle \in X$ for all variables x and all $\vec{P}, \vec{N} \subset \mathrm{SN}_{\bar{\beta}}$ (where, if $\vec{N}$ is empty, this is $x\vec{P}$).

— If $M[x := N]\vec{P} \in X$, then $(\lambda x{:}\sigma.M)N\vec{P} \in X$, if $N \in \mathrm{SN}_{\bar{\beta}}$.
— If $M \in X$, then $\langle M, \vec{P}\rangle \in X$, if $\vec{P} \subset \mathrm{SN}_{\bar{\beta}}$.
— If $\langle MN, \vec{P}\rangle\vec{R} \in X$, then $\langle M, \vec{P}\rangle N\vec{R} \in X$.

We now give an (sound) interpretation of types as saturated sets.

Definition 4.10. The interpretation of types as saturated sets $\mathcal{V}(-)$ is defined as follows.
— $\mathcal{V}(\alpha) := \mathrm{SN}_{\bar{\beta}}$
— $\mathcal{V}(\sigma{\to}\tau) := \{M \mid \forall N \in \mathcal{V}(\sigma)(MN \in \mathcal{V}(\tau))\}$

Lemma 4.11. For all types σ, $\mathcal{V}(\sigma)$ is a saturated set.

Proof. This is proved by induction on σ. If σ is a variable α, $\mathcal{V}(\alpha) = \mathrm{SN}_{\bar{\beta}}$, so we have to check that $\mathrm{SN}_{\bar{\beta}}$ is a saturated set. This is easily checked by verifying that the closure conditions all hold for $\mathrm{SN}_{\bar{\beta}}$. For arrow types we proceed by induction on the structure of types. So suppose that $\mathcal{V}(\sigma)$ and $\mathcal{V}(\tau)$ are saturated and we have to verify the closure conditions for saturated sets for $\mathcal{V}(\sigma{\to}\tau)$.

— $\mathcal{V}(\sigma{\to}\tau)$ is obviously a subset of $\mathrm{SN}_{\bar{\beta}}$ (given that $\mathcal{V}(\tau) \subset \mathrm{SN}_{\bar{\beta}}$ and that $\mathcal{V}(\sigma) \neq \emptyset$).
— $\langle x\vec{P}, \vec{N}\rangle \in \mathcal{V}(\sigma{\to}\tau)$ for all variables x and all $\vec{P}, \vec{N} \subset \mathrm{SN}_{\bar{\beta}}$, because for all $Q \in \mathcal{V}(\sigma)$ we have $\langle x\vec{P}Q, \vec{N}\rangle \in \mathcal{V}(\tau)$ and thus $\langle x\vec{P}, \vec{N}\rangle Q \in \mathcal{V}(\tau)$. So $\langle x\vec{P}, \vec{N}\rangle \in \mathcal{V}(\sigma{\to}\tau)$.
— Suppose $M[x := N]\vec{P} \in \mathcal{V}(\sigma{\to}\tau)$. Then for all $Q \in \mathcal{V}(\sigma)$, $M[x := N]\vec{P}Q \in \mathcal{V}(\tau)$, thus $(\lambda x{:}\sigma.M)N\vec{P}Q \in \mathcal{V}(\tau)$. So, $(\lambda x{:}\sigma.M)N\vec{P} \in \mathcal{V}(\sigma{\to}\tau)$.
— Suppose $M \in \mathcal{V}(\sigma{\to}\tau)$. Then for all $Q \in \mathcal{V}(\sigma)$, $MQ \in \mathcal{V}(\tau)$, so for all $Q \in \mathcal{V}(\sigma)$ and for all $\vec{N} \in \mathrm{SN}_{\bar{\beta}}$, $\langle MQ, \vec{N}\rangle \in \mathcal{V}(\sigma)$, so for all $Q \in \mathcal{V}(\sigma)$ and for all $\vec{N} \in \mathrm{SN}_{\bar{\beta}}$, $\langle M, \vec{N}\rangle Q \in \mathcal{V}(\sigma)$. We conclude that for all $\vec{N} \in \mathrm{SN}_{\bar{\beta}}$, $\langle M, \vec{N}\rangle \in \mathcal{V}(\sigma{\to}\tau)$.
— Suppose $\langle MN, \vec{P}\rangle\vec{R} \in \mathcal{V}(\sigma{\to}\tau)$, then for all $Q \in \mathcal{V}(\sigma)$, $\langle MN, \vec{P}\rangle\vec{R}Q \in \mathcal{V}(\tau)$. So for all $Q \in \mathcal{V}(\sigma)$, $\langle M, \vec{P}\rangle N\vec{R}Q \in \mathcal{V}(\tau)$ and we can conclude that $\langle M, \vec{P}\rangle N\vec{R} \in \mathcal{V}(\sigma{\to}\tau)$.

Lemma 4.12 (Soundness of $\mathcal{V}(-)$). If $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ in $\lambda{\to}\langle\rangle$ and $N_1 \in \mathcal{V}(\sigma_1), \ldots N_n \in \mathcal{V}(\sigma_n)$, then $M[x_1 := N_1, \ldots, x_n := N_n] \in \mathcal{V}(\tau)$.

Proof. By induction on the derivation. All cases are straightforward; as illustration we treat two. For simplicity we denote $M[x_1 := N_1, \ldots, x_n := N_n]$ as M^*.

abs So, $\Gamma \vdash \lambda x{:}\sigma.M : \sigma{\to}\tau$ was derived from $\Gamma, x{:}\sigma \vdash M : \tau$. As IH we have $\forall Q \in \mathcal{V}(\sigma)(M^*[x := Q] \in \mathcal{V}(\tau))$. One of the closure conditions of saturated sets then says that $\forall Q \in \mathcal{V}(\sigma)((\lambda x{:}\sigma.M^*)Q \in \mathcal{V}(\tau))$ and therefore $\lambda x{:}\sigma.M^* \in \mathcal{V}(\sigma{\to}\tau)$.

$\langle -, \ldots\rangle$ So, $\Gamma \vdash \langle M, \vec{P}\rangle : \sigma$ was derived from $\Gamma \vdash M : \sigma$ and $\Gamma \vdash \vec{P} : \vec{\tau}$. As IH we have $M^* \in \mathcal{V}(\sigma)$ and $\vec{N}^* \in \mathcal{V}(\tau)$ (and thus $\vec{N}^* \subset \mathrm{SN}_{\bar{\beta}}$). One of the closure conditions of saturated sets then says that $\langle M, \vec{P}\rangle \in \mathcal{V}(\tau)$.

Corollary 4.13. $\bar{\beta}$-reduction is strongly normalising for $\lambda{\to}\langle\rangle$ and thus cut-elimination is terminating for deduction graphs.

Proof. Take $N_i := x_i$ in the Lemma and we find that $M \in \tau \subset \mathrm{SN}_{\bar{\beta}}$, so M is strongly normalising. Henceforth cut-elimination is terminating for deduction graphs due to Lemma 4.8.

5. Computational content: from deduction graphs to λ + let-contexts

As stated in Lemma 4.2, the mapping $[\![-]\!]$ from deduction graphs to λ-terms does not reflect the structure of the deduction graph very well: it may ignore parts and does not reflect sharing in the term. The mapping $\langle\!|-|\!\rangle$ does a bit better, as all parts of the deduction graph are reflected in the term, but it also introduces duplications, because the sharing is not reflected. We would like a term:

— to show the structure of the whole graph G, instead of only the part that is relevant for some node n. This would be useful in the case of e.g. an $\rightarrow$-E rule, where we want to combine two parts of an existing graph.
— to show which parts of G are shared.
— to show which parts of G are repeated.
— to show how boxes are positioned with respect to each other.

To get a more faithful term-representation of deduction graphs, we define a mapping to a context calculus with let-expressions. As we want to represent the whole graph, we cannot take one node as a "focus" and define the translation from there (as we did for $[\![-]\!]$ in Definition 4.1). A context is a term with a "hole", an open place where we can fill in a term. If we fill in x_n (where n is a top-level node of G), we get the interpretation of G as a term *in node* n.

Before giving the interpretation, we make the calculus $\lambda\!\rightarrow$ + let precise.

We first define the set of *expressions* and the operation of *filling holes*. Then we define the subset of *terms* and *contexts*.

Definition 5.1. The syntactic domain of expressions is:

$$\text{Expressions} \quad E \;:=\; x \mid (xy) \mid [-] \mid \mathsf{let}\, x = E \,\mathsf{in}\, E \mid \lambda x{:}A.E$$

Definition 5.2. For every two expressions E_1 and E_2 we define *filling the holes of* E_1 *with* E_2, $E_1[E_2]$, as follows:

$$\begin{array}{lcllcl} x[E] & := & x & (\mathsf{let}\, x = F \,\mathsf{in}\, G)[E] & := & \mathsf{let}\, x = (F[E]) \,\mathsf{in}\, (G[E]) \\ xy[E] & := & xy & (\lambda x.F)[E] & := & \lambda x.(F[E]) \\ [-][E] & := & E & & & \end{array}$$

where E, F and G are expressions.

