Math. Struct. in Comp. Science (2007), *vol.* 17, *pp.* 485–526. © 2007 Cambridge University Press
doi:10.1017/S0960129507006123 Printed in the United Kingdom

Natural deduction via graphs: formal definition and computation rules

HERMAN GEUVERS and IRIS LOEB

Radboud University Nijmegen,
Institute for Computing and Information Sciences,
P.O. Box 9010, 6500 GL Nijmegen,
The Netherlands
Email: {H.Geuvers;I.Loeb}@cs.ru.nl

Received 1 September 2005; revised 11 August 2006

In this paper, we introduce the formalism of *deduction graphs* as a generalisation of both Gentzen–Prawitz style natural deduction and Fitch style flag deduction. The advantage of this formalism is that, as with flag deductions (but not natural deduction), subproofs can be shared, but 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. We give a precise definition of deduction graphs, together with some illustrative examples. 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, so we 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 offer a graph-like presentation of natural deduction, and we point out some similarities between the two formalisms.

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 clear, 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 one 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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

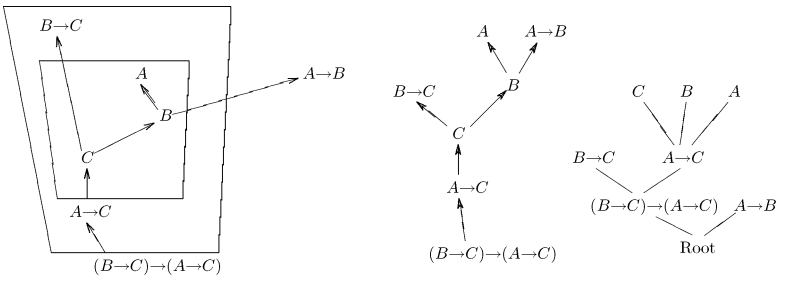


Fig. 1. Deduction graph of $(B{\to}C){\to}(A{\to}C)$ from $A{\to}B$ and link and place graphs.

of a specific shape that are intended to capture just the logical structure of a deduction without the bureaucratic details.

In deduction graphs, the nodes have formulas as labels and if node n with label A (which will be denoted (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 want to keep it simple. A node with no outgoing edges is a (local) hypothesis. Obviously, not all graphs are deductions graphs: we have to impose some restrictions. First, we should avoid cyclicity, as 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 hypotheses 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, but we want more: the deduction graphs should also in their fine structure correspond closely to the well-known Gentzen–Prawitz style deduction and Fitch style deduction. We achieve this by making sure that these well-known natural deductions are an (almost immediate) 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 the *discharging* of hypotheses, which is done, for example, in the $\to$ 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: for example, when the $\to$-introduction rule is applied, the scope ends. In deduction graphs, we formalise scope using *boxes*. A box is a subgraph (that is, a collection of nodes and the edges between them) that again forms a node itself. These boxes are typically used for the $\to$-introduction rule: the hypotheses A (to be 'discharged') are put within the box (so they are 'hidden' from the rest of the graph). To make all this a bit more intuitive, Figure 1 shows an example of a deduction graph, which can be seen as a derivation of $(B{\to}C){\to}(A{\to}C)$ from $A{\to}B$.

This is slightly reminiscent of Milner's bigraphs (Milner 2004). In these 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 also be applied to

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

our deduction graphs. 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, and 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 into 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. This was generalised to Fitch deduction in Geuvers and Nederpelt (2004): 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 more closely to typed terms in a λ-calculus with lets (but that was not made precise in Geuvers and Nederpelt (2004)).

In the present paper, as well as introducing and motivating the notion of deduction graphs (Section 2), the main topic is the study of the computational content of deduction graphs, notably that 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, for example, an unsharing of a part of the deduction graph is required. Cut-elimination is precisely defined and analysed by studying translations to other 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 $G \to_{\mathrm{cut}} G'$ 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 cannot 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 (that is, 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 'at that point'. Moreover, if $G \to_{\mathrm{cut}} G'$, then $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{\mathrm{let}}^{+} \langle\!\langle G' \rangle\!\rangle$, where the reduction steps code the changes in the deduction graph more directly.

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 from our deduction graphs. Nevertheless, the reduction rules look very similar. We discuss some future work on studying the connection between these formalisms in Section 6.1, and suggest some other ideas for future research in Section 6.2.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

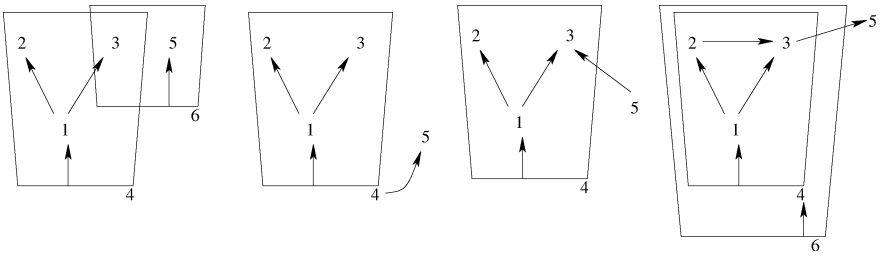


Fig. 2. Three non-examples and one example of a closed box directed graph.

2. Deduction Graphs

Definition 2.1. A *closed box directed graph* is a triple $\langle X, G, (\mathscr{B}_i)_{i\in I}\rangle$ where X is a set of *labels*, G is a directed graph where all nodes have a label in X and $(\mathscr{B}_i)_{i\in I}$ is a collection of sets of nodes of G, which are called the *boxes*. Each box $\mathscr{B}_i$ corresponds to a node, which is called the *box node* of $\mathscr{B}_i$. Moreover, the boxes $(\mathscr{B}_i)_{i\in I}$ should satisfy the following properties:

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

Formally, the labelled 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 $\in$ for the transitive relation on nodes: $n \in m$ denotes the fact that m is the box node of some box $\mathscr{B}$ and $n \in \mathscr{B}$. For example, in the rightmost graph of Figure 2, we have $1 \in 6$ and $4 \in 6$, but $5 \notin 6$ and $6 \notin 6$. Following Milner's idea, we also 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 labelled 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 labelled 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'.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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

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

In Figure 2 we give three examples that are not closed box directed graphs and one example that is.

Definition 2.5. Let G be a closed box directed graph. A *box-topological ordering* of G is a linear ordering of the nodes of G that both contains the link graph of G and its place graph.

When talking about graphs, we sometimes explicitly write the label with the node, for example, we write $(n, A) \longrightarrow\!\triangleright\, (m, B)$ to denote the edge from n with label A to m with label B. We will also sometimes 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 in three different ways, and for clarity we have used different arrow symbols for each of them:

— We use $\longrightarrow\!\triangleright$ to denote an edge.
— We use $\rightarrow$ to denote an implication.
— We use $\longrightarrow\!>$ to denote a reduction.

In a directed graph with boxes, we want to define which nodes are 'visible' and can thus 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 $\langle G, (\mathscr{B}_i)_{i\in I}\rangle$, be a closed-box directed graph, and n_0 and n_1 be nodes in this graph.

— Node n_1 is *in the scope of* n_0 if n_0 is in all boxes that n_1 is in. In a formula: $\forall i \in I(n_1 \in \mathscr{B}_i \Rightarrow n_0 \in \mathscr{B}_i)$. (So the nodes in the scope of n_0 are the nodes that are in 'wider' boxes.)
— The nodes n_0 and n_1 are *at the same depth*, when n_0 is in the scope of n_1, and n_1 is in the scope of n_0. Node n_0 is *at a greater depth* than n_1, when n_1 is in the scope of n_0, but n_0 is not in the scope of n_1.
— Node n_1 is a *top-level node* if n_1 is not contained in any box.
— The *free nodes* are the top-level nodes that have no outgoing edges.
— The *free nodes of a box* $\mathscr{B}$ are the nodes in $\mathscr{B}$ that become free nodes if we remove $\mathscr{B}$ and all boxes that contain $\mathscr{B}$.

We now define the deduction graphs that correspond to minimal proposition logic, and 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.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

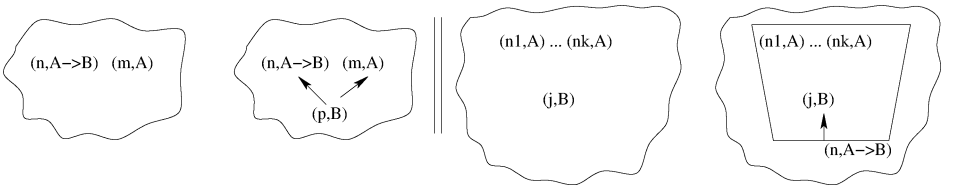


Fig. 3. The →-E and →-I rule of deduction graphs.

Definition 2.7. The collection of *deduction graphs for minimal proposition logic* is the set of closed box directed graphs over $\mathbb{N} \times \mathsf{Form}$ defined inductively 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 the left-hand part of Figure 3.

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

— a box $\mathscr{B}$ 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 *provided it is a well-formed closed box directed graph* (see the right-hand part Figure 3; the nodes $(n_1, A), \ldots, (n_k, A)$ are *discharged*).

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.

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

Example 2.8. We give examples of three simple deduction graphs in Figure 4. II is an example where no assumption is discharged in the introduction of the →. In III 'garbage' (nodes 2 and 3) occurs inside a box – it is just an involved way of proving $B{\to}B$. We have added III to show a slightly non-standard example.

Example 2.9. Figure 5 provides another example of a deduction graph, where the part H is some unspecified part of the graph that contains a node $(7, A)$. This deduction graph

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

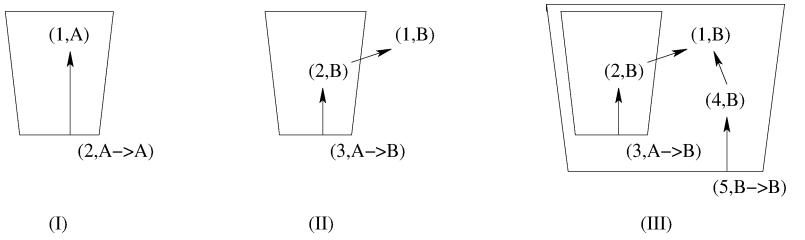


Fig. 4. Three simple deduction graphs.

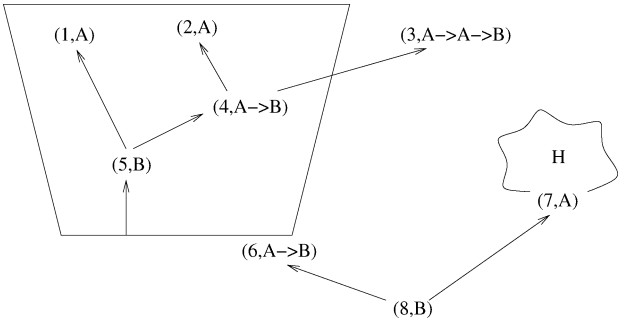


Fig. 5. Example of a deduction graph.

can be seen as a derivation of B from an assumption $A \to A \to B$ (assuming a derivation H of A).

