From Deduction Graphs to Proof Nets: Boxes and Sharing in the Graphical Presentation of Deductions

Herman Geuvers, Iris Loeb

Radboud University Nijmegen, Institute for Computing and Information Sciences,
6500 GL Nijmegen, The Netherlands
{H.Geuvers, I.Loeb}@cs.ru.nl

Abstract. Deduction graphs [3] provide a formalism for natural deduction, where the deductions have the structure of acyclic directed graphs with boxes. The boxes are used to restrict the scope of local assumptions. Proof nets for multiplicative exponential linear logic (MELL) are also graphs with boxes, but in MELL the boxes have the purpose of controlling the modal operator !. In this paper we study the apparent correspondences between deduction graphs and proof nets, both by looking at the structure of the proofs themselves and at the process of cut-elimination defined on them. We give two translations from deduction graphs for minimal proposition logic to proof nets: a direct one, and a mapping via so-called context nets. Context nets are closer to natural deduction than proof nets, as they have both premises (on top of the net) and conclusions (at the bottom). Although the two translations give basically the same results, the translation via context nets provides a more abstract view and has the advantage that it follows the same inductive construction as the deduction graphs. The translations behave nicely with respect to cut-elimination.

1 Introduction

Deduction graphs [3] provide a formalism for natural deduction where the deductions have the structure of an *acyclic directed graph* with *boxes*. The main advantage of this formalism is the re-use of sub-proofs, which is simply done by putting several arrows to its conclusion. Because we do not see this as a logical step, we call this kind of sharing *implicit*. This makes deduction graphs a generalization of both Gentzen-Prawitz style natural deduction and Fitch style natural deduction. Because of the graph structure, one has to be explicit about local assumptions, so the boxes are used to limit the scope of an assumption.

In proof nets for multiplicative exponential linear logic (MELL) [4], the re-use of formulas is done by contraction links, and is therefore *explicit*. Proof nets also have boxes. Here they sequentialise the proof net in order to introduce the global modality "!" (of course).

There are some well-known translations from simply typed λ-terms to proof nets. There are three reasons why a translation from deduction graphs to proof

nets is more difficult. Firstly, parts of a deduction graph can be shared. We want a translation to somehow reflect this sharing. Simply typed λ-terms do not have sharing, except for the variables. Secondly, deduction graphs contain boxes. We would like a translation to associate a proof net box to a deduction graph box. Furthermore, the sharing and the boxes are both affected during the process of cut-elimination on deduction graphs. We want the cut-elimination on deduction graphs to behave similarly to cut-elimination on proof nets (on the translated deduction graphs). Thirdly, deduction graphs may have many conclusions, unlike simply typed terms, which have just one type. In this paper we present two translations from deduction graphs to proof nets, which mainly differ in the way they handle multiple conclusions.

We start by presenting *deduction graphs with explicit sharing*, SG, which is a variant of the usual definition of deduction graphs. The main difference is that in SG, sharing is handled as a logical step (Section 2). Furthermore we add a step, "loop", that allows to distinguish between conclusions and garbage, that is between formulas that we can use as the premises of next steps and formulas that will not be used any more.

The translation $(-)^*$ of Girard is used to translate formulas of minimal proposition logic to formulas of MELL (Section 3). This translation is the most suitable to our needs because it allows to nicely map deduction graph boxes to proof net boxes.

The direct translation (Section 4) uses *tensors* ($\otimes$) to connect the various conclusions. Just as most translations from λ-calculus to proof nets, assumptions Γ occur in the translation as terminal nodes $\Gamma^{*\perp}$, so as the *negation* of the translated formulas. An advantage of the direct translation is, that it only uses concepts that are already known from the theory of proof nets. A disadvantage is, that it works from conclusions to assumptions. So to translate a deduction graph, we must know that the construction of the graph has been finished. If we would decide to extend the graph later, we would have to translate the whole graph again.

Context nets (Section 5) form an extension of the concept of proof nets. They have terminal nodes, but they also permit so called *initial nodes*. We argue that there are several sensible translations from deduction graphs to proof nets, the translation of Section 4 being one of them, and that the translation to context nets generalises them by investigating the common parts of those proof nets. The translation from deduction graphs to context nets also reveals a symmetry between assumptions and conclusions: not only do the assumptions Γ associate to terminal nodes $\Gamma^{*\perp}$, but the conclusions Δ associate to initial nodes $\Delta^{*\perp}$ as well. Moreover, the translation can be done in the same order as the construction of the deduction graph, allowing to work with graphs that are not yet finished. In fact, the construction of the translation of a deduction graph looks like the construction of the original deduction graph "up side down".

In Section 6 it is shown that the translation via context nets after a slight modification, yields the same results as the direct translation. Section 7, finally, shows that the translations behave well with respect to cut-elimination.

2 Deduction Graphs with explicit sharing

In [3], we have defined the notion of *deduction graph* (DG), which is a generalization of both Gentzen-Prawitz style natural deduction and Fitch-style flag-deduction. A deduction graph is an acyclic directed graph with boxes satisfying some conditions. The boxes take care of local assumptions that can be *discharged*: a box restricts the scope of nodes and a box is a node itself. The nodes in the graph are labelled with formulas and if, e.g. node k is labelled with $A{\rightarrow}B$ and node m is labelled with A, there can be a node n labelled with B with one edge pointing to $(k, A{\rightarrow}B)$ and one pointing to (m, A). The general idea is that the inverse of the edges represents logical derivability: if from node n with label A there are edges to nodes $n_1, \ldots, n_k$ with labels $A_1, \ldots, A_k$ then A should be derivable from $A_1, \ldots, A_k$ using a logical rule. In [3], an inductive definition of the notion of *deduction graph for minimal propositional logic* is given. We do not repeat the definition here, but give as an example the left graph in Figure 1, which should also clarify the use of boxes.
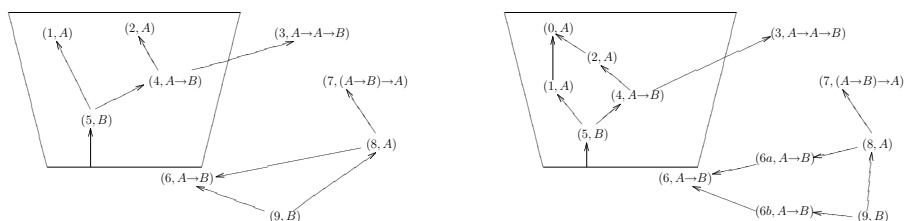
Fig. 1. Example of a deduction graph (left) and the same graph with an explicit sharing construction (right)

In the figure, node 6 is the node of the *box* containing nodes $1, 2, 4, 5$. The idea is that a set of nodes $\mathcal{B}$ (a box) can form a node again, which is called the *box node* of $\mathcal{B}$. So 6 is the box node of $\{1, 2, 4, 5\}$. The nodes that are not inside a box are on the *top level* and the top level nodes without incoming edges are called *free nodes*. So 3 and 7 are the free nodes in the example. Their labels $A{\rightarrow}A{\rightarrow}B$ and $(A{\rightarrow}B){\rightarrow}A$ correspond to the assumptions of the deduction. We can view this deduction graph as a deduction of B from $A{\rightarrow}A{\rightarrow}B$ and $(A{\rightarrow}B){\rightarrow}A$. The nodes 1 and 2 have no incoming edges but are inside a box. These are the local assumptions, which are discharged when forming the box. In deduction graphs it is not allowed to have edges pointing into a box, because this would correspond to the global use of a local assumption. Also overlapping boxes are not allowed (but boxes may be contained in each other). This is taken care of by the requirement that deduction graphs should be *closed box directed graphs*.

