Non deterministic Linear logic: application to Boolean circuits

Virgile Mogbil

Non deterministic Linear logic: application to Boolean circuits

Virgile Mogbil

LIPN – UMR7030, CNRS – Université Paris 13,
99 av. J-B Clément, F–93430 Villetaneuse, France

Abstract. Subsystems of Linear logic (LL) are used to give Curry-Howard characterizations of complexity classes as P. By expressing the non-determinism by an explicit rule to sum up, one characterizes NP [Mau03]. Following the Curry-Howard isomorphism but for parallel model of computation we study the proof nets of the non-deterministic multiplicative LL. Considering $NNC(poly)$ i.e. NC (the efficiently parallelizable functions) with a polynomial amount of non-deterministic inputs, we define $nmBN(poly)$ the uniform families of multiplicative Boolean proof nets with polynomial amount of explicit non-determinism. Finally $nmBN(poly)$ is a Curry-Howard characterization of the complexity class $NP = NNC(poly)$.

1 Introduction

The *proof nets* [Gir87,DR89] of the Linear logic (LL) are a parallel syntax for logical proofs without all the bureaucracy of the sequent calculus. Their study is also motivated by the well known Curry-Howard isomorphism: there is a correspondence between proofs and programs which associates cut-elimination in proofs and execution in programs. Proof nets were used in subsystems of LL to give Curry-Howard characterizations of complexity classes. Usually this is done in LL by reducing the deductive power of the exponentials, also known as modalities, which are in charge of controlling duplication in the cut-elimination process. The most known restrictions characterize P. One of them, the Intuitionistic Light Affine Logic (ILAL), just to name it, corresponds to P. By expressing the non-determinism by an explicit rule to sum up, the non deterministic extension of ILAL characterizes quite naturally NP [Mau03]. This *sum rule* is a logical counterpart to non-deterministic choice in process calculi. From proof nets other characterizations were given by correspondence with models of parallel computation. We do the same with explicit non-determinism.

Boolean circuits (see [Vol99,BS90] for instance) are a standard models of parallel computation. Several important complexity classes are defined in terms of Boolean circuits. E.g. NC can be thought of as the problems that can be efficiently solved on a parallel computer just as the class P can be thought of as the tractable problems. Because a circuit has a fixed input size, an infinite family of circuits is needed to do computations on arbitrary inputs. With a *uniformity*

condition on each circuit, a family can be regarded as an implementation of an algorithm. The circuit depth is the time on a parallel computer where the size is the number of processors. For instance NC is the set of Boolean functions computable by uniform Boolean circuits of polynomial size and polylogarithmic depth.

By restricting the proved formulae and with a logical depth notion, there is a proofs-as-programs correspondence between the proof nets and NC s.t. it preserves both the size and the depth of the models [Ter04] (and [MR06] for uniformity preservation). We use the same tools for proof nets in a uniform setting: the logical depth, an higher order gate simulation and small proof nets for duplication managing arbitrary fan-out simulation, for conditional i.e. if-then-else and for composition of proof nets. We give a proof-as-programs correspondence with $NNC(k(n))$, the class defined from NC circuits with $O(k(n))$ non-deterministic variables. $NNC(polylog)$ was introduced as a candidate for a class separating $NC = NNC(log\ n)$ and $NP = NNC(poly)$ by showing that it contains an algorithm for the quasigroup isomorphism problem not known to be in P or be NP-complete [Wol94]. Another motivation to obtain a characterization of NP comes from the separation with P which holds if NP is not a subset of $P/poly$ (the non-uniform NC circuits without depth restriction). $P/poly$ is characterized by non-uniform Boolean proof net families [Ter04] from which our study is a variation. So the Curry-Howard isomorphism for parallel model of computation gives us new tools for studying theoretical implicit complexity. The approach of Boolean proof nets is more than just a circuit reformulation because for e.g. it allows higher order gates and the proof net cut-elimination is a proper method.

In the section 2 we present MLL_u, the multiplicative LL with arbitrary arity, introduced to simplify relationship with the unbounded fan-in circuits [Ter04]. We give its non-deterministic extension $nMLL_u$ and consider proof nets for it. We recall the reduction steps of the cut-elimination. We define several size and depth notions used in the proofs, all are natural graph theoretic notions. In the section 3 we mostly study the cut elimination from a parallel point of view. We give the central theorems which allow us to establish the results on the complexity classes. In the section 4 we recall the Boolean circuit definitions and properties. It includes, the uniformity both of proof nets and circuits, the hierarchy of NC and $NNC()$. We define the Boolean proof nets of $nMLL_u$ i.e with sum-boxes which generalize the ones of MLL_u. In the section 5 we apply the previous theorems to Boolean proof nets with sum-boxes and we establish a proofs-as-programs correspondence with $NNC()$ circuits. I.e. the translation and simulation theorems preserve both size and depth of the models. Finally in section 6 we summarize the obtained results via $nmBN()$, a hierarchy of proof net complexity classes defined analogously to the $NNC()$ hierarchy. The classes $nmBN(poly)$, $nmBN(polylog)$ and $nmBN(1)$ of uniform Boolean proof net families with respectively $n^{O(1)}$, $log^{O(1)}\ n$ and $O(1)$ sum-boxes are respectively equal to NP, $NNC(polylog)$ and NC.