Lemma 2.10. 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 $m > n$. □

Example 2.11. In the deduction graph of Example 2.9 (Figure 5), if we ignore the nodes of H, which are not shown, 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, which are defined in the Lemma). These types correspond to the Axiom, Join, →-I and →-E construction cases of Definition 2.7. The lemma also enables us to define a function inductively on nodes of a deduction graph, independently of how the graph has been created.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

Lemma 2.12. *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 $(m, A{\to}B)$ and one to a node (p, A), both within the scope of *n*.
   - I *n* is a box-node of a box $\mathscr{B}$ with label $A{\to}B$ and has exactly one outgoing edge, which is to a node (j, B) inside the box $\mathscr{B}$ 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.

$\Rightarrow$ The proof is 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 $>$-ordering, 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 by putting $n > m$ for every $n \in G_1, m \in G_2$.

$\Leftarrow$ The proof is 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 with respect to $>$. 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 the 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 2.13. Let *G* be a deduction graph.

1 For *n* a node of *G*, we define G_n, the *subgraph of G generated from n*, as the subgraph of *G* consisting of all nodes reachable from *n* (with boxes inherited from *G*).
2 For $\mathscr{B}$ a box in *G* with discharged nodes $(n_1, A), \ldots, (n_k, A)$, and (m, A) a fresh node (that is, not in $\mathscr{B}$), we define $\mathscr{B}^m$, the *m-closure of* $\mathscr{B}$, as the graph consisting of all nodes inside $\mathscr{B}$ plus all nodes that can be reached from within $\mathscr{B}$ 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 2.12.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press



Fig. 6. From natural deductions to deduction graphs

Corollary 2.14.

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

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

Definition 2.15 (Natural deductions as deduction graphs). Given a natural deduction Σ with conclusion B from assumptions Γ, we define a deduction graph $\bar{\Sigma}$ with a top-level node with label B and free nodes with labels Γ as follows.

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

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

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

In the $\rightarrow$ introduction rule, where $A \rightarrow B$ 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-level node (j, B). We now create a box $\mathscr{B}$, consisting of $\bar{\Sigma}$, and for all the top-level nodes (p, C) of $\bar{\Sigma}$ that correspond to occurrences of hypotheses C that are not discharged at the introduction of $A \rightarrow B$, we add a new node (p', C) outside $\mathscr{B}$ and a repeat edge from (p, C) to (p', C). This is shown in Figure 6.

The following lemma is now immediate.

Lemma 2.16. If Σ is a natural deduction with conclusion B from assumptions Γ, $\bar{\Sigma}$ is a deduction graph with a top-level node with label B and free nodes with labels Γ.

For the translation from Fitch deductions to deduction graphs, we have first to make precise what a Fitch deduction is exactly. We follow the definition given in Geuvers and Nederpelt (2004), which we do not repeat here. Instead, we restrict ourselves to giving a clarifying example – see deduction Σ_1 on the left-hand side of Figure 7. This deduction proves $A \rightarrow D$ (on the last line) from the hypotheses $A \rightarrow (B \rightarrow C) \rightarrow D$ and $A \rightarrow C$, on lines 1 and 2. The open (that is, non-discharged) hypotheses are indicated by a 'flag' whose pole extends to the line of the conclusion. Discharging a hypothesis corresponds to ending the

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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

Fig. 7. Fitch deductions Σ_1, Σ_2 and Σ_2.

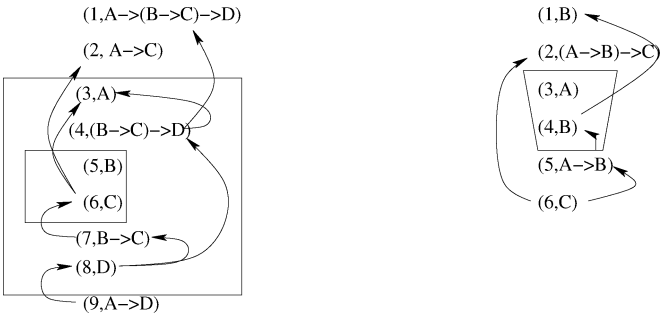


Fig. 8. From flag deductions to deduction graphs.

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 2.17 (Fitch deductions as deduction graphs). Given a Fitch deduction Σ with conclusion B and flags Γ, we define a deduction graph $\hat{\Sigma}$ with a top-level node with label B and free nodes with labels Γ as follows.

We 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' (see flag deduction Σ_2 in Figure 7). 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 8, which are translations of the Fitch deductions of Figure 7.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

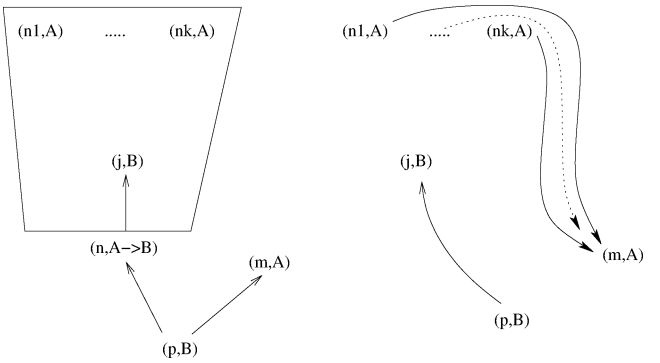


Fig. 9. Cut-elimination in deduction graphs.

Lemma 2.18. If Σ is a Fitch deduction with conclusion B and flags Γ, $\hat{\Sigma}$ is a deduction graph with a top-level node with label B and free nodes with labels Γ.

3. Cut-elimination

We now define the notion of a '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 $\rightarrow$-I immediately followed by an $\rightarrow$-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 9. However, it could also happen that the $\rightarrow$-I and the $\rightarrow$-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 $(n, A{\rightarrow}B)$
— A node (p, B)
— A node (m, A)
— A sequence of R-nodes $(s_0, A{\rightarrow}B), \ldots, (s_i, A{\rightarrow}B)$
— Edges $(p, B) \longrightarrow\!\triangleright (s_i, A{\rightarrow}B) \longrightarrow\!\triangleright \ldots \longrightarrow\!\triangleright (s_0, A{\rightarrow}B) \longrightarrow\!\triangleright (n, A{\rightarrow}B)$
— An edge $(p, B) \longrightarrow\!\triangleright (m, A)$.

The operation sketched in Figure 9 is only defined on cuts where the $\rightarrow$-I is immediately followed by the $\rightarrow$-E on the same formula. However, it may not yield a deduction graph again: when the box-node $(n, A{\rightarrow}B)$ 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 10, where three possible positions of the cut with respect to boxes are depicted. In the third case, contracting the cut yields an edge pointing into a box, so we cannot contract it. For reasons related to the correspondence with contexts, we also discard the second case. We call these cases a *depth conflict* in Definition 3.9. We will now define a 'safety' criterion telling us when cut-elimination (as in Figure 9) can be performed

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

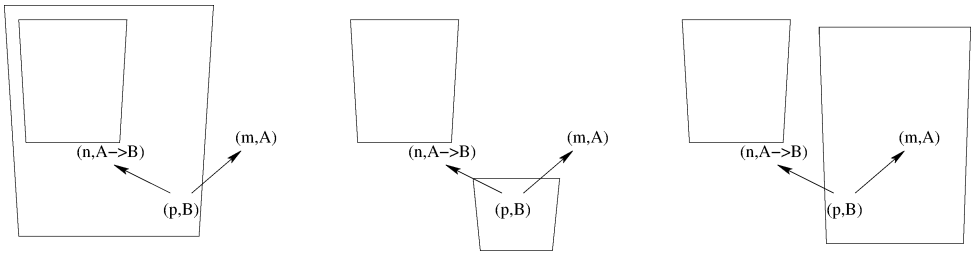


Fig. 10. Three possible positions of the cut with respect to boxes. When n and p are at the same depth (first picture) the cut is *safe*. In the other cases we have a *depth-conflict*.

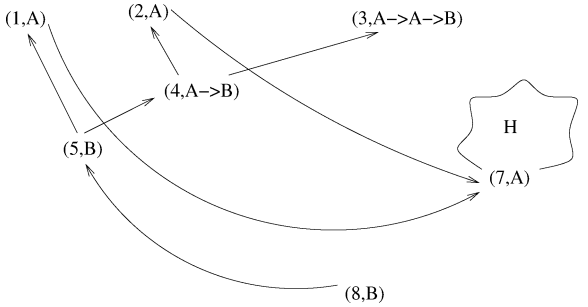


Fig. 11. Deduction graph of Figure 5 after a cut-elimination step.

without problems. In the general situation, in order to carry out cut-elimination, we may first have to do some 'R-elimination', 'unsharing' and 'incorporation' steps.

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 nodes $(n, A{\to}B)$ and (p, B) are at the same depth.

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

1 Remove the edges to and from n.
2 Remove the edge from (p, B) to (m, A).
3 Remove the box node n.
4 Add an edge from (p, B) to (j, B) (the B-node that n pointed to).
5 Add edges from $n_1, \ldots, n_k$ (the A-nodes that were inside the box) to (m, A).

Example 3.3. The deduction graph of Example 2.9 contains a cut that we can eliminate. If we do so, we obtain the deduction graph in Figure 11

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

Proof. We use Lemma 2.12. All nodes in G' are of the correct form: A, E, I or R. To see that G' is a closed box directed graph, we check that there are no edges pointing into a box: (m, A) is in the scope of $(n, A{\to}B)$, so (also after removing the box node), (m, A) is in the scope of $n_1, \ldots, n_k$ and the edges from $n_1, \ldots, n_k$ to m do not point into a box.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

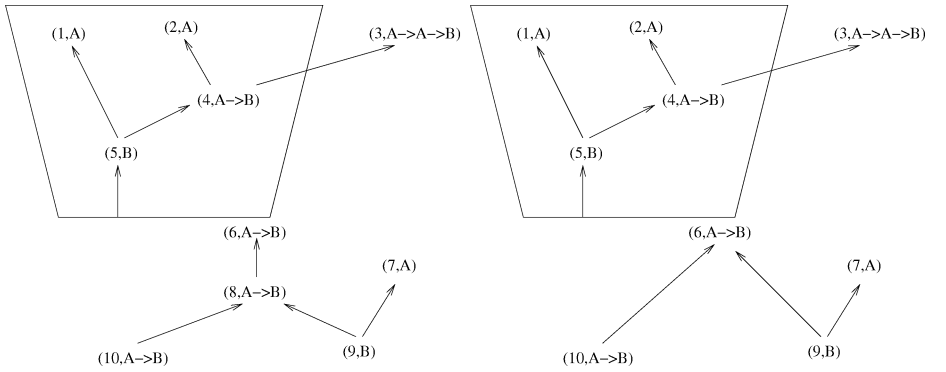


Fig. 12. Deduction graph with a cut hidden by a repeat and the same graph with the cut made explicit.