Definition 5.3. The calculus of contexts $\lambda\!\rightarrow$ + let is defined as follows. Types are the simple types, Typ. The syntactic domains are:

$$\text{Terms} \quad T \;:=\; x \mid (xy) \mid \lambda x{:}A.C[y] \qquad \text{Contexts} \quad C \;:=\; [-] \mid \mathsf{let}\, x = T \,\mathsf{in}\, C$$

A context C has exactly one 'hole', $[-]$. We sometimes write $C[-]$ to emphasize this. In the syntax for Terms, $C[y]$ denotes the filling of the unique hole in C with y.

Typing Rules for Terms and Contexts. As usual, Γ is the environment consisting of term variable declarations: $x_1 : A_1, \ldots, x_n : A_n$, where all the x_i are distinct. In addition we have a *let-environment* Δ, which denotes the sequence of typed let-bound variables that surrounds a context-hole. There are two judgments $\Gamma \vdash M : A$ (*the term* M *is of*

type A) and $\Gamma;\Delta \vdash C$ (*C is a well-formed context*).

$$\Gamma;<>\vdash [-] \qquad \frac{\Gamma \vdash T : A \;\; \Gamma, x{:}A;\Delta \vdash C}{\Gamma; x{:}A, \Delta \vdash \mathsf{let}\, x = T \,\mathsf{in}\, C}\ \mathsf{let}\text{-rule}$$

$$\Gamma \vdash x : A \quad \text{if } (x{:}A) \in \Gamma \qquad \frac{\Gamma, x{:}A;\Delta \vdash C}{\Gamma \vdash \lambda x{:}A.C[y] : A{\to}B}\ \lambda\text{-rule} \quad \text{if } (y{:}B) \in \Delta$$

$$\frac{\Gamma \vdash x : A{\to}B \;\; \Gamma \vdash y : A}{\Gamma \vdash xy : B}\ \text{app-rule}$$

We now list some properties that give an intuition for the $\lambda{\to}+\mathsf{let}$-calculus. The proofs are by a straightforward induction on the derivation.

Lemma 5.4.

1 Shape of well-formed contexts. If $\Gamma; x_1{:}A_1, \ldots, x_n{:}A_n \vdash C[-]$, then $C[-] \equiv \mathsf{let}\, x_1 = t_1 \,\mathsf{in}\, \ldots \mathsf{let}\, x_n = t_n \,\mathsf{in}\, [-]$ with $\Gamma \vdash t_i : A_i$ (for $1 \leq i \leq n$).
2 Substitution of variables. If $\Gamma, x : A, \Gamma';\Delta \vdash C[-]$ and $y{:}A \in \Gamma, \Gamma'$, then $\Gamma, \Gamma';\Delta \vdash C[-][x := y]$ and similarly for terms.
3 Substitution of contexts: if $\Gamma;\Delta \vdash C[-]$ and $\Gamma, \Delta;\Theta \vdash D[-]$, then $\Gamma;\Delta,\Theta \vdash C[D[-]]$.

Definition 5.5. Given a deduction graph G and a box-topological ordering φ of G we define a context in $\lambda{\to}+\mathsf{let}$ $\langle\!\langle G \rangle\!\rangle_\varphi$ as follows, by induction on the number of nodes of G.

— If $|G| = 1$ (G has only one node), $\langle\!\langle G \rangle\!\rangle_\varphi = [-]$.
— If $|G| > 1$ and the φ-maximal element, say n, is an A-node, $\langle\!\langle G \rangle\!\rangle_\varphi = \langle\!\langle G \setminus n \rangle\!\rangle_{\varphi_{|G\setminus n}}$
— If the φ-maximal element, say n, is an I-node with $(n, A{\to}B) \rightarrowtriangle (j, B)$ with associated box $\mathcal{B}$, then $\langle\!\langle G \rangle\!\rangle_\varphi = \langle\!\langle G \setminus \mathcal{B} \rangle\!\rangle_{\varphi_{|G\setminus\mathcal{B}}}[\mathsf{let}\, x_n = (\lambda x_m{:}A.\langle\!\langle \mathcal{B}^m \rangle\!\rangle_{\varphi_{|\mathcal{B}^m}}[x_j]) \,\mathsf{in}\, [-]]$, where $\varphi_{|\mathcal{B}^m}$ is $\varphi_{|\mathcal{B}}$ with some extra minimal elements;
— If the φ-maximal element, say n is an E-node with $(n, B) \longrightarrowtriangle (i, A{\to}B)$ and $(n, B) \rightarrowtriangle (j, A)$, then $\langle\!\langle G \rangle\!\rangle_\varphi := \langle\!\langle G \setminus n \rangle\!\rangle_{\varphi_{|G\setminus n}}[\mathsf{let}\, x_n = (x_i x_j) \,\mathsf{in}\, [-]]$;
— If the φ-maximal element, say n, is an R-node with $n{\twoheadrightarrow}l$, then $\langle\!\langle G \rangle\!\rangle_\varphi = \langle\!\langle G \setminus n \rangle\!\rangle_{\varphi_{|G\setminus n}}[\mathsf{let}\, x_n = x_l \,\mathsf{in}\, [-]]$

This translation 'turns the structure inside out': the variables corresponding to nodes that are first reached from the maximal node, will be deepest in the term.

Example 5.6. We give three simple deduction graphs in Figure 17 and the associated contexts in $\lambda{\to}+\mathsf{let}$. In example (III), 'garbage' occurs inside a box (it is just an involved way of proving $B{\to}B$); we have added it to show the translation on a slightly non-standard example.

The translations of these deduction graphs are (together with their context):

$$\begin{array}{lll} I & <>; x_2 : B & \mathsf{let}\, x_2 = (\lambda x_m{:}A.\mathsf{let}\, x_1 = x_m \,\mathsf{in}\, x_1) \,\mathsf{in}\, [-] \\ II & x_1 : B; x_3 : A{\to}B & \mathsf{let}\, x_3 = (\lambda x_m{:}A.\mathsf{let}\, x_2 = x_1 \,\mathsf{in}\, x_2) \,\mathsf{in}\, [-] \end{array}$$

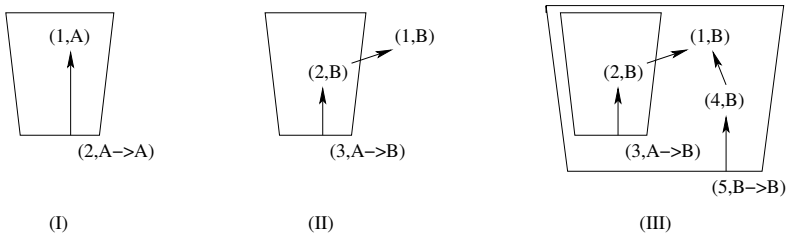

Fig. 17. Three simple deduction graphs

$$III \quad <>; x_5 : B{\to}B \quad \mathsf{let}\, x_5 = (\lambda x_p{:}B.\mathsf{let}\, x_1 = x_p\, \mathsf{in}\, \mathsf{let}\, x_4 = x_1\, \mathsf{in}\\ \mathsf{let}\, x_3 = (\lambda x_m{:}A.\mathsf{let}\, x_2 = x_1\, \mathsf{in}\, x_2)\, \mathsf{in}\, x_4)\, \mathsf{in}\, [-]$$

Lemma 5.7 (Soundness of $\langle\!\langle - \rangle\!\rangle$). If G is a deduction graph and φ is a box-topological ordering on G, then

$$\Gamma; \Delta \vdash \langle\!\langle G \rangle\!\rangle_\varphi,$$

where $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ with $(1, A_1), \ldots, (n, A_n)$ the free nodes of G, and $\Delta = y_k : B_k, \ldots, y_{k+m} : B_{k+m}$ with $(k, B_k), \ldots, (k+m, B_{k+m})$ the non-free top nodes of G. So the interpretation of G as a $\lambda{\to} + \mathsf{let}$ context yields a well-formed context indeed.

Proof. By induction on the definition of $\langle\!\langle G \rangle\!\rangle_\varphi$, using the substitution property for contexts (Lemma 5.4 (3)).

Example 5.8. We give the interpretations of the deduction graphs in Figure 4 (G) and Figure 12 (G'), including the contexts they are well-formed in. We assume that $\langle\!\langle H \rangle\!\rangle = C[-]$ in the context $\Gamma; \Delta$.

$$\begin{aligned}
\langle\!\langle G \rangle\!\rangle \;=\;& \mathsf{let}\, x_6 = (\lambda x{:}A.\mathsf{let}\, x_1 = x, x_2 = x, x_4 = (x_3 x_2), x_5 = (x_4 x_1)\, \mathsf{in}\, x_5)\, \mathsf{in}\\
& C[\mathsf{let}\, x_8 = (x_6 x_7)\, \mathsf{in}\, [-]]\\
& \text{in context } \Gamma \cup x_3{:}A{\to}A{\to}B; \Delta \cup x_7{:}(A{\to}B){\to}A, x_6{:}A{\to}B, x_8{:}B\\
\langle\!\langle G' \rangle\!\rangle \;=\;& \mathsf{let}\, x_6 = (\lambda x{:}A.\mathsf{let}\, x_1 = x, x_2 = x, x_4 = (x_3 x_2), x_5 = (x_4 x_1)\, \mathsf{in}\, x_5)\, \mathsf{in}\\
& \mathsf{let}\, x_8 = (x_7 x_6)\, \mathsf{in}\, \mathsf{let}\, x_9 = (x_6 x_8)\, \mathsf{in}\, [-]\\
& \text{in context } x_3{:}A{\to}A{\to}B, x_7{:}(A{\to}B){\to}A; x_6{:}A{\to}B, x_8{:}A, x_9{:}B
\end{aligned}$$

Note that the let-environment Δ only plays a role in restricting the variables that we are allowed to "plug in" the context when forming a λ-abstraction.