All definitions regarding deduction graphs are in [3], but because it is quite important in this paper we repeat the notion of closed box directed graph here:

3

Definition 1. *closed box directed graph is a triple* $\langle X, G, (\mathcal{B}_i)_{i\in I}\rangle$ *where G is a directed graph where all nodes have a* label *in X and $(\mathcal{B}_i)_{i\in I}$ is a collection of sets of nodes of G, the* boxes*. Each box $\mathcal{B}_i$ corresponds to a node, the* box node *of $\mathcal{B}_i$. Moreover, the boxes $(\mathcal{B}_i)_{i\in I}$ satisfy the following properties.*

1. *(Non-overlap) Two boxes are disjoint or one is contained in the other:* $\forall i, j \in I(\mathcal{B}_i \cap \mathcal{B}_j = \emptyset \vee \mathcal{B}_i \subset \mathcal{B}_j \vee \mathcal{B}_j \subset \mathcal{B}_i)$,
2. *(Box-node edge) There is only one outgoing edge from a box-node and that points into the box itself (i.e. 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.*

We first redefine the notion of deduction graph (of [3]) a bit, where we make the sharing explicit via a *sharing construction* (SG). If we have a *conclusion node* (i.e. a top-level node without incoming edges) (n, A), then we can add two nodes (m, A) and (k, A), which both have one outgoing edge to (n, A).

Definition 2. *The collection of* deduction graphs with explicit sharing *(SG) is the set of closed box directed graphs over* $\mathbb{N} \times \mathsf{Form}$ *inductively defined as follows.*

Axiom *A single node (n, A) is an SG,*

Join *If G and G' are disjoint SGs, then $G'' := G \cup G'$ is an SG.*

→-E *If G is an SG containing two conclusion nodes $(n, A{\to}B)$ and (m, A), then the graph $G' := G$ with*
- *a new node (p, B) at the top level*
- *an edge $(p, B) \longrightarrow (n, A{\to}B)$,*
- *an edge $(p, B) \longrightarrow (m, A)$,*

is an SG.

Repeat *If G is an SG containing a conclusion node (n, A), the graph $G' := G$ with*
- *a new node (m, A) at the top level,*
- *an edge $(m, A) \longrightarrow (n, A)$*

is an SG.

Share *If G is an SG containing a conclusion node (n, A), the graph $G' := G$ with*
- *two new nodes (m, A), (k, A) at top level,*
- *an edge $(m, A) \longrightarrow (n, A)$,*
- *an edge $(k, A) \longrightarrow (n, A)$*

is an SG.

→-I *If G is an SG containing a conclusion node (j, B) and a free node (m, A) then the graph $G' := G$ with*
- *A box $\mathcal{B}$ with box-node $(n, A{\to}B)$, containing the node (j, B) as the only conclusion node, and (m, A), and no other nodes that were free in G,*
- *An edge from the box node $(n, A{\to}B)$ to (j, B)*

is an SG under the proviso that it is a well-formed closed box directed graph.

Loop *If G is an SG containing the conclusions (m, A) and (n, B) $(m \neq n)$, then the graph $G := G'$ with an edge $m \longrightarrow m$ is an SG.*

Furthermore, in the construction one should always take care that, if $k \longrightarrow l$ and $k \in \mathcal{B}$ and $l \notin \mathcal{B}$, then k has not been constructed in a **Share***-step.*

So, apart from the four types of nodes A, I, E and R that we have already distinguished for deduction graphs in [3], we now also have S. A node is of a certain type, if it has been added in the corresponding construction rule of Definition 2. For example, a node is of type E if it has been added in a $\rightarrow$-E-step.

Definition 3. *A top-level node* (n, A) *such that* $n \twoheadrightarrow n$ *is called a* fake conclusion node.

It is easy to turn a deduction graph into a deduction graph with explicit sharing, simply by making the implicit sharing in DGs explicit via S-nodes. This is indicated in Figure 1. That the changes in sharing, looping and arrow introduction are not serious, is stated more precisely in the following lemma.

Lemma 4. *If* G *is an* DG *with free nodes* Γ *and conclusion nodes* Δ*, then there exists an* SG *G' with free nodes Γ and conclusion nodes Δ.*

We now modify the process of cut-elimination, because the deduction graphs with explicit sharing are not closed under the rules we defined for DGs. For example, we started the process by eliminating all repeats that separated the I-node of the cut from the E-node. Because in the deduction graphs with explicit sharing it is required that outgoing edges from S-nodes do not cross the border of any box, we now have to postpone the removal of some R-nodes. Therefore the process will transform the SG levelwise. The changes only affect the order in which we make the cut explicit. The elimination of the *safe cut* itself remains the same. A *safe cut* is the situation where we have an E-node where the edge to the $\rightarrow$-formula points to an I-node at the same level, so we have an $\rightarrow$-introduction followed immediately by an $\rightarrow$-elimination. In that case, the cut can be removed by a simple reordering of edges and the removal of some nodes. This is discussed in detail for DGs in [3]; the situation for SGs is slightly simpler.

Definition 5. *A cut in an* SG *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 *and* S *nodes* $(s_0, A{\rightarrow}B), \ldots, (s_i, A{\rightarrow}B)$,
- *Edges* $(p, B) \longrightarrow (s_i, A{\rightarrow}B) \longrightarrow \ldots \longrightarrow (s_0, A{\rightarrow}B) \longrightarrow (n, A{\rightarrow}B)$,
- *An edge* $(p, B) \longrightarrow (m, A)$.

The sequence $(p, B) \twoheadrightarrow (s_i, A{\rightarrow}B) \twoheadrightarrow \ldots \twoheadrightarrow (n, A{\rightarrow}B)$ *is called the* cut-sequence.