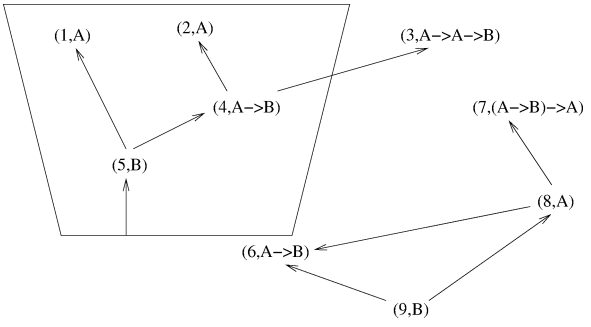


Fig. 13. Deduction graph with a cut hidden by sharing.

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 $\mathscr{B}$. So, without loss of generality, we may assume that $l > m$ for all $l \in \mathscr{B}$. □

If a cut is not safe, we need to make it safe before performing a cut-elimination step. A first problem with non-safe deduction graphs is that the →-I step and the →-E step could be separated by Repeats – see Figure 12 for an example. To eliminate the cut that arises from $A \to B$ and A, we must first 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:

1 When an edge points to n_1, redirect it to n_0.
2 Remove n_1.

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

Another problem arises when a boxed part is 'shared' – see Figure 13.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

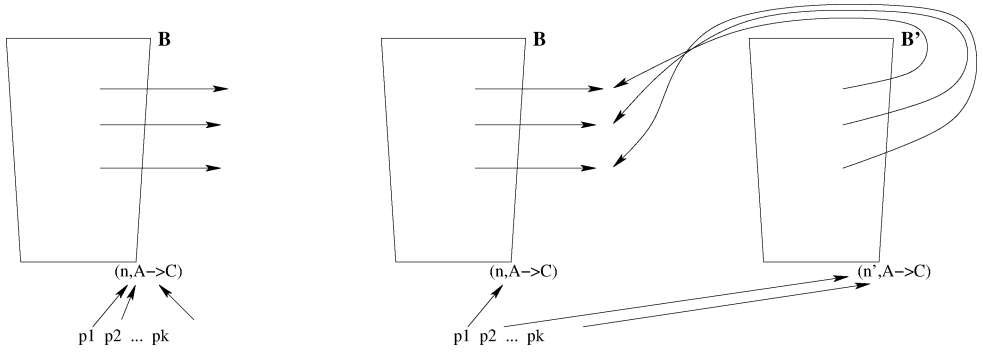


Fig. 14. Unsharing a deduction graph.

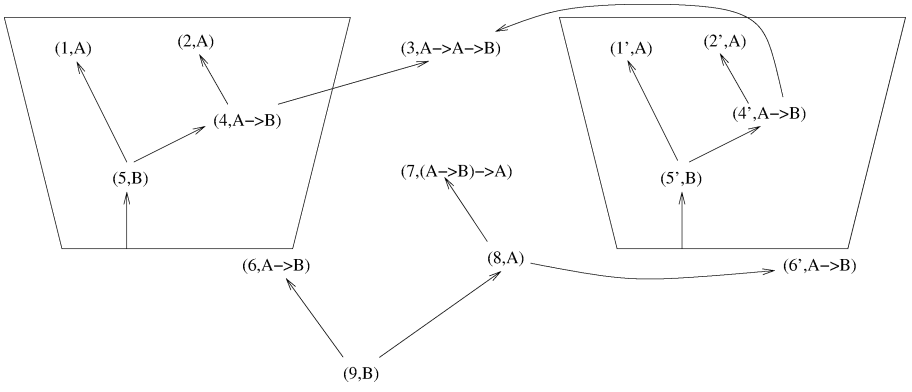


Fig. 15. The deduction graph of 13, with the cut made explicit.

Definition 3.7 (Cut hidden by sharing). Let G be a deduction graph that contains a box $\mathscr{B}$ with box-node $(n, A{\rightarrow}C)$ and $k \geqslant 2$ ingoing edges from $p_1, \ldots, p_k$. The *unsharing of G at nodes* $n, p_1, \ldots, p_k$ is obtained as follows (see Figure 14):

1 Make a box $\mathscr{B}'$, that contains a copy of all nodes and edges of $\mathscr{B}$.
2 Copy all the outgoing edges of $\mathscr{B}$ to $\mathscr{B}'$ (thus if we had $q \longrightarrow\!\triangleright\, m$ with $q \in \mathscr{B}$, $q' \in \mathscr{B}'$ and $m \notin \mathscr{B}$, we add $q' \longrightarrow\!\triangleright\, m$, where q' is the copy of q in $\mathscr{B}'$).
3 Let $p_2, \ldots p_k$ point to n' (the box-node of $\mathscr{B}'$) instead of n.

Figure 15 shows the unsharing of the deduction graph of Figure 13.

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

Proof. We use Lemma 2.12. The graph G' is a finite closed box directed graph. We make a box-topological ordering of G' by putting all nodes of $\mathscr{B}'$ immediately after the last node of $\mathscr{B}$ in the box-topological order of G. Finally, each node of G' satisfies one of the cases of Lemma 2.12. □

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

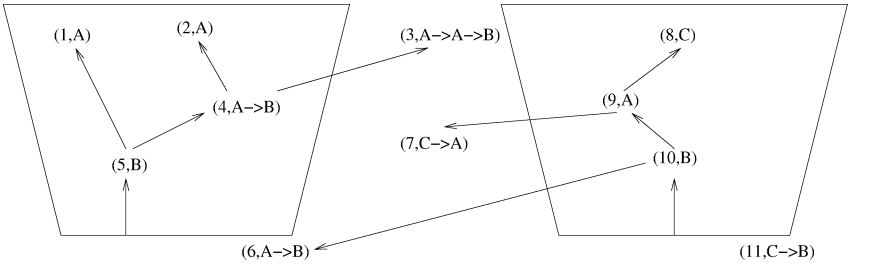


Fig. 16. Deduction graph with hidden cut due to a depth conflict.

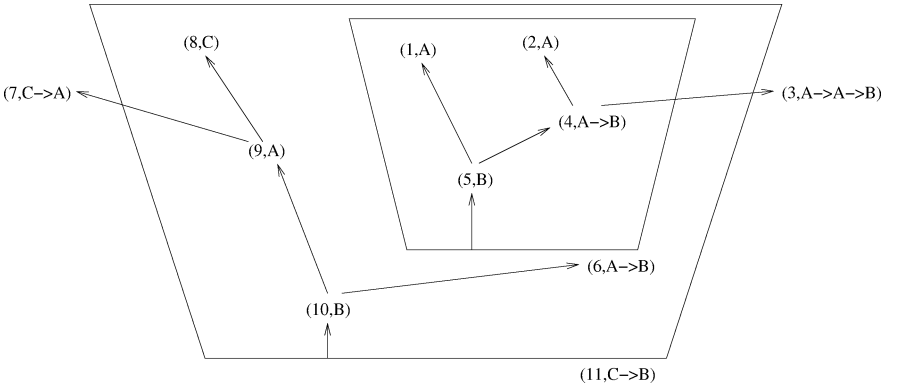


Fig. 17. Deduction graph of Figure 16 with the cut made explicit via an incorporation.

Finally, the problem of cuts being hidden as a consequence of a depth conflict is solved by *incorporating* one box within another – see Figure 16 for an example of a depth conflict.

Definition 3.9 (Cut hidden by a depth conflict, and incorporation). We have a *depth conflict* in the deduction graph G if G contains a box $\mathscr{B}$ with box-node $(n, A{\rightarrow}B)$ that has exactly one ingoing edge, from (p, B) say, 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 $\mathscr{B}$ into a deeper box that includes (p, B).

Figure 17 shows the incorporation of the deduction graph of Figure 16.

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 to be the same as for G. □

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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

1 *Making explicit cuts hidden by repeat* – Provided there is no edge from (p, B) to $(n, A{\to}B)$, 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 $(n, A{\to}B)$ and this is not the only edge to $(n, A{\to}B)$, perform the appropriate unsharing step, as defined in Definition 3.7;
3 *Making explicit cuts hidden by a depth conflict* – Provided (p, B) is not at the same depth as $(n, A{\to}B)$, 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*

In this section we map deduction graphs to λ-terms. Every node in a deduction graph can be mapped to a term, which corresponds 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 $(n, B) \longrightarrow\!\triangleright (p, A)$:

$$[\![G, n]\!] := [\![G_m, m]\!]\, [\![G_p, p]\!] .$$

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 $(n_1, A), \ldots, (n_k, A)$:

$$[\![G, n]\!] := \lambda x{:}A.([\![G_j, j]\!][x_{n_1} := x, \ldots, x_{n_k} := x]) .$$

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

$$[\![G, n]\!] := [\![G_m, m]\!] .$$

Note that $[\![G, n]\!] = [\![G_n, n]\!]$ for all n. This mapping ignores quite a lot of the 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 n be a node of G and G'. Then:

— If G' is obtained from G by an unsharing step, $[\![G, n]\!] = [\![G', n]\!]$.
— If G' is obtained from G by an incorporation step, $[\![G, n]\!] = [\![G', n]\!]$.
— If G' is obtained from G by an R-elimination step, $[\![G, n]\!] = [\![G', n]\!]$.

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.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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 $m \in \mathscr{B}$ iff $f(m) \in f(\mathscr{B})$.
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 $f(\mathscr{B})$. This clause should obviously be understood as '$m \in \mathscr{B}$ where $\mathscr{B}$ has box node n if and only if $f(m) \in \mathscr{B}'$ where $\mathscr{B}'$ has box node $f(n)$'.

A dg-substitution cannot change much of the graph: G' may contain parts that G does not 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, and let f be a dg-substitution from G to G'. We have

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

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 (we use the symbol $\overset{\text{IH}}{=}$ to indicate an application of the induction hypothesis).

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

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

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

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

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 free nodes of the box with box-node $f(n)$. Using the abbreviations

$$\begin{aligned} [\overrightarrow{x_n} := x] \;&\text{for}\; [x_{n_1} := x, \ldots, x_{n_l} := x] \\ [\overrightarrow{x_{f(n)}} := x] \;&\text{for}\; [x_{f(n_1)} := x, \ldots, x_{f(n_l)} := x] \\ [\overrightarrow{x_n} := [\![G', \overrightarrow{f(n)}]\!]] \;&\text{for}\; [x_{n_1} := [\![G', f(n_1)]\!], \ldots, x_{n_l} := [\![G', f(n_l)]\!]]\,, \end{aligned}$$

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

we have:

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

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, $f(n_i)$ is an A node also).