If we view the context $\langle\!\langle G \rangle\!\rangle$ as a representation of the complete deduction graph, we can obtain the "interpretation of G at node i" by plugging x_i into $\langle\!\langle G \rangle\!\rangle$, for i any top node of G. So, for x_i any top-level variable (i.e. any variable of $\Gamma \cup \Delta$), $\langle\!\langle G \rangle\!\rangle[x_i]$ is the interpretation of G at node i. Note that $\langle\!\langle G \rangle\!\rangle[x_i]$ belongs to another domain and is not a well-typed term according to Definition 5.3. If we reduce $\langle\!\langle G \rangle\!\rangle[x_i]$ by contracting (only) all let-redexes, we obtain the λ-term $[\![G, i]\!]$ of Definition 4.1.

In the translation $\langle\!\langle G \rangle\!\rangle$, separate parts of the process of cut-elimination can be recognized. We now define a reduction relation on $\lambda{\to} + \mathsf{let}$-contexts that captures cut-elimination on deduction graphs.

Definition 5.9. We define the *free variables* of expressions E, $\mathrm{FV}(E)$, as follows:

$$\begin{array}{lcllcl} \mathrm{FV}(x) & := & \{x\} & \mathrm{FV}(\mathsf{let}\, x = E_1 \,\mathsf{in}\, E_2) & := & \mathrm{FV}(E_1) \cup (\mathrm{FV}(E_2) \setminus \{x\}) \\ \mathrm{FV}(xy) & := & \{x, y\} & \mathrm{FV}(\lambda x.E) & := & \mathrm{FV}(E) \setminus \{x\} \\ \mathrm{FV}([-]) & := & \emptyset & & & \end{array}$$

Definition 5.10. We define the following (conditional) rewrite rules for expressions.

$$\begin{array}{rcll} \mathsf{let}\, x = \lambda z{:}A.D[y] \,\mathsf{in}\, \mathsf{let}\, p = (xq) \,\mathsf{in}\, E & \longrightarrow_B & D[z := q][\mathsf{let}\, p = y \,\mathsf{in}\, E] & \\ & & \text{if } D \text{ a context}, x \notin \mathrm{FV}(E) & \\ \mathsf{let}\, x = P \,\mathsf{in}\, \mathsf{let}\, y = \lambda z.Q \,\mathsf{in}\, E & \longrightarrow_{CM} & \mathsf{let}\, y = \lambda z.(\mathsf{let}\, x = P \,\mathsf{in}\, Q) \,\mathsf{in}\, E & \\ & & \text{if } x \notin \mathrm{FV}(E) & \\ \mathsf{let}\, x = P \,\mathsf{in}\, E[y := x] & \longrightarrow_{CP} & \mathsf{let}\, y = P \,\mathsf{in}\, \mathsf{let}\, x = P \,\mathsf{in}\, E & \\ & & \text{if } x, y \in \mathrm{FV}(E) & \\ \mathsf{let}\, y = x \,\mathsf{in}\, E & \longrightarrow_L & E[y := x] \quad \text{if } E \neq C[y] & \\ & & \text{for any context } C & \\ \mathsf{let}\, x = P \,\mathsf{in}\, \mathsf{let}\, y = Q \,\mathsf{in}\, E & \simeq & \mathsf{let}\, y = Q \,\mathsf{in}\, \mathsf{let}\, x = P \,\mathsf{in}\, E & \\ & & \text{if } x \notin \mathrm{FV}(Q), y \notin \mathrm{FV}(P) & \end{array}$$

We extend the rewrite rules to a reduction relation on expressions by taking the contextual closure, thus, if $E \longrightarrow_B E'$, then

$$\begin{array}{rcl} \lambda x{:}A.E & \longrightarrow_B & \lambda x{:}A.E' \\ \mathsf{let}\, x = T \,\mathsf{in}\, E & \longrightarrow_B & \mathsf{let}\, x = T \,\mathsf{in}\, E', \\ \mathsf{let}\, x = E \,\mathsf{in}\, T & \longrightarrow_B & \mathsf{let}\, x = E' \,\mathsf{in}\, T. \end{array}$$

Similarly for the other rewrite rules and for $\simeq$.

We write $\longrightarrow_{\mathrm{let}}$ for the union of these reduction rules.

Remark 5.11. Note that the contextual closure of the rewrite rules that we use in the previous Definition takes the FV-side conditions into account.

As a consequence, it may occur that $C[-] \twoheadrightarrow_B D[-]$ but *not* $\lambda u{:}B.C[x] \twoheadrightarrow_B \lambda u{:}B.D[x]$, because $x \notin \mathrm{FV}(C[-])$, allowing $C[-]$ to reduce, but $x \in \mathrm{FV}(C[x])$, disallowing $\lambda u{:}B.C[x]$ to reduce. This situation occurs for example in the term

$$t := \lambda u{:}B.(\mathsf{let}\, x = (\lambda z{:}A.z) \,\mathsf{in}\, \mathsf{let}\, p = (xq) \,\mathsf{in}\, x).$$

The context $C[-] = \mathsf{let}\, x = (\lambda z{:}A.z) \,\mathsf{in}\, \mathsf{let}\, p = (xq) \,\mathsf{in}\, [-]$ can be reduced, but the term $t := \lambda u{:}B.C[x]$ cannot (and rightly so).

Reversely, it may occur that $\lambda u{:}B.C[x] \twoheadrightarrow_{CP} \lambda u{:}B.D[x]$, but *not* $C[-] \twoheadrightarrow_{CP} D[-]$, because $x \notin \mathrm{FV}(C[-])$, disallowing $C[-]$ to reduce, but $x, y \in \mathrm{FV}(C[x])$, allowing $\lambda u{:}B.C[x]$ to reduce. This situation occurs for example in the term

$$t := \lambda u{:}B.\mathsf{let}\, x = x_1 x_2 \,\mathsf{in}\, \mathsf{let}\, x_3 = x_4 x \,\mathsf{in}\, x,$$

which CP-reduces to $\lambda u{:}B.\mathsf{let}\, y = x_1 x_2 \,\mathsf{in}\, \mathsf{let}\, x = x_1 x_2 \,\mathsf{in}\, \mathsf{let}\, x_3 = x_4 y \,\mathsf{in}\, x$, whereas the context $\mathsf{let}\, x = x_1 x_2 \,\mathsf{in}\, \mathsf{let}\, x_3 = x_4 x \,\mathsf{in}\, [-]$ does not reduce (and rightly so).

We now want to show that the well-formed terms and contexts are closed under reduction. Under reduction, a context may get transformed quite a bit, with as consequence

that the Δ in which the context is typable has to be transformed. We now 'fix' a context in which a context is well-formed.

Definition 5.12. Let $C = \mathsf{let}\, x_1 = t_1 \,\mathsf{in}\, \ldots \mathsf{let}\, x_n = t_n \,\mathsf{in}\, [-]$ be a well-formed context in $\Gamma; \Delta$ and let T be a well-formed term in Γ. The *let-context* of C is $x_1{:}A_1, \ldots, x_n{:}A_n$, where A_i is the type of x_i in Δ. We denoted the let-context of C as Δ^C.

Lemma 5.13 (Subject Reduction). The sets of well-formed contexts and well-typed terms are closed under reduction: if $\Gamma; \Delta \vdash C$ and $C \twoheadrightarrow_{\mathrm{let}} D$, then $\Gamma; \Delta^D \vdash D$. If $\Gamma \vdash M : A$ and $M \twoheadrightarrow_{\mathrm{let}} N$, then $\Gamma \vdash N : A$.

See the appendix A for a proof.

We now relate the transformations of deduction graphs to the rewrite rules of the calculus.

Lemma 5.14. If G is a deduction graph and φ and ψ are both box-topological orderings of G, then

$$\langle\!\langle G \rangle\!\rangle_\varphi \simeq \langle\!\langle G \rangle\!\rangle_\psi$$

See appendix B for the proof.

This lemma shows that $\langle\!\langle G \rangle\!\rangle_\varphi$ is independent of the box-topological ordering φ, up to the equivalence defined in Definition 5.10. It allows us to speak of $\langle\!\langle G \rangle\!\rangle$ and leave the box-topological ordering unspecified, if we consider terms modulo $\simeq$.