Definition 6 (Cut hidden by repeats). *Let G be an* SG *and let c be a cut of G and let S be that part of the cut-sequence that is at the same level as $(n, A{\rightarrow}B)$. If s_j is an* R*-node in S, for some $0 \le j \le i$, then the repeat-elimination at s_j is obtained by:*

- *When an edge points to s_j, redirect it to s_{j-1} (or to n, if $j = 0$);*
- *Remove s_j.*

Definition 7 (Cut hidden by sharing). *Let G be an* SG *with a cut c that contains a box $\mathcal{B}$ with box-node $(n, A{\to}C)$ and in-going edges from p_1, p_2. Then the* unsharing *of G at nodes n, p_1, p_2 is obtained by:*

- *removing $\mathcal{B}$,*
- *adding copies $\mathcal{B}'$ and $\mathcal{B}''$ of $\mathcal{B}$ including copies of the nodes reachable within one step,*
- *Connect the copies outside the boxes, to the original nodes. (thus if we had $q \longrightarrow\!\!\triangleright\, m$ with $q \in \mathcal{B}$, $m \notin \mathcal{B}$ and m' is the copy of m connected to $\mathcal{B}'$, and m'' is the copy of m connected to $\mathcal{B}''$, then we add the edges $m' \longrightarrow\!\!\triangleright\, m$ and $m'' \longrightarrow\!\!\triangleright\, m$),*
- *replacing n'' by p_1 and replacing n' by p_2.*

Figure 2 shows the unsharing of the SG in Figure 1.
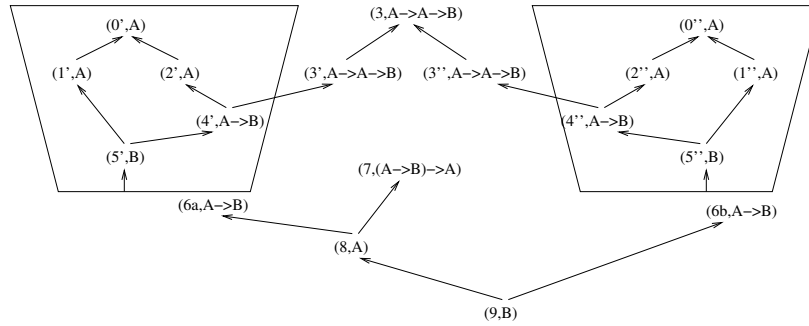
Fig. 2. Unsharing of the SG in Figure 1

Definition 8 (Cut hidden by a depth conflict; incorporation). *We have a* depth conflict *in the* SG *G with cut c, if $(n, A{\to}B)$ has one incoming edge, which comes from a node at another level. Let $\mathcal{B}$ the box of which $(n, A{\to}B)$ is the box-node. In that case the* incorporation *of G at c is obtained by putting $\mathcal{B}$ at one level deeper.*

Figure 3 shows an SG with a cut hidden by a depth conflict; Figure 4 shows its incorporation at $7, 12$.

Definition 9. *Given an* SG *G with a cut c, the process of eliminating the cut c is the following:*

1. *(Repeat-elimination) As often as possible, perform the repeat-elimination step as described in Definition 6.*
2. *(Unsharing) As often as possible, perform the unsharing step as described in Definition 7.*
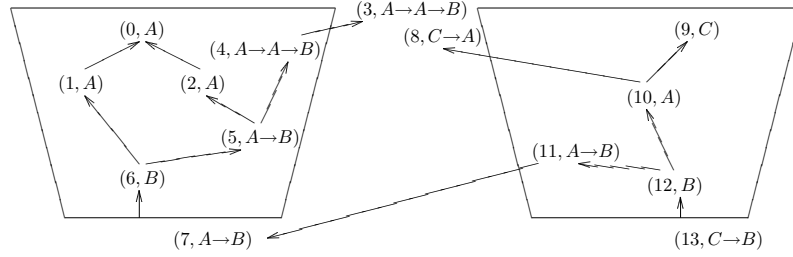
Fig. 3. SG with cut hidden by a depth conflict
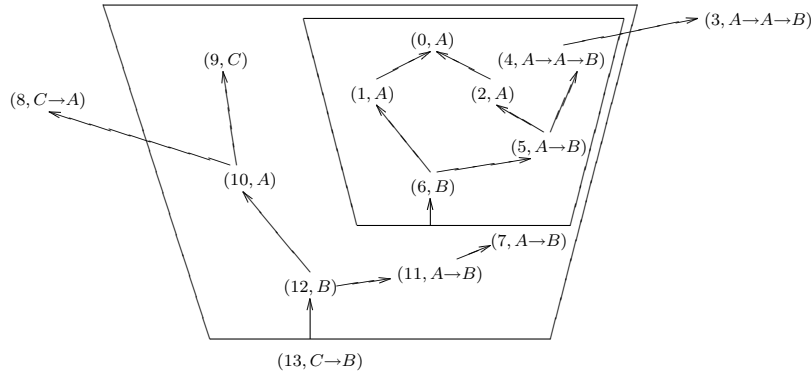
Fig. 4. Incorporation of the SG in Figure 3

3. *(Incorporation) If possible, perform the incorporation step as described in Definition 8.*
4. *(Moving up one level) If c is not yet safe, repeat the procedure, starting at 1.*
5. *(Eliminating the safe cut) Eliminate the safe cut.*

The proof of strong normalisation of the process of cut-elimination on DGs in [3] makes only use of the fact that it always ends in the elimination of a safe cut; the order in which we apply the other steps –Repeat-elimination, unsharing and incorporation– plays no role. As the process of cut-elimination on SGs still ends in the elimination of a safe cut, the proof goes through without much adjustments.

3 Translation of Formulas

To translate SGs to proof nets, we first have to translate the formulas of minimal propositional logic into linear logic formulas. This is done via the translation $(-)^*$ defined as follows.

$$A^* := \,!A \text{ for } A \text{ an atom}$$
$$(A \rightarrow B)^* := \,!(A^{*\perp} \invamp B^*)$$

7

Because we want to unify deduction graph boxes with proof net boxes, other translations seem less suitable.

4 Direct translation to Proof Nets

In the definition we will use the *rank* of a node: the conclusion nodes of an SG we give rank 0, and – roughly speaking – a node is given rank $i+1$ if there is an arrow to it from a node with rank i. We have to be careful in the case of sharing.

For $\mathcal{B}$ a box in an SG G the *closure of* $\mathcal{B}$, $\overline{\mathcal{B}}$ is the graph consisting of all nodes inside $\mathcal{B}$ plus all nodes that can be reached from within $\mathcal{B}$ in one step. $\overline{\mathcal{B}}$ is also an SG. (This follows from results in [3].)