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

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

□

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 $[\![G, n]\!] \twoheadrightarrow\!\!\!\!\twoheadrightarrow_{\beta} [\![G', n]\!]$.

This lemma can be made more precise: if there is a path from n to the cut-formula, 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- and left-hand sides 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 (SN) 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 $G \twoheadrightarrow_{\text{cut}} G'$, 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 $\rightarrow$-I is empty (there are no $n_1, \ldots, n_k$), we create a new top-level 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 together into one 'term' all the λ-terms that arise from a node without incoming edges. To do this, we extend the simply typed λ-calculus with a

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

new term construction $\langle -, \ldots \rangle$. So we have the following new term syntax and additional derivation rule:

$$\mathsf{T}_{\langle\rangle} ::= x \mid (\mathsf{T}_{\langle\rangle}\mathsf{T}_{\langle\rangle}) \mid \lambda x : A.\mathsf{T}_{\langle\rangle} \mid \langle \mathsf{T}_{\langle\rangle}, \mathsf{T}_{\langle\rangle}, \ldots, \mathsf{T}_{\langle\rangle} \rangle$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N_1 : B_1 \quad \ldots \quad \Gamma \vdash N_k : B_k}{\Gamma \vdash \langle M, N_1, \ldots, N_k \rangle : A} .$$

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.

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

As can be observed from the rules, the $N_1, \ldots, N_k$ in $\langle M, N_1, \ldots, N_k \rangle$ act as a kind of 'garbage'. The order of the terms in $N_1, \ldots, N_k$ is irrelevant, so we consider terms *modulo* permutation of these vectors, which we will write as $\equiv_p$.

The ideas that we use in this section are very reminiscent of Klop (1980), where techniques are proposed (and used) to prove strong normalisation from weak normalisation (WN). This goes back to older work (Nederpelt 1973), which proved SN for Automath systems via a proof of WN. A good overview of Klop and Nederpelt's work, and related later work, can be found in Sørensen's Ph.D. thesis (Sørensen 1997). The $\lambda{\to}\langle\rangle$-calculus is close to a simply typed variant of the λ-calculus with the *non-erasing* reductions of Klop (1980). The difference is mainly in the fact that Klop uses pairs, while we use (more general) tuples. Moreover, Klop studies the normalisation of terms $\iota(M)$, where $\iota(M)$ is a term where all erasing redexes (K-redexes) are turned into non-erasing ones (I-redexes). In our formalism, the reduction rules $\bar\beta$ themselves are non-erasing, but, and most importantly, the use of the new reductions is different: Klop uses the non-erasing reductions to prove that WN for $\bar\beta$ implies SN for β, while we are interested in proving SN for $\bar\beta$ itself. We could have used Klop's ideas to prove SN for $\bar\beta$ via WN for $\bar\beta$, but we will instead prove SN for $\bar\beta$ directly, because it is shorter.

The last three reduction rules for $\lambda{\to}\langle\rangle$ make sure that a $\langle -\rangle$ pair can move up through the term structure and disappear – as long as the term does not contain abstractions. Put another way, applications can be moved under a $\langle -\rangle$. We make this explicit via the following definition of $\lambda{\to}\langle\rangle$-*context* and lemma.

Definition 4.7. A $\lambda{\to}\langle\rangle$ *context* is given by the following abstract syntax $K[-]$.

$$K[-] := [-] \mid \mathsf{T}_{\langle\rangle}K[-] \mid K[-]\mathsf{T}_{\langle\rangle} .$$

So a $\lambda{\to}\langle\rangle$ *context* is a $\lambda{\to}\langle\rangle$-term consisting only of applications (no abstractions) with one open place. The following lemma is immediate by induction on $K[-]$.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

Lemma 4.8. For all $\lambda{\to}\langle\rangle$ contexts $K[-]$ and $\lambda{\to}\langle\rangle$-terms $M, N_1, \ldots, N_k$

$$K[\langle M, N_1, \ldots, N_k\rangle] \twoheadrightarrow_{\bar{\beta}} \langle K[M], N_1, \ldots, N_k\rangle\,.$$

We now give an interpretation $\langle\![-]\!\rangle$ of deduction graphs as $\lambda{\to}\langle\rangle$-terms and 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 conclude that cut-elimination for deduction graphs is strongly normalising.

Definition 4.9. 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 $m_1, \ldots, m_t$, then

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

The interpretation $\langle\![G]\!\rangle$ of the deduction graph G is defined as $\langle\langle\![G, r_1]\!\rangle, \ldots, \langle\![G, r_l]\!\rangle\rangle$, where $r_1, \ldots, r_l$ are the top-level nodes without incoming edges in the deduction graph G.

Lemma 4.10 (Cut-elimination is $\bar{\beta}$-reduction in $\lambda{\to}\langle\rangle$). If $G \twoheadrightarrow_{\mathrm{cut}} G'$, then $\langle\![G]\!\rangle \twoheadrightarrow^{+}_{\bar{\beta}} \langle\![G']\!\rangle$.

Proof. The proof is by induction on the structure of G. Note that if G' is obtained from G via a 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. Looking at the subterm $\langle\![G, p]\!\rangle$, we observe the following (bear Figure 9 in mind):

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

where the superscript * denotes the substitution

$$[x_{n_1} := \langle\![G, m]\!\rangle, \ldots, x_{n_k} := \langle\![G, m]\!\rangle]\,.$$

This is the case where

$$x \in \mathrm{FV}(\langle\langle\![G, j]\!\rangle, \langle\![G, m_1]\!\rangle, \ldots, \langle\![G, m_t]\!\rangle\rangle[x_{n_1} := x, \ldots, x_{n_k} := x])\,,$$

that is, where the discharge is non-empty.

For the case where

$$x \notin \mathrm{FV}(\langle\langle\![G, j]\!\rangle, \langle\![G, m_1]\!\rangle, \ldots, \langle\![G, m_t]\!\rangle\rangle[x_{n_1} := x, \ldots, x_{n_k} := x])$$

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

(an empty discharge), we find

$$\begin{aligned}
\langle\![G,p]\!\rangle &\equiv \langle\![G,n]\!\rangle\langle\![G,m]\!\rangle \\
&\equiv (\lambda x{:}A.\langle\langle\![G,j]\!\rangle,\langle\![G,m_1]\!\rangle,\ldots,\langle\![G,m_t]\!\rangle\rangle[x_{n_1} := x,\ldots,x_{n_k} := x])\langle\![G,m]\!\rangle \\
&\longrightarrow_{\bar\beta} \langle\langle\![G,j]\!\rangle,\langle\![G,m_1]\!\rangle,\ldots,\langle\![G,m_t]\!\rangle\rangle,\langle\![G,m]\!\rangle\rangle \\
&\longrightarrow_{\bar\beta} \langle\langle\![G,j]\!\rangle,\langle\![G,m_1]\!\rangle,\ldots,\langle\![G,m_t]\!\rangle,\langle\![G,m]\!\rangle\rangle \\
&\equiv \langle\langle\![G',p]\!\rangle,\langle\![G,m_1]\!\rangle,\ldots,\langle\![G,m_t]\!\rangle,\langle\![G,m]\!\rangle\rangle\,.
\end{aligned}$$

If we look at $\langle\![G,p]\!\rangle$ as a subterm of $\langle\![G]\!\rangle$, we first note that it occurs as a subterm of a $\langle\ldots,\ldots\rangle$ expression as follows: $\langle\![G]\!\rangle = \langle\ldots,K[\langle\![G,p]\!\rangle],\ldots\rangle$.

In a cut-elimination, the box is being removed and the nodes $m_1,\ldots,m_t$ are moved one level up (in terms of depth), thus we find, for the case where the discharge is not empty,

$$\begin{aligned}
\langle\![G]\!\rangle &\equiv \langle\ldots,K[\langle\![G,p]\!\rangle],\ldots\rangle \\
&\twoheadrightarrow^{+}_{\bar\beta} \langle\ldots,K[\langle\langle\![G',p]\!\rangle,\langle\![G,m_1]\!\rangle^{*},\ldots,\langle\![G,m_t]\!\rangle^{*}\rangle],\ldots\rangle \\
&\twoheadrightarrow_{\bar\beta} \langle\ldots,\langle K[\langle\![G',p]\!\rangle],\langle\![G,m_1]\!\rangle^{*},\ldots,\langle\![G,m_t]\!\rangle^{*}\rangle,\ldots\rangle \\
&\longrightarrow_{\bar\beta} \langle\ldots,K[\langle\![G',p]\!\rangle],\langle\![G,m_1]\!\rangle^{*},\ldots,\langle\![G,m_t]\!\rangle^{*},\ldots\rangle \\
&\equiv_p \langle\![G']\!\rangle\,.
\end{aligned}$$

For the case where the discharge is empty, we find, in a similar way,

$$\begin{aligned}
\langle\![G]\!\rangle &\twoheadrightarrow^{+}_{\bar\beta} \langle\ldots,K[\langle\![G',p]\!\rangle],\langle\![G,m_1]\!\rangle,\ldots,\langle\![G,m_t]\!\rangle,\langle\![G,m]\!\rangle,\ldots\rangle \\
&\equiv_p \langle\![G']\!\rangle\,.
\end{aligned}$$

□

We now prove strong normalisation of $\longrightarrow_{\bar\beta}$. 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 show that this interpretation is sound. All types will be interpreted as a so-called *saturated set*.

Definition 4.11. A set X of $\lambda{\to}\langle\rangle$-terms is called *saturated* if:

— $X \subset \mathrm{SN}_{\bar\beta}$.
— $\langle x\vec{P},N_1,\ldots,N_k\rangle \in X$ (if $k=0$, this is $x\vec{P}$) for all variables x and all $P_1,\ldots,P_l$, $N_1,\ldots,N_k \in \mathrm{SN}_{\bar\beta}$, where $\vec{P} = P_1\ldots P_l$.
— If $M[x := N]\vec{P} \in X$, then $(\lambda x{:}A.M)N\vec{P} \in X$, if $N \in \mathrm{SN}_{\bar\beta}$.
— If $M \in X$, then $\langle M,P_1,\ldots,P_k\rangle \in X$, if $P_1,\ldots,P_k \in \mathrm{SN}_{\bar\beta}$.
— If $\langle MN,P_1,\ldots,P_k\rangle\vec{R} \in X$, then $\langle M,P_1,\ldots,P_k\rangle N\vec{R} \in X$.

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

Definition 4.12. The interpretation of types as saturated sets $\mathscr{V}(-)$ is defined as follows:

— $\mathscr{V}(\alpha) := \mathrm{SN}_{\bar\beta}$.
— $\mathscr{V}(A{\to}B) := \{M \mid \forall N \in \mathscr{V}(A)(MN \in \mathscr{V}(B))\}$.