Lemma 5.15. Let G and G' be two deduction graphs.

1 If G' is obtained from G by an unsharing step, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CP} \langle\!\langle G' \rangle\!\rangle$.
2 If G' is obtained from G by an incorporation step, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CM} \langle\!\langle G' \rangle\!\rangle$.
3 If G' is obtained from G by a repeat eliminating step, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{L} \langle\!\langle G' \rangle\!\rangle$.
4 If G' is obtained from G by eliminating a safe cut, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{B} \langle\!\langle G' \rangle\!\rangle$.

See Appendix C for a proof.

6. Future work

6.1. *Relation with proof nets*

Deduction graphs, like proof nets of multiplicative exponential linear logic (MELL) (Girard 1987), have boxes. And just as in proof nets, these border a part of the graph that is affected by a non-local operation, so it is an interesting question whether the similarities go beyond the (superficial) fact that both have boxes. It should be pointed out that the boxes serve a different non-local purpose in the two systems: in deduction graphs, this is the 'discharging' of a hypothesis, in proof nets this is the 'resource-sensitivity' of the terminal nodes, so the correspondence is not at all immediate. When we compare transformations that are involved in cut-elimination, however, the similarities between these two formalisms seem to extend to this area too.

Let us remind the formulas and construction of proof nets of MELL. We start with a set ATOM of atomic formulas, together with the linear negation $\perp : \mathsf{ATOM} \to \mathsf{ATOM}$ and

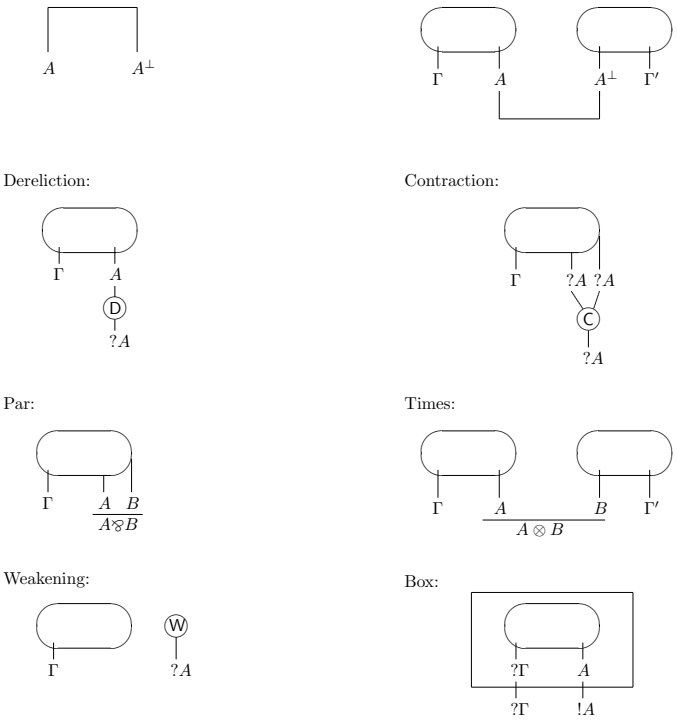

Fig. 18. Proof nets

postulate that $A^{\perp\perp} = A$ for every $A \in \mathsf{ATOM}$. The formulas are given by the following grammar:

$$\mathcal{F} ::= A \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \parr \mathcal{F} \mid !\mathcal{F} \mid ?\mathcal{F}$$

where $A \in \mathsf{ATOM}$. The linear negation is now extended to the set of all formulas as follows:

$$\begin{array}{lcllcl} (?A)^{\perp} & = & !(A^{\perp}) & \qquad (A \otimes B)^{\perp} & = & A^{\perp} \parr B^{\perp} \\ (!A)^{\perp} & = & ?(A^{\perp}) & \qquad (A \parr B)^{\perp} & = & A^{\perp} \otimes B^{\perp} \end{array}$$

For the definition of proof nets, the "natural deduction" counterpart of linear logic derivations, which are usually given in sequent form, see Fig 18. Especially notice the Contraction rule, which expresses sharing. Remark also that there is a Cut rule. So cut-elimination in proof nets expresses the redundancy of the this rule, by giving a procedure for eliminating all occurrences of this rule from a proof net. In contrast, in deduction graphs cut-elimination states that we can get rid of a situation where we have an introduction rule immediately followed by an elimination rule.

We follow (Girard 1987) in the definition of the transformations on proof nets. See the Figures 19 – 24. So the c-b rule duplicates a box, just like the unsharing rule on deduction graphs; the b-b rule absorbs one box into another, like the incorporation rule; the d-b rule opens a box, similar to what happens during the elimination of a safe cut

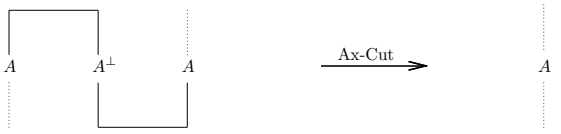

Fig. 19. The Ax-cut rule

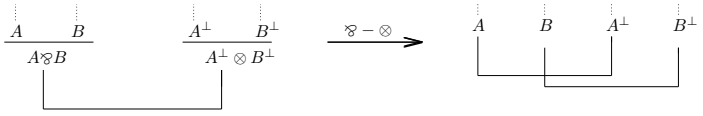

Fig. 20. The $\parr - \otimes$ rule

in deduction graphs; the Ax-cut rule removes the repetition of a formula, like the repeat elimination rule.

There is, however, one difference that causes difficulties in making this connection concrete: sharing is *implicit* in deduction graphs, but *explicit* in proof nets.

A distinction between deduction graphs and other graph formalisms, such as proof nets (but also bigraphs (Milner 2004), sharing graphs (Asperti and Guerrini 1998) and interaction nets (Lafont 1990)), is that nodes in the latter have a fixed arity, whereas nodes in the former have not. Sharing in deduction graphs is expressed by having more than one incoming edge. In proof nets, sharing is explicit and is done via a special type of node, the contraction node. Moreover, contractions can only be performed on formulas of a certain shape ($?A$). This difference emphasizes that deduction graphs are meant for resource-insensitive proposition logic and are not a way to depict resource-sensitive linear logic proofs. In natural deduction formalisms, even in the ones like Fitch style flag deductions that do have a notion of sharing, there is no explicit construct to express it. An advantage of the fact that deduction graphs do not have explicit sharing is that many subgraphs that naturally arise when reasoning about deduction graphs, like G_n of Definition 2.12, are deduction graphs themselves.

To better understand the relation between deduction graphs and proof nets, it would be interesting to study deduction graphs with an explicit sharing construction and its behaviour under cut-elimination. Correspondingly, we would like to formulate a $\lambda{\rightarrow}{+}\mathsf{let}$-calculus with explicit contractions (see (Kesner and Lengrand 2005) for a calculus with

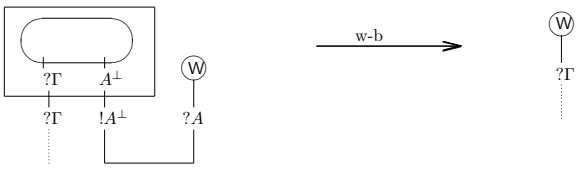

Fig. 21. The w-b rule

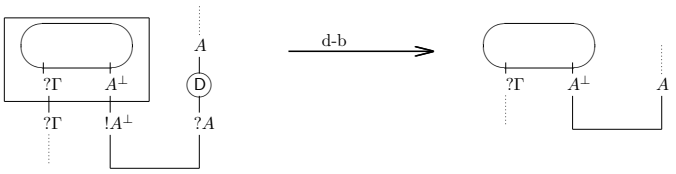

Fig. 22. The d-b rule

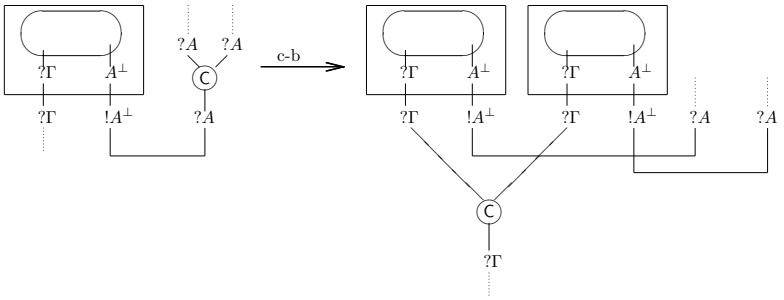

Fig. 23. The c-b rule

explicit contractions). From these enriched structures, the similarities with proof nets should be made concrete. Then we could also investigate the meaning of the process of cut-elimination on deduction graphs (Definition 3.11) on the proof net side.