Definition 10. *Given a deduction graph G, the rank of the nodes in G is defined as follows.*

- *If n is a conclusion node, $\mathsf{rank}(n) := 0$.*
- *If n is a fake conclusion node, $\mathsf{rank}(n) := 0$.*
- *If n is not an S-node or an I-node and $\mathsf{rank}(n) = i$ and $n \longrightarrow\!\!\!\triangleright m$, then $\mathsf{rank}(m) := i + 1$.*
- *If n is a S-node and $n \longrightarrow\!\!\!\triangleright m$, $k \longrightarrow\!\!\!\triangleright m$, $\mathsf{rank}(n) = i$ and $\mathsf{rank}(k) = j$, then $\mathsf{rank}(m) := \mathsf{max}(i, j) + 1$.*
- *If n is a I-node with box $\mathcal{B}$ and $\mathsf{rank}(n) = i$, then $\mathsf{rank}(j) := i + 1$ for all nodes $j \in \overline{\mathcal{B}}$.*

The rank of an SG G, $\mathsf{rank}(G)$, is the maximum of the rank of its nodes.

Lemma 11. *Let G be an SG with $\mathsf{rank}(G) = i + 1$ for some i.*

- *Suppose (n, B) of rank i is an E-node with edges to (m, A) and $(l, A{\rightarrow}B)$. Then $G \setminus m, l$ is an SG.*
- *Suppose $(n, A{\rightarrow}B)$ of rank i is an I-node of box $\mathcal{B}$. Then both $\overline{\mathcal{B}}$ and $G \setminus \overline{\mathcal{B}}$ are SGs.*
- *Suppose (m, A) of rank i is an S-node with an edge to (n, A). Then $G \setminus n$ is an SG.*
- *Suppose (n, A) of rank i is an R-node with an edge to (m, A). Then $G \setminus m$ is an SG.*

Definition 12. *Given an SG G we define a proof net (with labelled nodes) $V(G)$, which gives the proof net associated to G, by induction on the number of nodes of G.*
The invariant that we maintain is the following. If G has
– free nodes $(n_1, A_1), \ldots (n_k, A_k)$,
– conclusion nodes $(m_1, B_1), \ldots, (m_l, B_l)$,
then $V(G)$ has
– terminal nodes $B_1^* \otimes \ldots \otimes B_l^*$, and $A_1^{*\perp}, \ldots A_k^{*\perp}$, labelled $n_1, \ldots, n_k$ respectively.

To relieve the notational burden, we just write A instead of A^* in the proof nets. The definition of $V(G)$ is as follows. We make a case-distinction according

8

*to* $\mathsf{rank}(G)$.

Case $\mathsf{rank}(G) = 0$.

We only have to consider the conclusions $(n_1, A_1), \ldots (n_k, A_k)$ *of* G *and the fake conclusions* $(q_1, D_1), \ldots, (q_t, D_t)$. *Now* $V(G)$ *is the proof net containing axiom links* $A_s - A_s^\perp$ *(with* $A_s^\perp$ *labelled by* n_s *for* $1 \le s \le k$) *and the* $A_1, \ldots, A_n$ *nodes made into a tensor product* $A_1 \otimes \ldots \otimes A_n$ *by* $n-1$ *tensor links. Furthermore it contains weakening links on nodes* $D_s^\perp$ *(labelled* n_s) *for* $1 \le s \le t$.
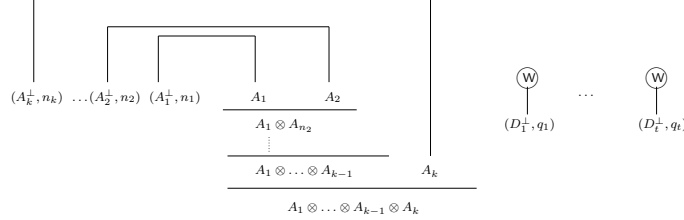
Case $\mathsf{rank}(G) = i + 1$ **for some** i.

Suppose that $V(F)$ *has already been defined for* SG*s* F *with less nodes than* G. *We consider the non-*A*-nodes of rank* i *in some order (say the box-topological ordering). We distinguish cases according to the type of the node.*

– E. *Suppose node* (n, B) *is of rank* i *with two edges to* (m, A) *and* $(l, A{\rightarrow}B)$. *So the nodes* m *and* l *are of rank* $i+1$. *Then:*
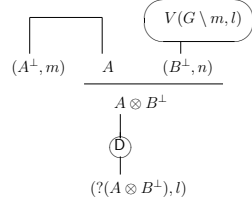
– I. *Suppose node* $(n, A{\rightarrow}B)$ *of rank* i *is a box node of box* $\mathcal{B}$ *with an edge to* (j, B) *and let* (m, A) *be the discharged node. Let* $(\vec{q}, \Gamma)$ *be the nodes that can be reached from within* $\mathcal{B}$ *with one edge. Then:*

9
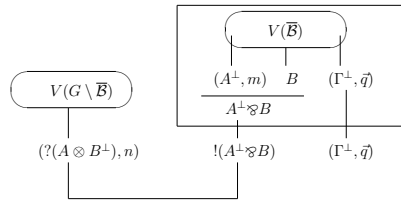
– S *Suppose node* (m, A) *of rank* i *is an* S-*node with an edge to* (n, A). *Let* (l, A) *be the other* S-*node with the same target. Then:*
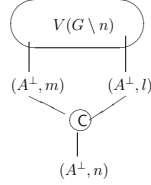
– R *Suppose* (n, A) *of rank* i *is an* R-*node with an edge to* (m, A). *Then:*

Example 13. Here we see an example of a simple SG G and the translation $V(G)$, written out completely.
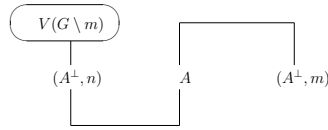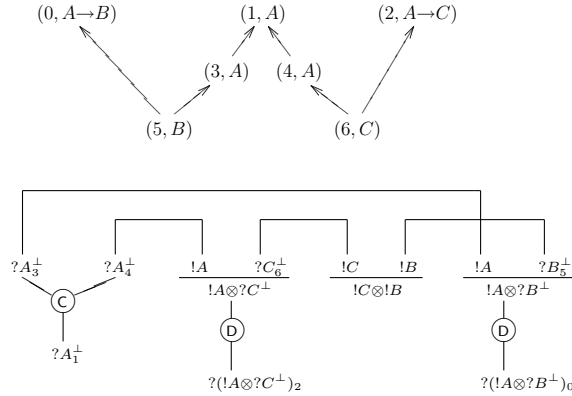

5 Translation via Context Nets

5.1 Context Structures, Context Nets, and Deduction Nets

In [3] we have made a translation that gives the simply typed λ-term associated to *a node* in a deduction graph. This solved the discrepancy that deduction graphs can have many conclusions, whereas simply typed λ-terms have just one type. Another solution could have been to make a translation from deduction graphs to typed λ-terms with conjunction types. So then we would associate a λ-term of type $B_0 \wedge B_1 \wedge \ldots \wedge B_k$ to a deduction graph with conclusions $(n_0, B_0), (n_1, B_1), \ldots, (n_k, B_k)$. In [3] we have not investigated this, but instead we have made a translation from deduction graphs to *contexts*, solving the problem of the many conclusions in a different way. Contexts do not have types

(they can only be "well-formed"), but they can obtain a type later, by filling in a variable associated to a node of a deduction graph.