Lemma 4.13. For all types A, $\mathscr{V}(A)$ is a saturated set.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

Proof. The proof is by induction on A. If A is a variable α, $\mathscr{V}(\alpha) = \mathrm{SN}_{\bar{\beta}}$, 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, supposing that $\mathscr{V}(A)$ and $\mathscr{V}(B)$ are saturated, we have to verify the closure conditions for saturated sets for $\mathscr{V}(A{\to}B)$.

— $\mathscr{V}(A{\to}B)$ is obviously a subset of $\mathrm{SN}_{\bar{\beta}}$ (given that $\mathscr{V}(B) \subset \mathrm{SN}_{\bar{\beta}}$ and $\mathscr{V}(A) \neq \varnothing$).
— $\langle x\vec{P}, N_1, \ldots, N_k\rangle \in \mathscr{V}(A{\to}B)$ for all variables x and all $P_1, \ldots, P_l$, $N_1, \ldots, N_k \in \mathrm{SN}_{\bar{\beta}}$, where $\vec{P} = P_1 \ldots P_l$, because for all $Q \in \mathscr{V}(A)$ we have $\langle x\vec{P}Q, N_1, \ldots, N_k\rangle \in \mathscr{V}(B)$ and thus $\langle x\vec{P}, N_1, \ldots, N_k\rangle Q \in \mathscr{V}(B)$. So $\langle x\vec{P}, N_1, \ldots, N_k\rangle \in \mathscr{V}(A{\to}B)$.
— Suppose $M[x := N]\vec{P} \in \mathscr{V}(A{\to}B)$. Then for all $Q \in \mathscr{V}(A)$, we have $M[x := N]\vec{P}Q \in \mathscr{V}(B)$. So $(\lambda x{:}A.M)N\vec{P}Q \in \mathscr{V}(B)$, and thus $(\lambda x{:}A.M)N\vec{P} \in \mathscr{V}(A{\to}B)$.
— Suppose $M \in \mathscr{V}(A{\to}B)$. Then for all $Q \in \mathscr{V}(A)$, we have $MQ \in \mathscr{V}(B)$. So for all $Q \in \mathscr{V}(A)$ and for all $N_1, \ldots, N_k \in \mathrm{SN}_{\bar{\beta}}$, we have $\langle MQ, N_1, \ldots, N_k\rangle \in \mathscr{V}(A)$. Hence, for all $Q \in \mathscr{V}(A)$ and for all $N_1, \ldots, N_k \in \mathrm{SN}_{\bar{\beta}}$, we have $\langle M, N_1, \ldots, N_k\rangle Q \in \mathscr{V}(A)$. We can then conclude that for all $N_1, \ldots, N_k \in \mathrm{SN}_{\bar{\beta}}$, we have $\langle M, N_1, \ldots, N_k\rangle \in \mathscr{V}(A{\to}B)$.
— Suppose $\langle MN, P_1, \ldots, P_k\rangle\vec{R} \in \mathscr{V}(A{\to}B)$. Then for all $Q \in \mathscr{V}(A)$, we have that $\langle MN, P_1, \ldots, P_k\rangle\vec{R}Q \in \mathscr{V}(B)$. So for all $Q \in \mathscr{V}(A)$, we have $\langle M, P_1, \ldots, P_k\rangle N\vec{R}Q \in \mathscr{V}(B)$ and can conclude that $\langle M, P_1, \ldots, P_k\rangle N\vec{R} \in \mathscr{V}(A{\to}B)$. □

Lemma 4.14 (Soundness of $\mathscr{V}(-)$). If $x_1 : A_1, \ldots, x_n : A_n \vdash M : B$ in $\lambda{\to}\langle\rangle$ and $N_1 \in \mathscr{V}(A_1), \ldots N_n \in \mathscr{V}(A_n)$, then $M[x_1 := N_1, \ldots, x_n := N_n] \in \mathscr{V}(B)$.

Proof. The proof is by induction on the derivation. All cases are straightforward; as an illustration we just treat two. For simplicity we just use M^* to denote $M[x_1 := N_1, \ldots, x_n := N_n]$.

abs In this case, $\Gamma \vdash \lambda x{:}A.M : A{\to}B$ was derived from $\Gamma, x{:}A \vdash M : B$. As IH we have $\forall Q \in \mathscr{V}(A)(M^*[x := Q] \in \mathscr{V}(B))$. One of the closure conditions of saturated sets then says that $\forall Q \in \mathscr{V}(A)((\lambda x{:}A.M^*)Q \in \mathscr{V}(B))$, so $\lambda x{:}A.M^* \in \mathscr{V}(A{\to}B)$.

$\langle -, \ldots\rangle$ In this case, $\Gamma \vdash \langle M, P_1, \ldots, P_k\rangle : A$ was derived from $\Gamma \vdash M : A$ and $\Gamma \vdash P_1 : B_1$, $\ldots$, $\Gamma \vdash P_k : B_k$. As IH we have $M^* \in \mathscr{V}(A)$ and $P_i^* \in \mathscr{V}(B_i)$ for $1 \leqslant i \leqslant k$ (and thus $P_i^* \in \mathrm{SN}_{\bar{\beta}}$ for $1 \leqslant i \leqslant k$). One of the closure conditions of saturated sets then says that $\langle M, P_1, \ldots, P_k\rangle^* \in \mathscr{V}(A)$. □

Corollary 4.15. $\bar{\beta}$-reduction is strongly normalising for $\lambda{\to}\langle\rangle$, so cut-elimination is terminating for deduction graphs.

Proof. Taking $N_i := x_i$ in the Lemma, we find that $M \in B \subset \mathrm{SN}_{\bar{\beta}}$, so M is strongly normalising. Hence cut-elimination is terminating for deduction graphs because of Lemma 4.10. □

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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, and not just the part that is relevant for some node n. This would be useful in the case of, for example, an $\rightarrow$-E rule, where we want to combine two parts of an existing graph.
— Which parts of G are shared.
— Which parts of G are repeated.
— 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', that is, 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*.

The translation from deduction graphs to contexts is a bit involved, as are the reduction rules on contexts that correspond to the reduction rules on deduction graphs. This is to be expected, because we want to represent a graph faithfully as an expression. On the other hand, this translation to contexts also clarifies a couple of things. First, it clarifies the translation $\langle\!|[-]\!|\rangle$ to simply typed terms, as it can be factored through the translation to contexts, which we will now define (this is made precise in Appendix A). Furthermore, the translation to contexts clarifies the connection with let calculi, for example, see Maraist *et al.* (1998). Our contexts have 'let declarations' of the form $\mathsf{let}\, x = P \,\mathsf{in}\, Q$, which are used to represent the sharing. (These let declarations act as an *explicit substitution*, where P is to be substituted for x in Q.) The reduction rules on our contexts show how the reduction rules on deduction graphs can be understood as an explicit substitution calculus.

Before giving the interpretation, we will make the $\lambda{\rightarrow}C\mathsf{let}$-calculus precise. We first define the set of *expressions* and the operation of *filling holes*, and 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 $E_1[E_2]$, *filling the holes of* E_1 *with* E_2, as follows:

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

where E, F and G are expressions.

Definition 5.3. The calculus of contexts $\lambda{\rightarrow}C\mathsf{let}$ is defined as follows. Types are the simple types, Typ. The syntactic domains are

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

A context C has exactly one 'hole', $[-]$. We sometimes write $C[-]$ to emphasise 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*).

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

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

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

We now list some properties that will help to give an intuitive understanding of the $\lambda{\to}C\mathsf{let}$-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 $1 \leqslant i \leqslant n$).
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 $\lambda{\to}C\mathsf{let}$ $\langle\!\langle G \rangle\!\rangle_\varphi$ as follows using induction on the number of nodes and boxes 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 $(n, A{\to}B) \longrightarrow\!\triangleright (j, B)$ with associated box $\mathscr{B}$, then

$$\langle\!\langle G \rangle\!\rangle_\varphi = \langle\!\langle G \setminus \mathscr{B} \rangle\!\rangle_{\varphi_{|G\setminus\mathscr{B}}} [\mathsf{let}\, x_n = (\lambda x_m{:}A.\langle\!\langle \mathscr{B}^m \rangle\!\rangle_{\varphi_{|\mathscr{B}^m}} [x_j]) \,\mathsf{in}\, [-]]$$

where $\varphi_{|\mathscr{B}^m}$ is $\varphi_{|\mathscr{B}}$ with some extra minimal elements.

— If the φ-maximal element, say n, is an E-node with $(n, B){\longrightarrow\!\triangleright}(i, A{\to}B)$ and $(n, B){\longrightarrow\!\triangleright}(j, A)$, then

$$\langle\!\langle G \rangle\!\rangle_\varphi := \langle\!\langle G \setminus n \rangle\!\rangle_{\varphi_{|G\setminus n}} [\mathsf{let}\, x_n = (x_i x_j) \,\mathsf{in}\, [-]]\,.$$

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

— If the φ-maximal element, say n, is an R-node with $n \longrightarrow\!\triangleright\, l$, 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. As an example, we give the associated contexts in $\lambda{\to}C\mathsf{let}$ for the three simple deduction graphs of Figure 4. Note that in example III, where 'garbage' occurs inside a box, the $\mathsf{let}\, x_3 = \ldots$ declaration corresponds to the piece of 'garbage'. This declaration is under the λ-abstraction $(\lambda x_p{:}B.\ldots)$.

The translations of the deduction graphs of Figure 4 together with their contexts are:

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

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 $\lambda{\to}C\mathsf{let}$ context yields a well-formed context indeed.

Proof. The proof is 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. As an example, we give the interpretations of the deduction graphs in Figure 5 (G) and Figure 13 (G'), including the contexts they are well-formed in. We assume that $\langle\!\langle H \rangle\!\rangle = C[-]$ in the context $\Gamma; \Delta$.

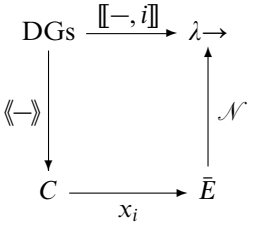
$$\begin{aligned}
\langle\!\langle G \rangle\!\rangle \;=\; & \mathsf{let}\, x_6 = (\lambda x{:}A.\mathsf{let}\, x_1 = x, x_2 = x, x_4 = (x_3 x_2), x_5 = (x_4 x_1)\,\mathsf{in}\, x_5)\,\mathsf{in} \\
& C[\mathsf{let}\, x_8 = (x_6 x_7)\,\mathsf{in}\, [-]] \\
& \text{in context } \Gamma \cup x_3{:}A{\to}A{\to}B\,;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

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-level node of G. So, for x_i any top-level variable (that is, any variable of $\Gamma \cup \Delta$), we have $\langle\!\langle G \rangle\!\rangle[x_i]$ 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)

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

all let-redexes, we obtain the λ-term $[\![G, i]\!]$ of Definition 4.1. The connections are spelled out in detail in Appendix A; here we just describe the situation in the diagram



Here, DGs stands for the collection of deduction graphs and $\bar{E}$ is the collection of expressions of the simply typed lambda calculus with let, of which $\lambda\!\!\rightarrow$ is an obvious subset. $\mathcal{N}$ is the function that contracts all let-redexes (so a $\mathsf{let}\, x := P \,\mathsf{in}$ is replaced by a substitution). The map indicated by x_i substitutes the variable x_i in the hole of the context. The diagram states that if G is a deduction graph with top-level node i, then

$$\mathcal{N}(\langle\!\langle G \rangle\!\rangle[x_i]) \equiv [\![G, i]\!]\,.$$

One can recognise separate parts of the cut-elimination process in the translation $\langle\!\langle G \rangle\!\rangle$. We now define a reduction relation on $\lambda\!\!\rightarrow C\mathsf{let}$-contexts that captures cut-elimination on deduction graphs.