6.2. *Other future work*

In the present paper we prove the termination of cut-elimination in deduction graphs by a translation to an adapted simple typed λ-calculus. This is a bit ad hoc. The other reduction preserving map that we have defined maps deduction graphs to contexts in $\lambda\!\rightarrow\!+\,\mathsf{let}$, which is a more faithful translation (in terms of preserving the structure). One might expect to be able to prove termination of cut-elimination by proving termination

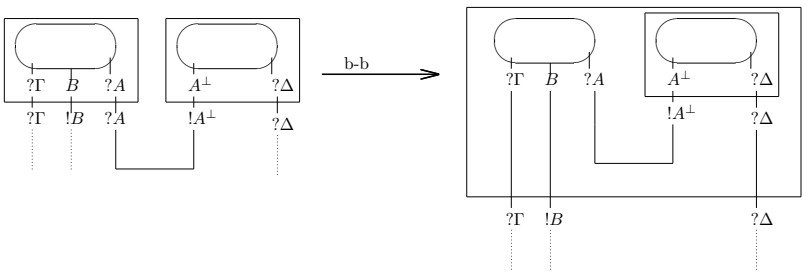

Fig. 24. The b-b rule

of $\longrightarrow_{\mathrm{let}}$ in $\lambda\!\rightarrow + \mathsf{let}$. However, the $\longrightarrow_{\mathrm{let}}$-reduction rules are quite complicated and do not lend themselves very well for a detailed analysis. It would be interesting to see whether we can prove SN for the $\longrightarrow_{\mathrm{let}}$-reduction rules (or an appropriate restriction or a variant of them) and conclude termination of cut-elimination from that.

Another interesting issue is to study the flexibility of the notion of deduction graph. Does it generalise easily to full propositional logic and predicate logic? And what if we allow e.g. overlapping boxes or if we use a variant of natural deduction with a sequent notation? Preliminary studies show that deduction graphs extend to full propositional logic easily.

The Curry-Howard isomorphism reveals a deep connection between deduction and computation, where the notions of cut-elimination and β-reduction are the central issues. We have studied the notion of cut-elimination for deduction graphs here. From there on there are many potentially interesting links to the research fields of term-graph reduction (for implementing functional programs), optimal λ-reduction (making optimal use of sharing) and interaction nets (that provide a very basic graphical computational model).

Finally, deductions are static objects, that can be (proof) checked for correctness. The process of constructing a proof (deduction) deals with unfinished deductions that are built up in a mix of forward and backward (goal-directed) steps. It would be interesting to understand the notion of 'open' (unfinished) deduction graph.

Acknowledgments

We would like to thank Delia Kesner and Stephane Lengrand for discussions about the subject of the last section. We also thank the anonymous referees for their comments.

References

A. Asperti and S. Guerrini, *The Optimal Implementation of Functional Programming Languages*, volume 45 of em Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998.

R. Di Cosmo and D. Kesner, Strong normalization of explicit substitutions via cut-elimination in proof nets, In *Twelfth Annual IEEE Symposium on Logic in Computer Science (LICS)*, 35-46, IEEE Computer Society Press, 1997.

F. B. Fitch, *Symbolic Logic*, Ronald Press Company, New York, 1952.

G. Gentzen. *Collected Works*. M.E. Szabo (ed.). North-Holland, 1969.

H. Geuvers and R. Nederpelt, Rewriting for Fitch style natural deductions, Proceedings of RTA 2004, ed. V. van Oostrom, LNCS 391, pp. 134-154, Springer.

J.-Y. Girard, Linear Logic, *Theoretical Computer Science*, 50(1):1-101, 1987.

D. Kesner and S. Lengrand, Extending the Explicit Substitution Paradigm, in *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA)* LNCS 3467, pages 407-422, Nara, Japan, April 2005.

Y. Lafont. Interaction nets. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages (POPL'90)*, pp. 95–108. ACM Press, 1990.

R. Milner, Axioms for bigraphical structure, Technical Report UCAM-CL-TR-581, Computer Laboratory, University of Cambridge UK, 2004.

D. Prawitz, *Natural Deduction*, Almquist & Wiksell, Stockholm, 1965.

D. Prawitz, *Ideas and results in Proof Theory*, Proc. of the second Scandinavian Logic Symposium, ed. J.E. Fenstad, pp. 235-307, North-Holland, 1971.

Appendix A. Subject Reduction

Lemma 5.13 [Subject Reduction] If $\Gamma;\Delta \vdash C$ and $C \twoheadrightarrow_{\mathrm{let}} D$, then $\Gamma;\Delta^D \vdash D$. If $\Gamma \vdash M : A$ and $M \twoheadrightarrow_{\mathrm{let}} N$, then $\Gamma \vdash N : A$.

To prove the SR property, it is convenient to change the typing rules a little, giving also a type to the subexpresion $C[y]$ (which is not well-typed in $\lambda\!\!\rightarrow + \mathsf{let}$). Thus, we change the λ-rule into the following two rules: the *fill-rule* and the *abs-rule*.

$$\frac{\Gamma;\Delta \vdash C}{\Gamma \vdash C[y] : B}\ \text{fill-rule}\quad \text{if } (y{:}B) \in \Delta \qquad \frac{\Gamma, x{:}A \vdash t : B}{\Gamma \vdash \lambda x{:}A.t : A{\rightarrow}B}\ \text{abs-rule}\quad \text{if } (t = \mathsf{let}\ \ldots)$$

If we call the new systems $\lambda\!\!\rightarrow + \mathsf{let}^+$ and denote derivability in this system by $\vdash^+$, we immediately see that

$$\begin{aligned} \Gamma;\Delta \vdash^+ C &\ \Leftrightarrow\ \Gamma;\Delta \vdash C \\ \Gamma \vdash^+ t : A &\ \Leftarrow\ \Gamma \vdash t : A \\ \Gamma \vdash^+ t : A \wedge t \neq \mathsf{let}\ \ldots &\ \Rightarrow\ \Gamma \vdash t : A \end{aligned}$$

We will now prove SR for $\lambda\!\!\rightarrow + \mathsf{let}^+$ and we conclude SR for $\lambda\!\!\rightarrow + \mathsf{let}$ by observing that, if $\Gamma \vdash t : A$ and $t \twoheadrightarrow_{\mathrm{let}} q$, then $q \neq \mathsf{let}\ \ldots$.

SR for $\lambda\!\!\rightarrow + \mathsf{let}^+$ By induction on the structure of the expression, distinguishing cases according to the reduction rule. We use some additional meta-theoretic properties of $\lambda\!\!\rightarrow + \mathsf{let}^+$ (apart from the ones already listed for $\lambda\!\!\rightarrow + \mathsf{let}$ in Section 5, which also hold for $\lambda\!\!\rightarrow + \mathsf{let}^+$). The most important ones are

— Weakening: if $\Gamma;\Delta \vdash^+ C$ and $\Gamma \subseteq \Gamma'$, then $\Gamma';\Delta \vdash^+ C$ and similarly for terms.
— Strengthening: if $\Gamma;\Delta \vdash^+ C$, then $\Gamma\lceil \mathrm{FV}(C);\Delta \vdash^+ C$ and similarly for terms.
— Generation: a well-formed context or a well-typed term can only have been created in one way (see the derivation rules).

For the proof of SR, there are 4 base cases, where $\twoheadrightarrow_B$, $\twoheadrightarrow_{CM}$, $\twoheadrightarrow_{CP}$ or $\twoheadrightarrow_L$ is applied to a context on the "top level". Then there are 4 contextual closure cases:

1 $\lambda x{:}A.P \twoheadrightarrow \lambda x{:}A.P'$, because $P \twoheadrightarrow P'$; this case is easy.
2 $\mathsf{let}\, x = P\, \mathsf{in}\, C \twoheadrightarrow \mathsf{let}\, x = P'\, \mathsf{in}\, C$, because $P \twoheadrightarrow P'$; this case is also easy.
3 $\mathsf{let}\, x = P\, \mathsf{in}\, C \twoheadrightarrow \mathsf{let}\, x = P\, \mathsf{in}\, C'$, because $C \twoheadrightarrow C'$; this case is also easy.
4 $C[v] \twoheadrightarrow D[w]$. This can be caused by a reduction in $C[v]$ on the "top level": this is the interesting case, to be treated below. If $C[v] = \mathsf{let}\, x_1 = P_1 \ldots \mathsf{let}\, x_n = P_n\, \mathsf{in}\, v \twoheadrightarrow D[w]$ because $P_i \twoheadrightarrow P_i'$, we easily conclude by induction.

We first consider the base cases where $\twoheadrightarrow_B$ or $\twoheadrightarrow_{CM}$ is applied to a context on the "top level". Below we use $\Delta_{x,y}$ to denote Δ with the declarations of x and y removed.

For the $\twoheadrightarrow_B$ case:

$$\mathsf{let}\, x = \lambda z{:}A.D[y]\, \mathsf{in}\, \mathsf{let}\, p = xq\, \mathsf{in}\, C \twoheadrightarrow_B D[z := q][\mathsf{let}\, p = y\, \mathsf{in}\, C],$$

where $x \notin \mathrm{FV}(C)$, we find that

1 $\Gamma, z{:}A; \Delta^D \vdash^+ D$ and $q{:}A \in \Gamma$ and $y{:}E \in \Delta^D$, therefore $\Gamma; \Delta^D \vdash^+ D[z := q]$
2 $\Gamma, x{:}A{\to}E, p{:}E; \Delta_{x,p} \vdash^+ C$ and so $\Gamma, p{:}E; \Delta_{x,p} \vdash^+ C$ because $x \notin \mathrm{FV}(C)$.