The direct translation of Section 4, can be seen as the proof net equivalent of the translation from DGs to λ-terms with conjunction types. We can also define a translation to proof nets from SGs in a specific conclusion, as shown in Example 14. This can be seen as the proof net equivalent of the translation to simply typed terms.

Example 14. Here we see a translation of the SG in Example 13 in node 6. The translation of $(5, B)$ is weakened, and thus it plays the role of a fake conclusion.
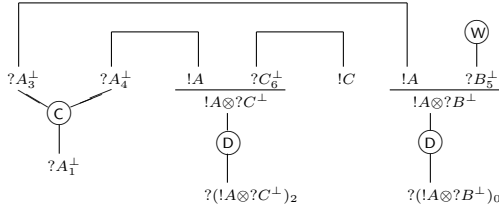
We do not go into this or other translations to proof nets, but we explore a translation to the net equivalent of contexts: context nets. Not only are context nets closer to deduction graphs as they have both "conclusions" and "assumptions", but this translation also generalises the translations to proof nets, because we can recover them by "gluing" something to the context net; the net equivalent of filling a hole (see Section 6).

We define *context structures*, an extension of Girard's proof structures [4]. It allows *initial nodes*, i.e. nodes that are not a conclusion. Similar extensions can be found in Danos and Regnier [1] and Puite [5]. Intuitively, this restores the duality found in two-sided sequents, as initial nodes of a context net can be seen as the formula occurrences left (as long as they are not inside a box) of the turn-style and the terminal nodes can be seen as the occurrences on the right of it. Another way to look at it, is to consider a context structure as an "open proof structure": if we connect proof structures with corresponding terminal nodes to the initial nodes of the context structure, we get a proof structure. In order to obtain structures that are subgraphs of proof structures, and in contrast to Puite, we do not expand the set of links. Another difference is that we consider an extension of proof nets of MELL (instead of MLL).

Definition 15 (Context Structures). *A* context structure *for MELL is a proof structure for MELL, with the difference that nodes that are not a conclusion, are allowed. These nodes are called* initial nodes.

Just as the construction rules of proof nets determine the subset of proof structures that are correct, which means that they agree with a sequent calculus proof, we would like to have construction rules for "context nets", which somehow should determine correct context structures. One could say that context structures are correct, when they agree with a two-sided sequent calculus proof. Another

notion of correctness is to say that, if a context net happens to be a proof structure, it is a proof net. It is this latter notion we adopt and we will give the construction rules, which are essentially the rules put forward by David and Kesner [2]. These rules are at the same time a bit rigid, as they consider only the terminal nodes, leaving the initial nodes untouched throughout the construction, and extremely useful, as the class of context structures singled out by these construction rules can easily be seen to be correct.

Definition 16 (Context Nets). Context nets *are inductively defined as follows.*
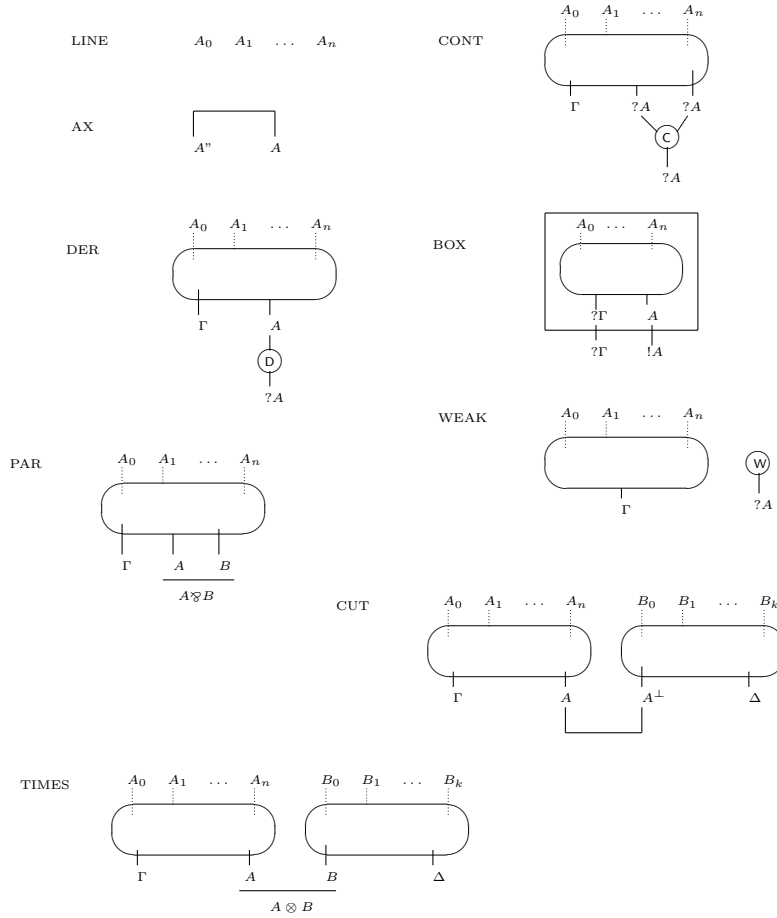
Lemma 17 (Correctness of Context Nets). *Every context net without initial nodes is a proof net.*

Proof. Immediate from the definition of context nets.

We now narrow our attention to a subclass of context nets, the *closed one-liners*. We could see a closed one-liner as a context net Θ such that, if we connect

every initial node of it to a corresponding terminal node of *one* proof net, say Σ, we get again a proof net.