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

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

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

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

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

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

And smilarly for the other rewrite rules and for $\simeq$.

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

Remark 5.11. Note that the contextual closure of the rewrite rules that we used in the previous definition takes side conditions concerning the free variables into account.

As a consequence, it may happen that $C[-] \to_B D[-]$ but *not* $\lambda u{:}B.C[x] \to_B \lambda u{:}B.D[x]$, because $x \notin \mathrm{FV}(C[-])$, which allows $C[-]$ to reduce, but $\lambda u{:}B.C[x]$ is not allowed to reduce because $x \in \mathrm{FV}(C[x])$. 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).

Conversely, it may happen that $\lambda u{:}B.C[x] \to_{CP} \lambda u{:}B.D[x]$, but *not* $C[-] \to_{CP} D[-]$, because $x \notin \mathrm{FV}(C[-])$, which prevents the reduction of $C[-]$, but $x, y \in \mathrm{FV}(C[x])$, which allows the reduction of $\lambda u{:}B.C[x]$. 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 $\lambda u{:}B.\mathsf{let}\, y = x_1 x_2\, \mathsf{in}\, \mathsf{let}\, x = x_1 x_2\, \mathsf{in}\, \mathsf{let}\, x_3 = x_4 y\, \mathsf{in}\, x$, while 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 the result 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 use Δ^C to denote the let-context of 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 $C \to_{\mathrm{let}} D$, then $\Gamma; \Delta^D \vdash D$, and if $\Gamma \vdash M : A$ and $M \to_{\mathrm{let}} N$, then $\Gamma \vdash N : A$.

See Appendix B 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 C 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 $\langle\!\langle G \rangle\!\rangle \to_{CP} \langle\!\langle G' \rangle\!\rangle$.
2 If G' is obtained from G by an incorporation step, then $\langle\!\langle G \rangle\!\rangle \to_{CM} \langle\!\langle G' \rangle\!\rangle$.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

3 If G' is obtained from G by a repeat eliminating step, then $\langle\!\langle G \rangle\!\rangle \to_L \langle\!\langle G' \rangle\!\rangle$.
4 If G' is obtained from G by eliminating a safe cut, then $\langle\!\langle G \rangle\!\rangle \to_B \langle\!\langle G' \rangle\!\rangle$.

See Appendix D for a proof.

6. Future work

6.1. *Relation to 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 it is the 'discharging' of a hypothesis; in proof nets it 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 recall 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 postulate that $A^{\perp\perp} = A$ for every $A \in \mathsf{ATOM}$. The formulas are given by the following grammar:

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

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

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

For the definition of proof nets, we have the 'natural deduction' counterpart of linear logic derivations, which are usually given in sequent form, see Figure 18. Note particularly the Contraction rule, which expresses sharing. Note also that there is a Cut rule. So cut-elimination in proof nets expresses the redundancy of 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 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, which is similar to what happens during the elimination of a safe cut 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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

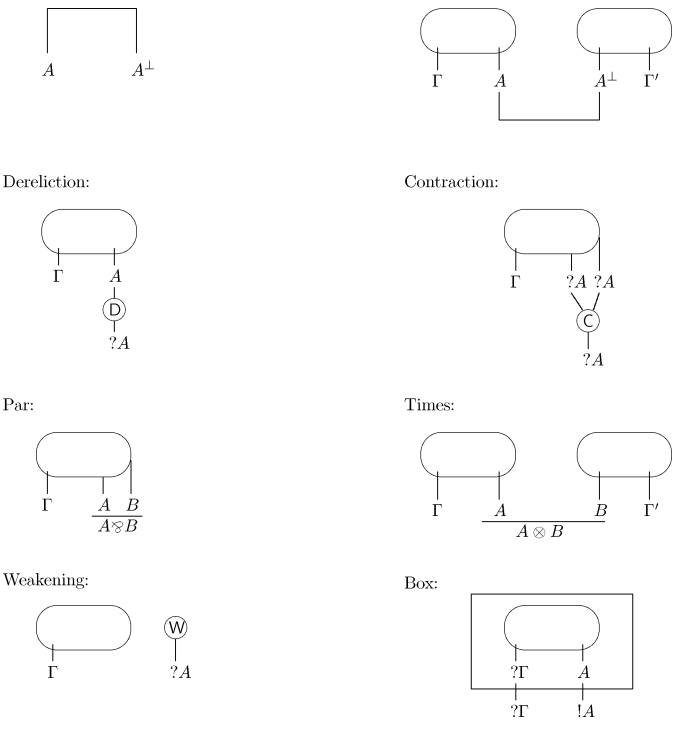


Fig. 18. Proof nets.

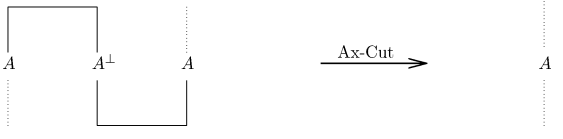


Fig. 19. The Ax-cut rule.

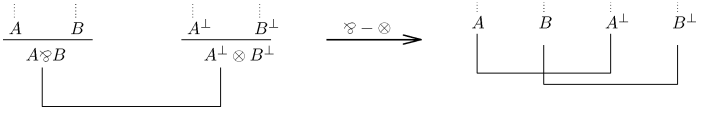


Fig. 20. The ⅋ – ⊗ rule.

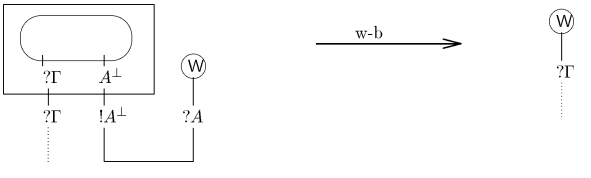


Fig. 21. The w-b rule.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

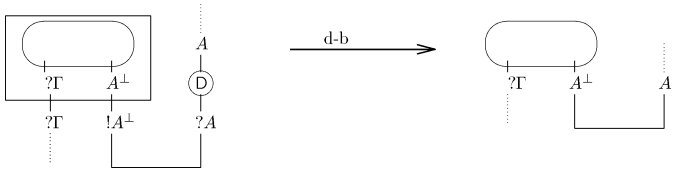


Fig. 22. The d-b rule.

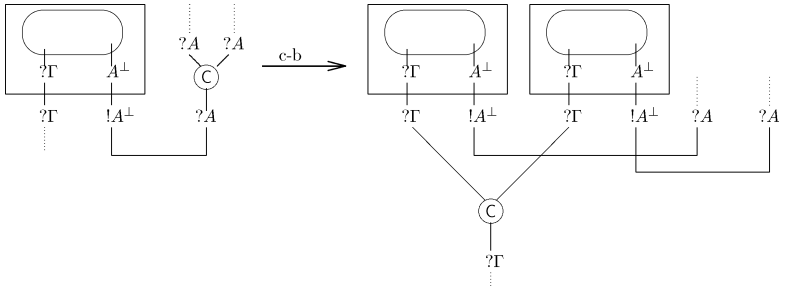


Fig. 23. The c-b rule.

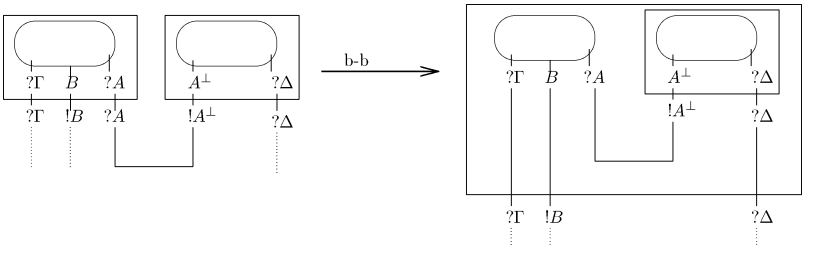


Fig. 24. The b-b rule.

(Lafont 1990)), is that nodes in the latter have a fixed arity, whereas nodes in the former do 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 emphasises the fact that deduction graphs are meant for resource-insensitive propositional logic and are not a way to depict resource-sensitive linear logic proofs. In natural deduction formalisms, even in 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 2.13, are themselves deduction graphs.

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 $\lambda{\rightarrow}C\mathsf{let}$-calculus with explicit contractions (see Kesner and Lengrand (2005) for a calculus with

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

explicit contractions). From these enriched structures, the similarities shared 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 simply typed λ-calculus. This is a bit *ad hoc*. The other reduction preserving map that we have defined maps deduction graphs to contexts in $\lambda{\rightarrow}C\mathsf{let}$, 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 of $\twoheadrightarrow_{\mathrm{let}}$ in $\lambda{\rightarrow}C\mathsf{let}$. However, the $\twoheadrightarrow_{\mathrm{let}}$-reduction rules are quite complicated and do not lend themselves very well to a detailed analysis. It would be interesting to see whether we can prove SN for the $\twoheadrightarrow_{\mathrm{let}}$-reduction rules (or an appropriate restriction or a variant of them) and conclude termination of cut-elimination from that.

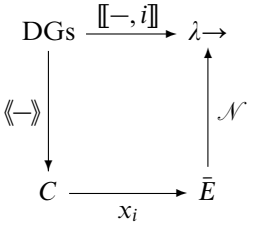
Another interesting issue is to study the flexibility of the notion of a deduction graph. For example: does it generalise easily to full propositional logic and predicate logic? what if we allow, for example, overlapping boxes? or what if we use a variant of natural deduction with a sequent notation? Preliminary studies show that it is easy to extend deduction graphs to full propositional logic.

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 this point there are many potentially interesting links to the research fields of term-graph reduction (for implementing functional programs), optimal λ-reduction (for making optimal use of sharing) and interaction nets (which provide a very basic graphical computational model).

Finally, deductions are static objects, which 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 an 'open' (unfinished) deduction graph.