From the second we conclude that $\Gamma, y{:}E, p{:}E; \Delta_{x,p} \vdash^+ C$ and thus $\Gamma, y{:}E; p{:}E, \Delta_{x,p} \vdash^+ \mathsf{let}\, p = y\, \mathsf{in}\, C$. From this, using context substitution and the first, we derive $\Gamma; \Delta^D, p{:}E, \Delta_{x,p} \vdash^+ D[z := q][\mathsf{let}\, p = y\, \mathsf{in}\, C]$.

For the $\twoheadrightarrow_{CM}$ case:

$$\mathsf{let}\, x = P\, \mathsf{in}\, \mathsf{let}\, y = \lambda z.D[q]\, \mathsf{in}\, C \twoheadrightarrow_{CM} \mathsf{let}\, y = \lambda z.(\mathsf{let}\, x = P\, \mathsf{in}\, D[q])\, \mathsf{in}\, C,$$

where $x \notin \mathrm{FV}(C)$ we find that

1 $\Gamma, x{:}A, y{:}B{\to}E; \Delta_{x,y} \vdash^+ C$ and therefore $\Gamma, y{:}B{\to}E; \Delta_{x,y} \vdash^+ C$ because $x \notin \mathrm{FV}(C)$.
2 $\Gamma, x{:}A, z{:}B; \Delta^D \vdash^+ D$ with $q : E \in \Delta^D$ and therefore $\Gamma, z{:}B, x{:}A; \Delta^D \vdash^+ D$.
3 $\Gamma \vdash^+ P : A$ and therefore $\Gamma, z{:}B \vdash^+ P : A$.

The second and third yield $\Gamma \vdash^+ \lambda z{:}B.\mathsf{let}\, x = P\, \mathsf{in}\, D[q] : B{\to}E$. So, using the first we derive $\Gamma; \Delta_x \vdash^+ \mathsf{let}\, y = \lambda z{:}B.(\mathsf{let}\, x = P\, \mathsf{in}\, D[q])\, \mathsf{in}\, C$.

For the contextual closure we only consider the most interesting case, where $C[v] \twoheadrightarrow E$ on the top level. There are four subcases:

— Suppose $C[v] \twoheadrightarrow_B D[v]$ or $C[v] \twoheadrightarrow_{CM} D[v]$. Note that in D, v is still a let-abstracted variable. Now, $C[-] \twoheadrightarrow D[-]$, so by induction hypothesis $\Gamma; \Delta^D \vdash^+ D[-]$, and $v : B$ must be in Δ^D, so $\Gamma \vdash^+ D[v] : B$.
— Suppose $C[v] \twoheadrightarrow_{CP} D[v]$. Then $C[v] = \mathsf{let}\, x = P\, \mathsf{in}\, E[y := x]$, and we can write $E[y := x]$ as $F[y := x][v]$. So $\Gamma, x{:}A; \Delta^{F[y:=x]} \vdash^+ F[y := x]$ with $v : B \in \Delta^{F[y:=x]}$ or $v = x$. We conclude that $\Gamma, y{:}A, x{:}A; \Delta^F \vdash^+ F$ and thus $\Gamma \vdash^+ \mathsf{let}\, y = P\, \mathsf{in}\, \mathsf{let}\, x = P\, \mathsf{in}\, E$.
— Suppose $C[v] \twoheadrightarrow_L D[v]$. Then $C[v] = \mathsf{let}\, y = x\, \mathsf{in}\, G[v]$, and we know that $\Gamma, y{:}A \vdash^+ G[v] : B$, with $x{:}A \in \Gamma$. By substitution, we conclude $\Gamma \vdash^+ G[v][y := x] : B$ and we are done.

Appendix B. Independence of $\langle\!\langle G \rangle\!\rangle_\varphi$ of the topological ordering φ

This section constitutes the proof of Lemma 5.14. Recall that a *box-topological ordering* of a deduction graph G is a linear ordering of the nodes of G, that respects both the partial ordering imposed by the link graph and the partial ordering imposed by the place graph.

Definition B.1. Let R be a binary relation. We will write $\mathsf{CL}(R)$ for the transitive closure of R. Let S be a set and let φ and ψ be two partial orderings of S (i.e. φ and ψ are reflexive, transitive and anti-symmetric). We say that φ is *consistent* with ψ, if $\mathsf{CL}(\varphi \cup \psi)$ is again a partial ordering of S.

We give two simple properties of partial orderings

Lemma B.2.

1 If ψ is a subset of the linear ordering φ, then so is $\mathsf{CL}(\psi)$.
2 If φ is a linear ordering of the set S, then every transitive closed subset ψ of φ is a partial ordering of S.

Lemma B.3. Let G be a deduction graph. Let φ be the partial ordering imposed by the link graph and let ψ be the partial ordering imposed by the place graph. Then:

1 ζ is a box-topological ordering of $G \leftrightarrow \zeta$ is a linear extension of $\mathsf{CL}(\varphi \cup \psi)$.
2 φ is consistent with ψ.

Proof.

1 Suppose ζ is a box-topological ordering of G. Then $\varphi \cup \psi$ is a subset of ζ. Then also $\mathsf{CL}(\varphi \cup \psi)$ is a subset of ζ. Thus ζ is a linear extension of $\mathsf{CL}(\varphi \cup \psi)$
Assume ζ is a linear extension of $\mathsf{CL}(\varphi \cup \psi)$. Suppose $(a, b) \in \varphi$. Then $(a, b) \in \mathsf{CL}(\varphi \cup \psi)$. So $(a, b) \in \zeta$. Similarly: if $(a, b) \in \psi$, then $(a, b) \in \zeta$. Hence ζ is a box-topological ordering of G.
2 According to Lemma 2.11, there exists a box-topological ordering ζ of G. So ζ is also a linear extension of $\mathsf{CL}(\varphi \cup \psi)$.

From now on φ will indicate the linear ordering as well as the associated isomorphism.

Definition B.4. Let φ be a linear order of n elements. We define for every $i \leq n-2$ the interchange of $\varphi(i)$ and $\varphi(i+1)$, π_i by:

$$(\pi_i \circ \varphi)(k) \begin{cases} \varphi(k) & \text{if } k \neq i, i+1 \\ \varphi(i+1) & \text{if } k = i \\ \varphi(i) & \text{if } k = i+1 \end{cases}$$

Lemma B.5. Let G be a set. Let α be a partial ordering of G. Suppose φ and ψ are linear extensions of α. Then there exist a k and $i_0, \ldots, i_k$ such that

$$\pi_{i_k} \circ \ldots \circ \pi_{i_0} \circ \varphi = \psi$$

and:

$$\pi_{i_j} \circ \ldots \circ \pi_{i_0} \circ \varphi$$

is a linear extension of α for every $j \leq k$.

Proof. Suppose $\varphi(m) = \psi(m)$ for all $m < n$ for some $n \in \mathbb{N}$. Take $k := \varphi^{-1}(\psi(n))$. Define the orderings $\zeta_0, \zeta_1, \ldots, \zeta_{k-(n+1)}$ by:

$$\zeta_0 = \pi_{k-1} \circ \varphi$$

$$\zeta_{p+1} = \pi_{k-(p+2)} \circ \zeta_p.$$

We see by induction that:

1 For all $i \leq k-(n+1)$, ζ_i is a linear extension of α.
2 For all $i \leq k-(n+1)$, for all $m < n$, $\zeta_i(m) = \varphi(m)$.
3 For all $i \leq k-(n+1)$, $\zeta_i(k-(i+1)) = \psi(n)$.

Hence $\zeta_{k-(n+1)}(m) = \psi(m)$ for all $m < n+1$. Thus the statement follows with induction.

Example B.6. Consider Figure 25.

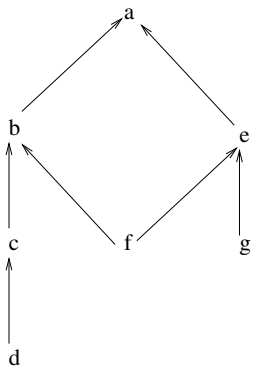

Fig. 25.

Let $\varphi := \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & b & c & d & e & f & g \end{pmatrix}$ and $\psi := \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & e & g & b & f & c & d \end{pmatrix}$. We will rewrite φ in ψ by just interchanging adjacent nodes in the linear ordering and keeping a topological ordering at each stage.

— We start by placing $\psi(1) = e$ at the right position. We see that:

$$\pi_1 \circ \pi_2 \circ \pi_3 \circ \varphi = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & e & b & c & d & f & g \end{pmatrix}$$

— We continue by placing $\psi(2) = g$ at the right position:

$$\pi_2 \circ \pi_3 \circ \pi_4 \circ \pi_5 \circ \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & e & b & c & d & f & g \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & e & g & b & c & d & f \end{pmatrix}$$

— Now, we place $\psi(4) = f$ at the right position:

$$\pi_4 \circ \pi_5 \circ \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & e & g & b & c & d & f \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ a & e & g & b & f & c & d \end{pmatrix} = \psi$$

Thus, we have found that:

$$\pi_4 \circ \pi_5 \circ \pi_2 \circ \pi_3 \circ \pi_4 \circ \pi_5 \circ \pi_1 \circ \pi_2 \circ \pi_3 \circ \varphi = \psi$$

Definition B.7. If φ is a linear ordering of a finite set S and $T \subseteq S$, then $\varphi_{|T}$ is the relation φ restricted to T.