Definition 18 (Closed one-liners). *A* closed one-liner *is a context net that has no initial nodes inside boxes and that can be formed by using the LINE-rule at most once.*

Definition 19 (Deduction Nets). Deduction nets *are inductively defined as follows :*

col — cl.one liner

box —
?Γ A
?Γ !A

point — A

ax — A $A^{\perp}$
Γ

der —
A
(D)
Δ ?A
Γ

cut —
Γ A $A^{\perp}$

weak —
(W)
Δ ?A
Γ

par —
Γ A B
$A \otimes B$

rep —
Δ A $A^{\perp}$ A
Γ

times —
Δ $A^{\perp}$ A B
$A \otimes B$
Γ

comb —
Δ_0
Δ_1 Σ Γ_0
Γ_1

cont —
?A ?A
(C)
Δ ?A
Γ
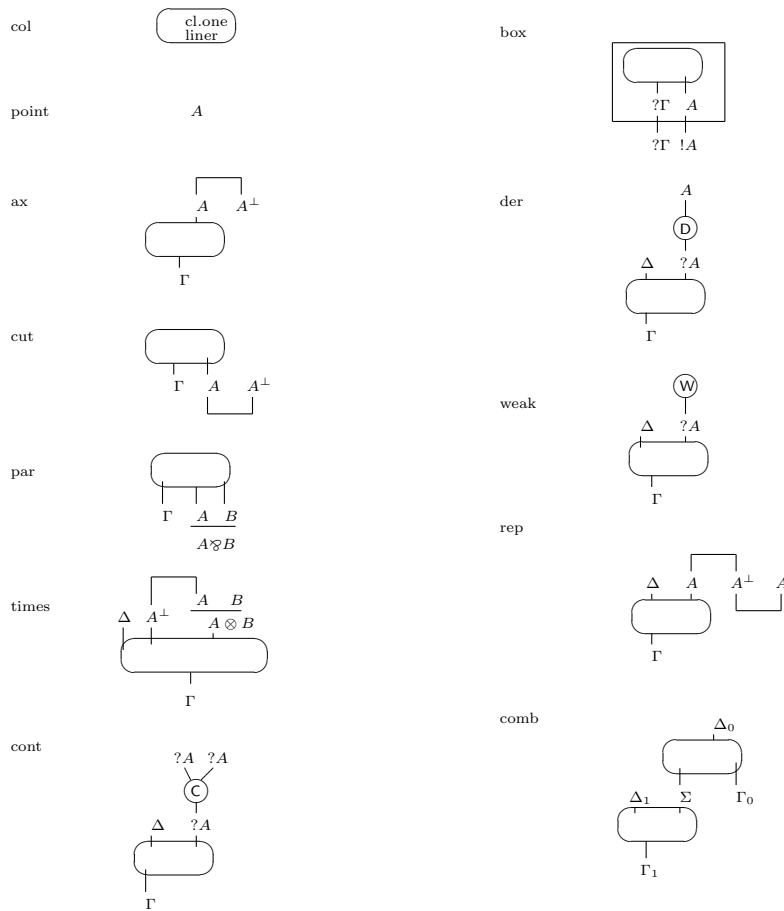
According to the col-rule, every closed one-liner is a deduction net. In the weak-rule, $\Delta \neq \emptyset$. For technical reasons, we only allow the comb-rule when $\Delta_0 = \emptyset$ implies $\Delta_1 = \emptyset$.

Lemma 20. *Every deduction net is a closed one-liner.*

13

Proof. By induction on the construction of deduction nets. We only treat the interesting cases.

ax By induction, we have a closed one-liner, say C with just one initial node, A. So it uses the LINE-rule with just this A. The new deduction net can be obtained by replacing this LINE-rule in the construction of C by an AX-rule.

cut By induction, we have a closed one-liner, say C, without any initial nodes. So a construction of C does not make use of the LINE-rule. We get the new deduction net by applying LINE on $A^\perp$ and doing a CUT with this on C.

times By induction, we have a closed one-liner, say C, which contains the initial nodes $A^\perp$ and $A \otimes B$. Let Δ be the initial nodes of C without $A^\perp$ and $A \otimes B$. The new deduction net can be obtained by replacing the LINE-rule in the construction of C, by the LINE-rule on Δ, B, an AX-rule and an TIMES-rule.

cont By induction, we have a closed one-liner, say C, with the initial nodes $\Delta, ?A$. Replace the LINE-rule on these nodes by a LINE-rule on $\Delta, ?A, ?A$, followed by a CONT-rule.

der By induction, we have a closed one-liner, say C, with the initial nodes $\Delta, ?A$. Replace the LINE-rule on these nodes by a LINE-rule on Δ, A, followed by a DER-rule.

weak By induction we have a closed one-liner, say C, with initial nodes $\Delta, ?A$, and terminal nodes Γ. Replace the LINE rule on $\Delta, ?A$, by a LINE-rule on Δ, followed by a WEAK-rule on $?A$.

rep By induction, we have a closed one-liner, say C, with initial nodes Δ, A. Let the LINE-rule of the construction of C be followed by the CUT-rule applied to an A and an $A^\perp$, created by an AX-rule.

comb By induction, we have two closed one-liners, say C_0, with initial nodes Δ_0 and terminal nodes Γ_0, Σ, and C_1, with initial nodes Δ_1, Σ and terminal nodes Γ_1. Then we can replace the LINE-rule of the construction of C_0 with a LINE-rule on Δ_0, Δ_1. Then we get a closed one-liner with terminal nodes $\Gamma_0, \Sigma, \Delta_1$. If we apply the construction of C_1 to this, we obtain the desired result.

Corollary 21. *The class of deduction nets is the class of closed one-liners.*

Theorem 22 (Correctness of Deduction Nets). *Every deduction net without initial nodes is a proof net.*

Proof. Immediate from the lemmas 20 and 17.

5.2 From Deduction Graphs to Context Nets

Definition 23. *Given a deduction graph G, we define a deduction net $I(G)$ associated to G by induction on the construction of G. The invariant we maintain is:* **If G is a deduction graph with assumptions Δ and conclusions Γ, then $I(G)$ is a deduction net with terminal nodes $\Delta^{*\perp}$ and initial nodes $\Gamma^{*\perp}$.** *We put a picture of G on the left and a picture of $I(G)$ on the right. Again, A stands for A^* and B stands for B^*.*

Axiom *The last step is an **Axiom** step:*

(n, A) $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ $(A^{\perp}, n)$

Join *The last step is a **Join** step, then by the comb-rule with $\Sigma = \emptyset$:*

| G' | | G'' | $\qquad\qquad$ | $I(G')$ | | $I(G'')$ |

Repeat *The last step is a **Repeat** step:*

$G \setminus m$

(n, A)

(m, A)

$(A^{\perp}, n)$ $\quad$ (A, k) $\quad$ $(A^{\perp}, m)$

$I(G \setminus m)$

Share *The last step is a **Share** step:*

$G \setminus m, n$

(k, A)

(m, A) $\quad$ (n, A)

$(A^{\perp}, m)$ $\quad$ $(A^{\perp}, n)$

C

$(A^{\perp}, k)$

$I(G \setminus m, n)$

→-E *The last step is a →-**E** step:*

$G \setminus n$

(p, A) $\qquad$ $(m, A \to B)$

(n, B)