Appendix A. Connection between $\langle\!\langle G \rangle\!\rangle[x_i]$ and $[\![G, i]\!]$

We want to give a formal statement of the connection between $[\![-,-]\!]$ and $\langle\!\langle - \rangle\!\rangle$ as already mentioned in Section 5. That is, we want to prove that the following diagram commutes: for i a top-level node of a deduction graph G, we want to prove that $\mathcal{N}(\langle\!\langle G \rangle\!\rangle[x_i]) = [\![G_i, i]\!]$, where $\mathcal{N}$ is the function that contracts let-redexes:



We will now make the ingredients of this diagram precise.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

To capture both simply typed terms and terms of the form $C[x]$, where $\Gamma;\Delta \vdash C$ and $x \in \Gamma \cup \Delta$, we consider the following syntactic domain of λ-let-expressions:

$$\bar{E} \;:=\; x \mid (\bar{E}\bar{E}) \mid \mathsf{let}\, x = \bar{E}\, \mathsf{in}\, \bar{E} \mid \lambda x{:}A.\bar{E}\,.$$

Definition A.1. We define the following typing rules for λ-let-terms:

$$\frac{\Gamma \vdash_l T : A \quad \Gamma, x{:}A \vdash_l S{:}B}{\Gamma \vdash_l \mathsf{let}\, x = T\, \mathsf{in}\, S{:}B}\ \mathsf{let}_l\text{-rule} \qquad \Gamma \vdash_l x : A \quad \text{if } (x{:}A) \in \Gamma$$

$$\frac{\Gamma, x{:}A \vdash_l T}{\Gamma \vdash_l \lambda x{:}A.T : A{\to}B}\ \lambda_l\text{-rule} \qquad \frac{\Gamma \vdash_l T : A{\to}B \quad \Gamma \vdash_l S : A}{\Gamma \vdash_l TS : B}\ \mathrm{app}_l\text{-rule}.$$

Lemma A.2.

1 If $\Gamma \vdash M{:}A$, then $\Gamma \vdash_l M{:}A$.
2 If $\Gamma;\Delta \vdash C$ and $x{:}A \in \Gamma \cup \Delta$, then $\Gamma \vdash_l C[x]{:}A$.

Proof.

— If $\Delta = <>$ and $C = [-]$, then for $x \in \Gamma \cup \Delta$, we have $\Gamma \vdash_l x{:}A$.
— If $M = x$ and $x{:}A \in \Gamma$, then $\Gamma \vdash_l x{:}A$.
— If $\Gamma \vdash x{:}A{\to}B$ and $\Gamma \vdash y{:}A$, then, by induction, $\Gamma \vdash_l x{:}A{\to}B$ and $\Gamma \vdash_l y{:}A$, so by the app_l-rule, we get $\Gamma \vdash_l xy{:}B$.
— If $\Gamma \vdash T{:}A$ and $\Gamma, x{:}A;\Delta \vdash C$ and $y{:}B \in \Gamma \cup \Delta, x{:}A$, then, by induction, $\Gamma \vdash_l T{:}A$ and $\Gamma, x{:}A \vdash_l C[y]{:}B$, so by the let_l-rule, we get $\Gamma \vdash_l \mathsf{let}\, x = T\, \mathsf{in}\, C[y]{:}B$.
— If $\Gamma, x{:}A;\Delta \vdash C$ and $y{:}B \in \Delta$, then, by induction, $\Gamma, x{:}A \vdash_l C[y]{:}B$, so, by the λ_l-rule, we get $\Gamma \vdash_l \lambda x{:}A.C[y]{:}B$. □

Definition A.3. We define the reduction rule $\twoheadrightarrow_{l\mathsf{let}}$ as the contextual closure of the following rewrite rule on λ-let-terms:

$$\mathsf{let}\, x = T\, \mathsf{in}\, S \twoheadrightarrow_{l\mathsf{let}} S[x := T]\,.$$

Lemma A.4 (Substitution). If $\Gamma, x{:}A \vdash_l T{:}B$ and $\Gamma \vdash_l S{:}A$, then $\Gamma \vdash_l T[x := S]$.

Proof. The proof is by induction on the construction of T. □

Lemma A.5 (Subject reduction). If $\Gamma \vdash_l T$ and $T \twoheadrightarrow_{l\mathsf{let}} T'$, then $\Gamma \vdash_l T'$.

Proof. Suppose $\Gamma \vdash_l \mathsf{let}\, x = T\, \mathsf{in}\, S{:}B$. Then $\Gamma \vdash_l T{:}A$ and $\Gamma, x{:}A \vdash_l S{:}B$. Hence, by substitution, $\Gamma, \Gamma \vdash_l [x := S]$, which is $\Gamma \vdash_l T[x := S]$. □

Lemma A.6. The reduction $\twoheadrightarrow_{l\mathsf{let}}$ is strongly normalising and confluent.

Proof. We make a map I from λ-let-terms to terms of the simply typed λ-calculus as follows: $I(\mathsf{let}\, x = T\, \mathsf{in}\, S) = (\lambda x{:}A.I(S))\, I(T)$, and I is the identity on all other terms. Then, if $T \twoheadrightarrow_{l\mathsf{let}} T'$, we have $I(T) \twoheadrightarrow_\beta I(T')$. Because $\twoheadrightarrow_\beta$-reduction is strongly-normalising, $\twoheadrightarrow_{l\mathsf{let}}$-reduction is also. Noting that $\twoheadrightarrow_{l\mathsf{let}}$ is weakly confluent, we are done. □

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

We will write $\mathcal{N}(T)$ for the $l$let-normal form of T and let C be a well-formed context. We use the following properties:

1 If $(C[x_n\ x_m])$ is a well-typed λ-let-term, then

$$\mathcal{N}(C[x_n\ x_m]) = (\mathcal{N}(C[x_n])\ \mathcal{N}(C[x_m]))\,.$$

2 If $(C[\lambda x.D])$ is a well-typed λ-let-term, then

$$\mathcal{N}(C[\lambda x.D]) = \lambda x.\mathcal{N}(C[D])\,.$$

Definition A.7. Let G be a deduction graph with free nodes $(n_1, A), \ldots, (n_k, A)$. Then $G^{m,n_1,\ldots,n_k}$ is the graph consisting of G, a new node (m, A) and edges $n_1 \longrightarrow\!\triangleright\, m, \ldots, n_k \longrightarrow\!\triangleright\, m$.

Lemma A.8. Let G be a deduction graph with free nodes $(n_1, A), \ldots, (n_k, A)$. Then $G^{m,n_1,\ldots,n_k}$ is a deduction graph.

Lemma A.9. Let G be a deduction graph. If $(n, A{\to}B)$ is a box-node with $(n, A{\to}B) \rightarrowtriangle (j, B)$ and the A-nodes of the box are $(n_1, A), \ldots, (n_k, A)$, then

$$[\![G, n]\!] = \lambda x_m{:}A.[\![G_j^{m,n_1,\ldots,n_k}, j]\!]\,.$$

Lemma A.10 (Gluing lemma). Let G and F be deduction graphs such that for some 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

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

□

Theorem A.11. Let G be a deduction graph and i be a top-level node of G. Then

$$\mathcal{N}(\langle\!\langle G \rangle\!\rangle[x_i]) = [\![G_i, i]\!]\,.$$

Proof. Without loss of generality, we assume that i is a maximal node.

A If i is an A-node, then x_i is not let-bound in $\langle\!\langle G \rangle\!\rangle$, so $\mathcal{N}\langle\!\langle G \rangle\!\rangle[x_i] \equiv x_i \equiv [\![G_i, i]\!]$.

R If i is an R-node, say $i \longrightarrow\!\triangleright\, n$ for some node n, then

$$\begin{aligned}
\mathcal{N}(\langle\!\langle G \rangle\!\rangle[x_i]) &= \mathcal{N}(\langle\!\langle G \setminus i \rangle\!\rangle[\mathsf{let}\, x_i = x_n\ \mathsf{in}\ x_i]) \\
&= \mathcal{N}(\langle\!\langle G \setminus i \rangle\!\rangle[x_n]) \\
&= [\![G_n, n]\!] \\
&= [\![G_i, i]\!]\,.
\end{aligned}$$

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

E If i is an E-node, say $(i, B) \longrightarrow\!\triangleright (n, A{\to}B)$ and $(i, B){\to}(m, A)$, then

$$\begin{aligned}
\mathcal{N}(\langle\!\langle G\rangle\!\rangle[x_i]) &= \mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[\mathsf{let}\, x_i = (x_n\ x_m)\ \mathsf{in}\ x_i]) \\
&= \mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[(x_n\ x_m)]) \\
&= (\mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[x_n]\ \mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[x_m]) \\
&= ([\![G_n, n]\!]\ [\![G_m, m]\!]) \\
&= [\![G_i, i]\!]\,.
\end{aligned}$$

I If i is an I-node, say $(i, A{\to}B) \longrightarrow\!\triangleright (j, B)$, with associated box $\mathscr{B}$, then

$$\begin{aligned}
\mathcal{N}(\langle\!\langle G\rangle\!\rangle[x_i]) &= \mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[\mathsf{let}\, x_i = \lambda x_m{:}A.\langle\!\langle B^m\rangle\!\rangle[x_j]\ \mathsf{in}\ x_i]) && (1)\\
&= \mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[\lambda x_m{:}A.\langle\!\langle B^m\rangle\!\rangle[x_j]]) && (2)\\
&= \lambda x_m{:}A.\mathcal{N}(\langle\!\langle G \setminus i\rangle\!\rangle[\langle\!\langle B^m\rangle\!\rangle[x_j]]) && (3)\\
&= \lambda x_m{:}A.\mathcal{N}(\langle\!\langle G^{m,y_1,\ldots,y_k} \setminus i\rangle\!\rangle[x_j]) && (4)\\
&= \lambda x_m{:}A.[\![(G^{m,y_1,\ldots,y_k} \setminus i)_j, j]\!] && (5)\\
&= [\![G_i, i]\!]\,. && (6)
\end{aligned}$$

We use Lemma A.10 to go from (3) to (4) and Lemma A.9 to go from (5) to (6). □

Appendix B. 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, by adding a typing for the subexpresion $C[y]$ (which is not well-typed in $\lambda{\to}C\mathsf{let}$). Thus, we change the λ-rule into the *fill-rule* and the *abs-rule*:

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

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

$$\begin{aligned}
\Gamma;\Delta \vdash^+ C &\Leftrightarrow \Gamma;\Delta \vdash C \\
\Gamma \vdash^+ t : A &\Leftarrow \Gamma \vdash t : A \\
\Gamma \vdash^+ t : A \wedge t \neq C[y] &\Rightarrow \Gamma \vdash t : A\,.
\end{aligned}$$

We will now prove SR for $\lambda{\to}C\mathsf{let}^+$ and conclude SR for $\lambda{\to}C\mathsf{let}$ by observing that if $\Gamma \vdash t : A$ and $t \twoheadrightarrow_{\mathrm{let}} q$, then $q \neq C[y]$.