So we will write $\varphi_{|T}$ for the restricted relation φ as well as for the isomorphism associated with it.

Definition B.8. Let G be a deduction graph. Let φ be a box-topological ordering of G. We will write $\widehat{\varphi}$ for the restriction of φ to top-level nodes and $\varphi_{\overline{\mathcal{B}}}$ for the restriction of φ to the nodes that are in $\overline{\mathcal{B}}$ and that are not in deeper boxes.

Observe that $\varphi_{\overline{\mathcal{B}}} \subseteq \varphi_{|\overline{\mathcal{B}}}$.

Lemma B.9. Let G be a deduction graph and let φ and ψ be box-topological orderings of G. If $\widehat{\varphi} = \widehat{\psi}$ and $\varphi_{\overline{\mathcal{B}}} = \psi_{\overline{\mathcal{B}}}$ for every box $\mathcal{B}$, then

$$\langle\!\langle G \rangle\!\rangle_{\varphi} = \langle\!\langle G \rangle\!\rangle_{\psi}.$$

Proof. The proof is with induction to the depth of G and the number of top-level nodes.

Lemma B.10. If G is a deduction graph and φ and $\pi_i \circ \varphi$ are both box-topological orderings of G for a certain i, then

$$\langle\!\langle G \rangle\!\rangle_{\varphi} \simeq \langle\!\langle G \rangle\!\rangle_{\pi_i \circ \varphi}$$

Proof. Note that if $\widehat{\varphi} = \widehat{\pi_i \circ \varphi}$ and $\varphi_{\overline{\mathcal{B}}} = (\pi_i \circ \varphi)_{\overline{\mathcal{B}}}$ for every box $\mathcal{B}$, then it follows by Lemma B.9. Furthermore, if $\widehat{\varphi} \neq \widehat{\pi_i \circ \varphi}$, then there exists a j such that $\widehat{\pi_i \circ \varphi} = \pi_j \circ \widehat{\varphi}$. Similarly, if there exists a box $\mathcal{B}$ such that $\varphi_{\overline{\mathcal{B}}} \neq (\pi_i \circ \varphi)_{\overline{\mathcal{B}}}$, then there exists a j such that $(\pi_i \circ \varphi)_{\overline{\mathcal{B}}} = \pi_j \circ \varphi_{\overline{\mathcal{B}}}$.

Suppose $\widehat{\varphi} \neq \widehat{\pi_i \circ \varphi}$ and $\langle\!\langle \overline{\mathcal{B}} \rangle\!\rangle_{\varphi_{|\overline{\mathcal{B}}}} \simeq \langle\!\langle \overline{\mathcal{B}} \rangle\!\rangle_{\psi_{|\overline{\mathcal{B}}}}$ for all boxes $\mathcal{B}$. Determine j such that $\widehat{\pi_i \circ \varphi} = \pi_j \circ \widehat{\varphi}$. Suppose neither of the nodes $\varphi(i)$ and $\varphi(i+1)$ is an A-node. Suppose $\widehat{\varphi}(m)$ is the φ-maximal element of G. Then there exist a context $R[-]$ and terms P, P', Q, Q' such that

$$\begin{aligned}
\langle\!\langle G \rangle\!\rangle_{\varphi} &\simeq \langle\!\langle G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j+2) \rangle\!\rangle_{\varphi_{|G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j+2)}} [R[-]] \\
&\simeq \langle\!\langle G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j+1) \rangle\!\rangle_{\varphi_{|G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(i+1)}} [\mathsf{let}\, x_{\widehat{\varphi}(j+1)} = P \,\mathsf{in}\, R[-]] \\
&\simeq \langle\!\langle G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j) \rangle\!\rangle_{\varphi_{|G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j)}} [\mathsf{let}\, x_{\widehat{\varphi}(j)} = Q \,\mathsf{in}\, \mathsf{let}\, x_{\widehat{\varphi}(j+1)} = P \,\mathsf{in}\, R[-]]
\end{aligned}$$

and:

$$\begin{aligned}
\langle\!\langle G \rangle\!\rangle_{\pi \circ \varphi} &\simeq \langle\!\langle G \setminus \widehat{\varphi}(m), \ldots, \overline{\varphi}(j+2) \rangle\!\rangle_{\varphi_{|G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j+2)}} [R[-]] \\
&\simeq \langle\!\langle G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j+2), \widehat{\varphi}(j) \rangle\!\rangle_{\varphi_{|G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j+2), \widehat{\varphi}(j)}} [\mathsf{let}\, x_{\widehat{\varphi}(j)} = Q' \,\mathsf{in}\, R[-]] \\
&\simeq \langle\!\langle G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j) \rangle\!\rangle_{\varphi_{|G \setminus \widehat{\varphi}(m), \ldots, \widehat{\varphi}(j)}} [\mathsf{let}\, x_{\widehat{\varphi}(j+1)} = P' \,\mathsf{in}\, \mathsf{let}\, x_{\widehat{\varphi}(j)} = Q' \,\mathsf{in}\, R[-]].
\end{aligned}$$

After examination of the cases, we see that $P = P'$, $Q = Q'$ and $x_{\widehat{\varphi}(j)} \notin Q$ and $x_{\widehat{\varphi}(j+1)} \notin P$. Hence $\langle\!\langle G \rangle\!\rangle_{\varphi} \simeq \langle\!\langle G \rangle\!\rangle_{\pi_i \circ \varphi}$. If $\varphi(i)$ or $\varphi(i+1)$ is an A-node, we can make a similar computation. The statement follows with induction to the depth of G.

We can now prove Lemma 5.14.

Lemma B.11. If G is a deduction graph and φ and ψ are both box-topological orderings of G, then

$$\langle\!\langle G \rangle\!\rangle_{\varphi} \simeq \langle\!\langle G \rangle\!\rangle_{\psi}$$

Proof. Immediate with Lemma B.3, Lemma B.5 and Lemma B.10.

Appendix C. Reduction in $\lambda\!\rightarrow\! + \mathsf{let}$ reflects reduction on deduction graphs

This section constitutes the proof of Lemma 5.15.

Lemma C.1 (Gluing Lemma). Let G and F be deduction graphs, such that for all free nodes (n, A) of F there exists a node (n, A) in G on top-level. Let $H = G \cup F$ (so the free nodes of F and the corresponding nodes of G have been identified.) Then

$$\langle\!\langle H \rangle\!\rangle = \langle\!\langle G \rangle\!\rangle[\langle\!\langle F \rangle\!\rangle].$$

Proof. The proof is by induction on the number of non-A-nodes of F. Suppose F has only A-nodes. Then

$$\langle\!\langle H \rangle\!\rangle = \langle\!\langle G \rangle\!\rangle = \langle\!\langle G \rangle\!\rangle[-] = \langle\!\langle G \rangle\!\rangle[\langle\!\langle F \rangle\!\rangle].$$

Suppose F has $k+1$ non-A-nodes. Then we can choose the maximal node of H to be one of them, say r. Then

$$\begin{array}{rcl} \langle\!\langle H \rangle\!\rangle & = & \langle\!\langle H \setminus r \rangle\!\rangle[\mathsf{let}\, x_r = P\, \mathsf{in}\, [-]] \\ & = & (\langle\!\langle G \rangle\!\rangle[\langle\!\langle F \setminus r \rangle\!\rangle])[\mathsf{let}\, x_r = P\, \mathsf{in}\, [-]] \\ & = & \langle\!\langle G \rangle\!\rangle([\langle\!\langle F \setminus r \rangle\!\rangle][\mathsf{let}\, x_r = P\, \mathsf{in}\, [-]]) \\ & = & \langle\!\langle G \rangle\!\rangle[\langle\!\langle F \rangle\!\rangle] \end{array}$$

Lemma C.2. Let G be a deduction graph with free node (m, A). Let G' be G with (m, A) replaced by (n, A). Then

$$\langle\!\langle G' \rangle\!\rangle = \langle\!\langle G \rangle\!\rangle[x_m := x_n].$$

Proof. The proof is by induction on the construction of $\langle\!\langle G \rangle\!\rangle$.

Lemma C.3. If G' is obtained from G by eliminating a safe cut, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_B \langle\!\langle G' \rangle\!\rangle$.

Proof. Suppose p is top-level and maximal. We will proceed by cutting both G and G' in three pieces. The pieces of G are:

— $\langle \{p, n, m\}, \{(p, n), (p, m)\} \rangle$;
— box $\mathcal{B}$ with the nodes that are reachable from within $\mathcal{B}$ in one step;
— $G \setminus \mathcal{B}, p$.