(A, k) $\quad$ $(B^{\perp}, n)$

$(A \otimes B^{\perp}, l)$

D

$(A^{\perp}, p)$ $\qquad$ $(?(A \otimes B^{\perp}), m)$

$I(G \setminus n)$

→-I *The last step is a →-**I** step:*

(m, A) $\qquad$ $G \setminus B, n$

B

(j, B)

$(n, A \to B)$

$(B^{\perp}, j)$

$I(\overline{B})$

$(\Gamma^{\perp}, i)$ $\quad$ $(A^{\perp}, m)$ $\quad$ (B, k)

$(A^{\perp} \mathbin{⅋} B, p)$

$(\Gamma^{\perp}, i)$ $\quad$ $(!(A^{\perp} \mathbin{⅋} B), l)$ $\quad$ $(?(A \otimes B^{\perp}), n)$

$I(G \setminus B, n)$
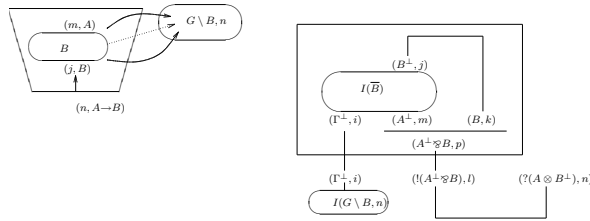
Loop *The last step is a **Loop** step:*

G

(n, A)

W

$(A^{\perp}, n)$

$I(G)$

Example 24. We show the proof net that we obtain via $I(-)$ from the example in Figure 1.

$$?(!A\otimes?B^\perp)_{6a} \quad \frac{!(?A^\perp\wp!B)\,?A_8^\perp}{!(?A^\perp\wp!B)\otimes?A^\perp} \quad \frac{!A \quad ?B_9^\perp}{!A\otimes?B^\perp}$$

$$?A_1^\perp \quad !A \quad ?B_5^\perp$$
$$!A\otimes?B^\perp$$

with circled D nodes and circled C nodes as shown, and the following conclusions:

$$?(!A\otimes?B^\perp)_6 \qquad !(?A^\perp\wp!B) \qquad ?(!A\otimes?(!A\otimes?B^\perp))_3$$
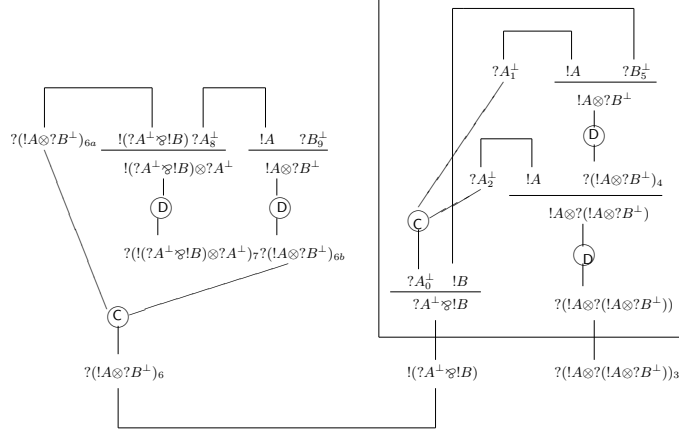
6 Connecting the two Translations

The translations $V(G)$ and $I(G)$ are different in their construction. If G is a deduction graph with assumptions Δ and conclusions Γ, then $I(G)$ is a deduction net with terminal nodes $\Delta^{*\perp}$ and initial nodes $\Gamma^{*\perp}$. By adding axiom links and tensor nodes, we turn this into a proof net with terminal nodes $\Delta^{*\perp}, \bigotimes\Gamma^*$, where $\bigotimes\Gamma^*$ is the formula obtained by putting a tensor between all formulas in Γ^*. This is indicated in the following figure.
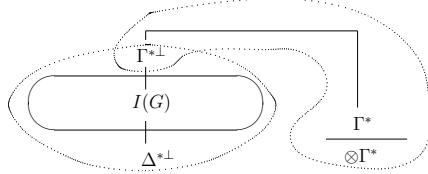
We call this $J(G)$. That this structure is a deduction net follows from the fact that the the two dashed parts are deduction nets and hence the whole structure is, using the comb rule. As it has no initial nodes, it is a proof net. We will see that J and V give the same results.

Lemma 25 (Gluing). *Let G be an* SG *with conclusions Σ,Γ and let F be an* SG *with assumptions Σ,Δ. Define $H := G \cup F$. Then*

1. $I(H) =$

2. $J(H) =$

(diagram: node $J(F)$ connected via $\Gamma^{*\perp}$, $\Sigma^{*\perp}$, $\Delta^{*\perp}$ to node $I(G)$)

Proof. By induction on the number of non A-nodes of F.

Theorem 26. *Let G be an* SG. *Then $V(G) = J(G)$.*

Proof. The proof is by induction on the number of nodes of G. We make a case-distinction to rank(G).
Case rank$(G) = 0$.
Then G has been constructed using **Axiom**, **Join**, and **Loop**. We easily verify that $V(G) = J(G)$.
Case rank$(G) = i+1$ for some i.
Suppose that $V(F)$ has already been defined for SGs F with less nodes than G. We consider the non-A-nodes in some order (say the box-topological ordering). We distinguish cases according to the type of the node. We treat here the E-case and the I-case.

- E Suppose node (n, B) is of rank i with an edge to (m, A) and an edge to $(l, A{\rightarrow}B)$. Let F be the graph consisting of the nodes n, m, and l and the edges $n \longrightarrow m$ and $n \longrightarrow l$. Then $V(G) =$

  (left diagram: box $V(G \setminus m, l)$ over nodes $(A^\perp, m)$, A, $(B^\perp, n)$; $A \otimes B^\perp$; node D; $(?(A \otimes B^\perp), l)$) $=_{IH}$ (right diagram: box $J(G \setminus m, l)$ over nodes $(A^\perp, m)$, A, $(B^\perp, n)$; $A \otimes B^\perp$; node D; $(?(A \otimes B^\perp), l)$)

  Note that this is $J(G \setminus m, l)$ glued to $I(F)$ and by Lemma 25 we are done.

- I Suppose node $(n, A{\rightarrow}B)$ of rank i is a box-node of box $\mathcal{B}$ with an edge to (j, B) and let (m, A) be the discharged node. Let $(\vec{q}, \Gamma)$ be the nodes that can be reached from $\mathcal{B}$ with one edge.

  (diagram: box $V(\overline{\mathcal{B}})$ over nodes $(A^\perp, m)$, B, $(\Gamma^\perp, \vec{q})$; $A^\perp \mathbin{⅋} B$; box $V(G \setminus \overline{\mathcal{B}})$; bottom nodes $(?(A \otimes B^\perp), n)$, $!(A^\perp \mathbin{⅋} B)$, $(\Gamma^\perp, \vec{q})$)

  Let F be the graph consisting of box $\mathcal{B}$ with box-node n and all nodes that are reachable from $\mathcal{B}$ within one step. By induction we conclude that $V(G)$ is $J(G \setminus \overline{\mathcal{B}})$ glued to $I(F)$. By Lemma 25 we are done.