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

— *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 $\rightarrow_B$, $\rightarrow_{CM}$, $\rightarrow_{CP}$ or $\rightarrow_L$ is applied to a context on the 'top level'. Then there are 4 contextual closure cases:

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

We first consider the base cases where $\rightarrow_B$ or $\rightarrow_{CM}$ 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.

— The $\rightarrow_B$ case:

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

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 $\Gamma, x{:}A{\rightarrow}E, p{:}E; \Delta_{x,p} \vdash^+ C$, so $\Gamma, p{:}E; \Delta_{x,p} \vdash^+ C$ because $x \notin \mathrm{FV}(C)$.

From the second we conclude that $\Gamma, y{:}E, p{:}E; \Delta_{x,p} \vdash^+ C$, so

$$\Gamma, y{:}E; p{:}E, \Delta_{x,p} \vdash^+ \mathsf{let}\, p = y \,\mathsf{in}\, C\,.$$

From this, using context substitution and the first, we derive

$$\Gamma; \Delta^D, p{:}E, \Delta_{x,p} \vdash^+ D[z := q][\mathsf{let}\, p = y \,\mathsf{in}\, C]\,.$$

— The $\rightarrow_{CM}$ case:

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

where $x \notin \mathrm{FV}(C)$. We find that:

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

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

We will only consider the most interesting case for the contextual closure, in which $C[v] \rightarrow E$ on the top level. There are four subcases:

1,2 $C[v] \rightarrow_B D[v]$ or $C[v] \rightarrow_{CM} D[v]$:
Note that in D, v is still a let-abstracted variable. Now, $C[-] \rightarrow D[-]$, so, by the induction hypothesis, $\Gamma; \Delta^D \vdash^+ D[-]$, and $v : B$ must be in Δ^D, so $\Gamma \vdash^+ D[v] : B$.

3 $C[v] \rightarrow_{CP} D[v]$:
In this case $C[v] = \mathsf{let}\, x = P \,\mathsf{in}\, E[y := x]$, 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 $\Gamma \vdash^+ \mathsf{let}\, y = P \,\mathsf{in}\, \mathsf{let}\, x = P \,\mathsf{in}\, E$.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

4 $C[v] \twoheadrightarrow_L D[v]$:
In this case $C[v] = \mathsf{let}\, y = x \,\mathsf{in}\, G[v]$, 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 C. Independence of $\langle\!\langle G \rangle\!\rangle_\varphi$ on the topological ordering φ

This section constitutes the proof of Lemma 5.14.

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

The following two lemmas give two simple properties of partial orderings.

Lemma C.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 C.3. Let G be a deduction graph, φ be the partial ordering imposed by the link graph and 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, so $\mathsf{CL}(\varphi \cup \psi)$ is also 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 2.12, 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, we will use φ to indicate the linear ordering as well as the associated isomorphism.

Definition C.4. Let φ be a linear order of n elements. We define for every $i \leqslant n - 2$ 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 C.5. Let G be a set and 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

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

and

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

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

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

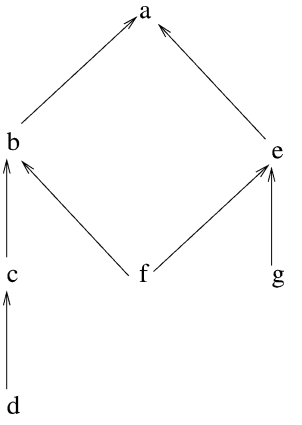
$$\begin{aligned} \zeta_0 &= \pi_{k-1} \circ \varphi \\ \zeta_{p+1} &= \pi_{k-(p+2)} \circ \zeta_p. \end{aligned}$$

We see by induction that:

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

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

Example C.6. Consider the following figure:



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:

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

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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

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

3 Now we place $\psi(4) = f$ at the correct 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

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

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

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

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

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

Proof. The proof is by induction on the depth of G and the number of top-level nodes. □

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

Suppose $\widehat{\varphi} \neq \widehat{\pi_i \circ \varphi}$ and $\langle\!\langle \overline{\mathscr{B}} \rangle\!\rangle_{\varphi_{|\overline{\mathscr{B}}}} \simeq \langle\!\langle \overline{\mathscr{B}} \rangle\!\rangle_{\psi_{|\overline{\mathscr{B}}}}$ for all boxes $\mathscr{B}$. 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

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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 by induction on the depth of G. □

We can now prove Lemma 5.14.

Lemma C.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. The proof is immediate from Lemmas C.3, C.5 and C.10. □

Appendix D. Reduction in $\lambda{\to}C\mathsf{let}$ reflects reduction on deduction graphs

This section constitutes the proof of Lemma 5.15.

Lemma D.1. Let G be a deduction graph with free node (m, A) and 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 D.2. 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' into three pieces. The pieces of G are:

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

The pieces of G' are:

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

Using Lemmas A.10 and D.1, we see that

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

and

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

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

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

Lemma D.3. 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 $\mathscr{B}_q$ with box-node q that contains p is a maximal element of G. Then

$$\begin{aligned}\langle\!\langle G \rangle\!\rangle &= \langle\!\langle G \setminus q \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.\langle\!\langle \mathscr{B}^k_q \rangle\!\rangle[y])\, \mathsf{in}\, [-]] \\ &= \langle\!\langle G \setminus q, n \rangle\!\rangle[\mathsf{let}\, x_n = (\lambda x_l.\langle\!\langle \mathscr{B}^l_n \rangle\!\rangle[x_j])\, \mathsf{in}\, \mathsf{let}\, x_q = (\lambda x_k.\langle\!\langle \mathscr{B}^k_q \rangle\!\rangle[y])\, \mathsf{in}\, [-]]\,.\end{aligned}$$

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

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

Now with the help of Lemma A.10 we see that

$$\begin{aligned}\langle\!\langle G' \rangle\!\rangle &= \langle\!\langle G' \setminus q \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.\langle\!\langle \mathscr{C}^k_q \rangle\!\rangle[y])\, \mathsf{in}\, [-]] \\ &= \langle\!\langle G' \setminus q \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.(\mathsf{let}\, x_n = (\lambda x_l.\langle\!\langle \mathscr{B}^l_n \rangle\!\rangle[x_j])\, \mathsf{in}\, [-])[\langle\!\langle \mathscr{C}^k_q \setminus \mathscr{B}_n \rangle\!\rangle][y])\, \mathsf{in}\, [-]] \\ &= \langle\!\langle G \setminus q, n \rangle\!\rangle[\mathsf{let}\, x_q = (\lambda x_k.(\mathsf{let}\, x_n = (\lambda x_l.\langle\!\langle \mathscr{B}^l_n \rangle\!\rangle[x_j])\, \mathsf{in}\, \langle\!\langle \mathscr{B}^k_q \rangle\!\rangle[y])\, \mathsf{in}\, [-]]\,.\end{aligned}$$

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

Lemma D.4. If G' is obtained from G by an unsharing step, $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{CP} \langle\!\langle G' \rangle\!\rangle$.

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

— H', that is, 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 $\mathscr{B}$ with the nodes that are reachable from within $\mathscr{B}$ in one step, plus the free nodes of H';
— box $\mathscr{B}'$ with the nodes that are reachable from within $\mathscr{B}'$ in one step, plus the free nodes of $H' \cup \mathscr{B}$;
— $(G' \setminus H, \mathscr{B}', \mathscr{B})$, together with all free nodes of the previous $H' \cup \mathscr{B}' \cup \mathscr{B}$.

With Lemma A.10 we see that

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

We cut G into three pieces as follows:

— H, that is, 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 $\mathscr{B}$ with the nodes that are reachable from within $\mathscr{B}$ in one step, plus the free nodes of H;
— $(G \setminus H, \mathscr{B})$, together with the free nodes of $H \cup \mathscr{B}$.

With Lemmas A.10 and D.1 we see that

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

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

Lemma D.5. If G' is obtained from G by a repeat-elimination step, $\langle\!\langle G \rangle\!\rangle \twoheadrightarrow_{L} \langle\!\langle G' \rangle\!\rangle$.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

Proof. Suppose n_0 and n_1 are top-level nodes. By cutting G into three pieces and G' into either two or three, we easily see that $\langle\!\langle G \rangle\!\rangle \rightarrow\!\!\!\!\rightarrow_L \langle\!\langle G' \rangle\!\rangle$. Now assume that n_1 is in a box. Using the fact that none of the nodes to n_1 is an !-node, we can make another deduction graph G'' as follows. Place a node n_1' at top level. Replace the edges $n_1 \longrightarrow\!\!\!\!\triangleright\, n_0$ by $n_1' \longrightarrow\!\!\!\!\triangleright\, n_0$ 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, $\langle\!\langle G'' \rangle\!\rangle \rightarrow\!\!\!\!\rightarrow_L \langle\!\langle G' \rangle\!\rangle$. On the other hand, G 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 $x_j \neq x_{n_1'}$:

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

So $\langle\!\langle G \rangle\!\rangle \rightarrow\!\!\!\!\rightarrow_L \langle\!\langle G' \rangle\!\rangle$. The other cases follow by induction. □

Acknowledgments

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

We have used Peter Selinger's Fitch style natural deduction macros and Paul Taylor's proof tree and commutative diagram macros.

References

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

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

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

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

Geuvers, H. and Nederpelt, R. (2004) Rewriting for Fitch style natural deductions. In: van Oostrom, V. (ed.) Proceedings of RTA 2004. *Springer-Verlag Lecture Notes in Computer Science* **391** 134–154.

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

Klop, J. W. (1980) *Combinatory Reduction Systems*, Ph.D. thesis, University of Utrecht. (Mathematical Centre Tracts 127, CWI, Amsterdam.)

Kesner, D. and Lengrand, S. (2005) Extending the Explicit Substitution Paradigm. In: Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA), Nara, Japan. *Springer-Verlag Lecture Notes in Computer Science* **3467** 407–422.

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

Maraist, J., Odersky, M., Turner, D. N. and Wadler, P. (1998) Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. *Theoretical Computer Science* **228** (1-2) 175–210.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press

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

Nederpelt, R. P. (1973) *Strong normalization in a typed lambda calculus with lambda structured types*, Ph.D. thesis, Technical University Eindhoven.

Prawitz, D. (1965) *Natural Deduction*, Almquist and Wiksell, Stockholm.

Prawitz, D. (1971) Ideas and results in Proof Theory. In: Fenstad, J. E. (ed.) *Proc. of the second Scandinavian Logic Symposium* 235–307.

Sørensen, M. H. (1997) *Normalization in Lambda-Calculus and Type Theory*, Ph.D. thesis, Department of Computer Science, University of Copenhagen.

https://doi.org/10.1017/S0960129507006123 Published online by Cambridge University Press