The pieces of G' are:

— $\langle \{p, j\}, \{(p, j)\} \rangle$;
— $\mathcal{B}^k$ with k replaced by m;
— $G \setminus \mathcal{B}, p$.

Using the Lemmas C.1 and C.2, we see that:

$$\langle\!\langle G \rangle\!\rangle = \langle\!\langle G \setminus \mathcal{B}, p \rangle\!\rangle[\mathsf{let}\, x_n = (\lambda x_k{:}A.\langle\!\langle B^k \rangle\!\rangle[x_j])\, \mathsf{in}\, \mathsf{let}\, x_p = (x_n x_m)\, \mathsf{in}\, [-]]$$

and

$$\langle\!\langle G' \rangle\!\rangle = \langle\!\langle G \setminus \mathcal{B}, p \rangle\!\rangle[\mathcal{B}^k[x_k := x_m][\mathsf{let}\, x_p = x_j\, \mathsf{in}\, [-]].$$

Thus $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_B \langle\!\langle G' \rangle\!\rangle$. The other cases follow by induction. Here we make use of the facts that n and p are at the same level and that there exists just one edge to n.

Lemma C.4. If G' is obtained from G by an incorporation step, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CM} \langle\!\langle G' \rangle\!\rangle$.

Proof. Suppose n is a top-level node. Suppose box $\mathcal{B}_q$ with box-node q that contains p is a maximal element of G. Then

$$\begin{array}{rcl} \langle\!\langle G \rangle\!\rangle & = & \langle\!\langle G \setminus q \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.\langle\!\langle \mathcal{B}_q^k \rangle\!\rangle[y])\, \mathsf{in}\, [-]] \\ & = & \langle\!\langle G \setminus q, n \rangle\!\rangle[\mathsf{let}\, x_n = (\lambda x_l.\langle\!\langle \mathcal{B}_n^l \rangle\!\rangle[x_j])\, \mathsf{in}\, \mathsf{let}\, x_q = (\lambda x_k.\langle\!\langle \mathcal{B}_q^k \rangle\!\rangle[y])\, \mathsf{in}\, [-]] \end{array}$$

The deduction graph G' contains now a box $\mathcal{C}_q$ that incorporates the box $\mathcal{B}_n$. To calculate $\langle\!\langle G' \rangle\!\rangle$, we divide $\mathcal{C}_q^k$ into two pieces:

— $\mathcal{B}_n$ together with all free nodes of $\mathcal{C}_q^k$;
— $\mathcal{C}_q^k \setminus \mathcal{B}_n$ (but with a free node n).

Now with the help of Lemma C.1 we see that:

$$\begin{array}{rcl} \langle\!\langle G' \rangle\!\rangle & = & \langle\!\langle G' \setminus q \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.\langle\!\langle \mathcal{C}_q^k \rangle\!\rangle[y])\, \mathsf{in}\, [-]] \\ & = & \langle\!\langle G' \setminus q \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.(\mathsf{let}\, x_n = (\lambda x_l.\langle\!\langle \mathcal{B}_n^l \rangle\!\rangle[x_j])\, \mathsf{in}\, [-])[\langle\!\langle \mathcal{C}_q^k \setminus \mathcal{B}_n \rangle\!\rangle[y])\, \mathsf{in}\, [-]] \\ & = & \langle\!\langle G \setminus q, n \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.(\mathsf{let}\, x_n = (\lambda x_l.\langle\!\langle \mathcal{B}_n^l \rangle\!\rangle[x_j])\, \mathsf{in}\, \langle\!\langle \mathcal{B}_q^k \rangle\!\rangle[y])\, \mathsf{in}\, [-]]. \end{array}$$

So $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CM} \langle\!\langle G' \rangle\!\rangle$. The other cases follow by induction.

Lemma C.5. If G' is obtained from G by an unsharing step, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CP} \langle\!\langle G' \rangle\!\rangle$.

Proof. Suppose box $\mathcal{B}$ with box-node n is at top-level. We will cut G' in four parts:

— H', i.e. all nodes from which n or n' is reachable, all boxes that contain these nodes, plus the nodes that can be reached from these in one step;
— box $\mathcal{B}$ with the nodes that are reachable from within $\mathcal{B}$ in one step, plus the free nodes of H';
— box $\mathcal{B}'$ with the nodes that are reachable from within $\mathcal{B}'$ in one step, plus the free nodes of $H' \cup \mathcal{B}$;
— $(G' \setminus H, \mathcal{B}', \mathcal{B})$, together with all free nodes of the previous $H' \cup \mathcal{B}' \cup \mathcal{B}$.

With Lemma C.1 we see that:

$$\langle\!\langle G' \rangle\!\rangle = \langle\!\langle G' \setminus H, \mathcal{B}', \mathcal{B} \rangle\!\rangle[\mathsf{let}\, x_{n'} = (\lambda x_m.\langle\!\langle \mathcal{B}^m \rangle\!\rangle[x_{j'}])\, \mathsf{in}\, \mathsf{let}\, x_n = (\lambda x_m.\langle\!\langle \mathcal{B}^m \rangle\!\rangle[x_j])\, \mathsf{in}\, \langle\!\langle H' \rangle\!\rangle]$$

We cut G in three parts as follows:

— H, i.e. all nodes from which n is reachable, all boxes that contain these nodes, plus the nodes that can be reached from these within one step;
— box $\mathcal{B}$ with the nodes that are reachable from within $\mathcal{B}$ in one step, plus the free nodes of H;
— $(G \setminus H, \mathcal{B})$, together with the free nodes of $H \cup \mathcal{B}$.

With Lemmas C.1 and C.2 we see that:

$$\begin{array}{rcl} \langle\!\langle G \rangle\!\rangle & = & \langle\!\langle G \setminus H, \mathcal{B} \rangle\!\rangle[\mathsf{let}\, x_n = (\lambda x_m.\langle\!\langle \mathcal{B}^m \rangle\!\rangle[x_j])\, \mathsf{in}\, \langle\!\langle H \rangle\!\rangle] \\ & = & \langle\!\langle G' \setminus H, \mathcal{B}', \mathcal{B} \rangle\!\rangle[\mathsf{let}\, x_n = (\lambda x_m.\langle\!\langle \mathcal{B}^m \rangle\!\rangle[x_j])\, \mathsf{in}\, \langle\!\langle H' \rangle\!\rangle[x_{n'} := x_n]] \end{array}$$

So $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CP} \langle\!\langle G' \rangle\!\rangle$. The other case follows by induction.

Lemma C.6. If G' is obtained from G by a repeat-elimination step, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_L \langle\!\langle G' \rangle\!\rangle$.

Proof. Suppose n_0 and n_1 are top-level nodes. By cutting G in three parts and G' in either two or three, we easily see that $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_L \langle\!\langle G' \rangle\!\rangle$. Now assume that n_1 is in a box. Using that none of the nodes to n_1 is an I-node, we can make another deduction graph, G'' as follows. Place a node n_1' at top level. Replace the edges $n_1 \rightarrowtriangle n_0$ by $n_1' \rightarrowtriangle n_0$ and redirect all edges to n_1 to n_1'. We can now remove n_1. We have that G' can be obtained from G'' by a repeat-elimination step, and because both n_0 and n_1' are at top-level, $\langle\!\langle G'' \rangle\!\rangle \twoheadrightarrow_L \langle\!\langle G' \rangle\!\rangle$. G, on the other hand, can be obtained from G'' by an incorporation step (this time on a single node, instead of on a box), and a renaming. Thus $\langle\!\langle G'' \rangle\!\rangle$, $\langle\!\langle G' \rangle\!\rangle$ and $\langle\!\langle G \rangle\!\rangle$ have the following form, where $x_j \neq x_{n_1'}$:

$$\langle\!\langle G'' \rangle\!\rangle = \langle\!\langle G'' \setminus n_1', \mathcal{B} \rangle\!\rangle[\mathsf{let}\, x_{n_1'} = x_{n_0}\, \mathsf{in}\, \mathsf{let}\, x_r = (\lambda x_k.\langle\!\langle \mathcal{B}^k \rangle\!\rangle[x_j])\, \mathsf{in}\, [-]]$$

$$\langle\!\langle G' \rangle\!\rangle = \langle\!\langle G'' \setminus n_1', \mathcal{B} \rangle\!\rangle[\mathsf{let}\, x_r = (\lambda x_k.\langle\!\langle \mathcal{B}^k \rangle\!\rangle[x_j])\, \mathsf{in}\, [-]][x_{n_1'} := x_{n_0}]$$

$$\langle\!\langle G \rangle\!\rangle = \langle\!\langle G'' \setminus n_1', \mathcal{B} \rangle\!\rangle[\mathsf{let}\, x_r = (\lambda x_k.\mathsf{let}\, x_{n_1'} = x_{n_0}\, \mathsf{in}\, \langle\!\langle \mathcal{B}^k \rangle\!\rangle)\, \mathsf{in}\, [-]]$$

So indeed $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_L \langle\!\langle G' \rangle\!\rangle$. Other cases follow by induction.