17

7 Preservation of Reduction

We will show that the transformations involved in the process of cut-elimination of SGs can be mimicked by reductions on their context net translations. Of course this implies that it can be mimicked by reductions on the result of the direct translation of Section 4 too, and indeed on any translation that is an extension of the translation via context nets.

The notion of reduction on proof structures can be extended to reduction on context structures without any problem. We will use the following reduction rules: Ax-cut, $\wp$-$\otimes$, d-b, c-b and b-b. (See [4] and [3].)

Theorem 27. *The class of deduction nets is closed under reduction.*

Proof. Note that we can make a proof net from any given deduction net by the comb-rule. Because all reduction rules preserve initial and terminal nodes, it now follows that the class of deduction nets is closed under reduction.

Theorem 28. *If G' is obtained from G by either elimination of a safe cut, an unsharing step, a repeat-elimination step, or an incorporation step, then $I(G')$ can be obtained from $I(G)$ by one or more reductions of context nets.*

Proof. – If G' is obtained from G by eliminating a safe cut, then there exists a deduction net F such that $I(G) \twoheadrightarrow_{d-b} F \twoheadrightarrow_{\wp-\otimes} I(G'')$.
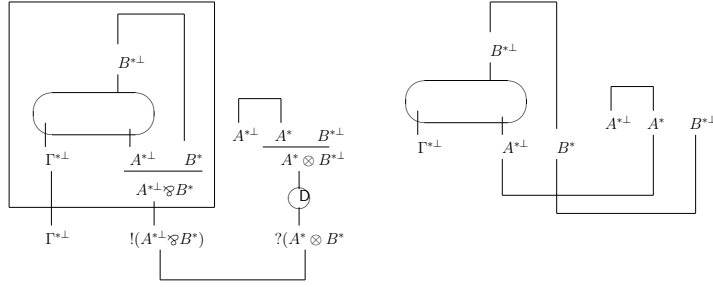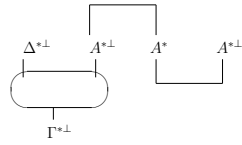

– If G' is obtained from G by unsharing, then $I(G) \twoheadrightarrow_{c-b} I(G')$.
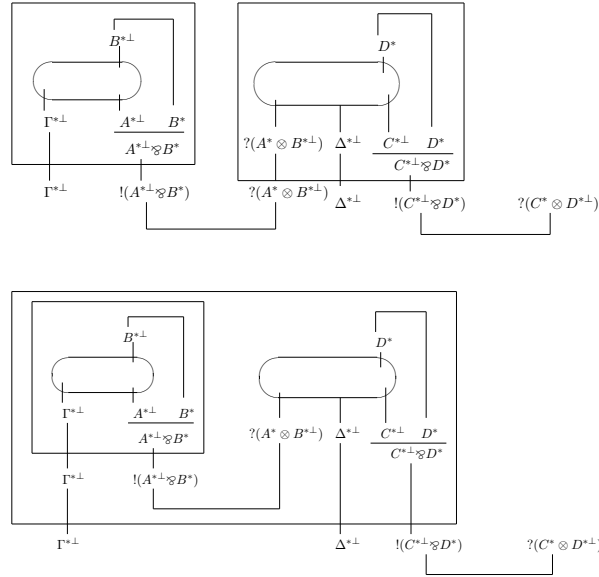

18
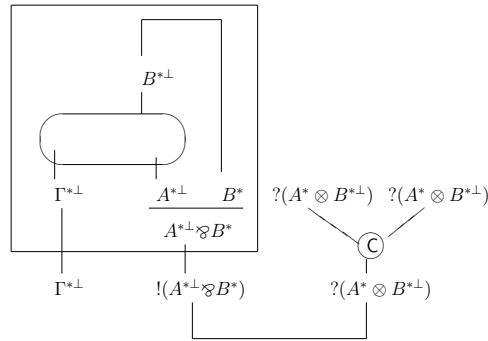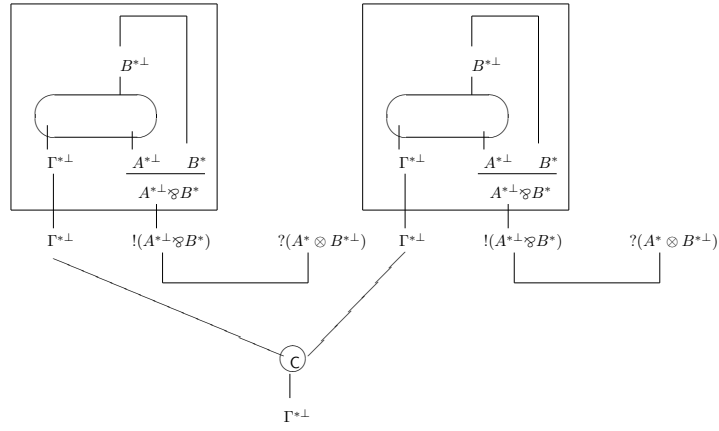
– If G' is obtained from G by a repeat elimination, then $I(G) \twoheadrightarrow_{Ax-cut} I(G')$.

– If G' is obtained from G by incorporation, then $I(G) \twoheadrightarrow_{b-b} I(G')$.

From this correspondence of reduction follows again strong normalisation for the process of cut-elimination of SGs. However, the result is stronger than the

one mentioned in Section 2. There we assumed the total removal of a cut before starting the process of cut-elimination for another cut. Moreover, the process of cut-elimination prescribes an order of the various steps. Seeing these steps of the process as separate reduction steps, one could say that the process describes a reduction *strategy* and the normalisation result is rather weak normalisation.

But now we see that the steps may be done in any order, even mixing the steps for various cuts. This gives us strong normalisation in a very general sense.

Acknowledgments

References

1. V.Danos and L.Regnier, *The structure of the multiplicatives*, Archive for Mathematical logic 28, 1989.
2. R.David and D.Kesner, *An arithmetical strong-normalisation proof for reduction modulo in proof-nets*, draft.
3. H.Geuvers and I.Loeb, *Natural Deduction via Graphs: Formal Definition and Computation Rules*, to appear in MSCS (http://www.cs.ru.nl/~herman/PUBS/gd.pdf)
4. J.-Y. Girard, Linear Logic, *Theoretical Computer Science*, 50(1):1-101, 1987.
5. Q.Puite, *Proof Nets with Explicit Negation for Multiplicative Linear Logic*, Preprint 1079, Department of Mathematics, Utrecht University, 1998.