2 Non deterministic Linear logic

– Formulae, Sequent calculus and cut-elimination –

The *formulae* of MLL_u and $nMLL_u$ are built on literals by multiplicative conjunction and disjunction: the usual multiplicative connectives $\otimes$ and $\parr$ but with unbounded arities. There are no differences with binary connectives as the usual fragments of linear logic except that this gives us depth-efficient proofs [DR89]. The negation of a non-literal formula is *defined* by de Morgan's duality: $(\otimes^n(\overrightarrow{A}))^\perp = \parr(\overleftarrow{A}^\perp)$ and $(\parr^n(\overrightarrow{A}))^\perp = \otimes(\overleftarrow{A}^\perp)$ for $\overrightarrow{A} \equiv A_1, \ldots, A_n$ and $\overleftarrow{A} \equiv A_n, \ldots, A_1$. The negation applies to the sequences of formulae as we can expect it.

The *sequents* of $nMLL_u$ are of the form $\vdash \Gamma$, where Γ is a multiset of formulae. The rules are described in Fig.1. As it is usual in Linear sequent calculus, MLL_u and $nMLL_u$ admit the cut-elimination theorem (Hauptsatz) and implicit exchanges in sequents. We recall in the Fig.2 the sum-rule reduction step [Mau03].

(a) $\dfrac{\vdash \Gamma, C \quad \vdash \Delta, C^\perp}{\vdash \Gamma, \Delta}\ cut$ $\qquad$ $\dfrac{}{\vdash A, A^\perp}\ ax$

$\dfrac{\vdash \Gamma_1, A_1 \quad \cdots \quad \vdash \Gamma_n, A_n}{\vdash \Gamma_1, \ldots, \Gamma_n, \otimes^n(\overrightarrow{A})}\ \otimes^n$ $\qquad$ $\dfrac{\vdash \Gamma, A_n, \ldots, A_1}{\vdash \Gamma, \parr^n(\overleftarrow{A})}\ \parr^n$

(b) $\dfrac{\vdash \Gamma \quad \cdots \quad \vdash \Gamma}{\vdash \Gamma}\ sum$

Fig. 1. Sequent calculuses: (a) MLL_u and (a+b) $nMLL_u$

$$\dfrac{\dfrac{\vdash \Gamma, A \quad \cdots \quad \vdash \Gamma, A}{\vdash \Gamma, A}\ sum \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ cut \longrightarrow \dfrac{\dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ cut \quad \cdots \quad \dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ cut}{\vdash \Gamma, \Delta}\ sum$$

Fig. 2. $nMLL_u$ sum-rule reduction step

– Proof nets and cut-elimination –

We suppose the reader is a little bit familiar with the proof nets of the Linear logic and more specially with the cut-elimination [Gir96].

A *proof net* [Gir87,DR89] is a kind of graph that just keep the structure of proofs without what is irrelevant for computation. It is a set of interconnected links that we build by inference from the rules of the sequent calculus. There are several *sorts* of links: the *ax*-link, the $\otimes$-link and the $\parr$-link corresponding

respectively to the MLL_u rules. Every link has several ports numbered by convention as in the Fig. 3, so we can abusively omit them. We note with zero the conclusion of a link which is also called *principal port*. We represent the cut-rule by connecting two principal ports rather than with a link.

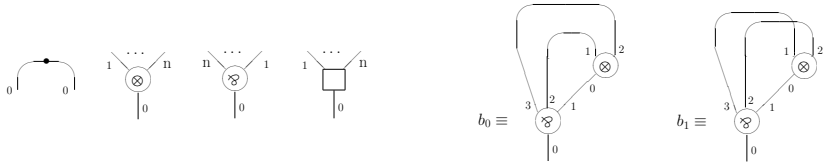

Fig. 3. *ax*-link, $\otimes$-link, $\parr$-link and $\square$-link ; small proof nets

For the sum-rule we use one box, so-called *sum-box* (Fig. 4), delimiting a part of the graph such that we associate one $\square$-link for each common conclusion of every sub-net. In a proof net a *summand* is a proof net obtained by erasing all but one sub-net in every sum-box. So in a proof net of $nMLL_u$ every summand is a proof net of MLL_u.

We give a description of a proof net by a finite set L of link, a function $\sigma : L \to \{\bullet, \square\} \cup \{\otimes^n, \parr^n\}_{n \geqslant 1}$, a symetric relation on $L \times \mathbb{N}$ and a function $\tau : L \to \mathbb{N}$ for sum-boxes (with indices). We only consider a proof net to be a description inferred from sequent calculus or equivalently inductively build from the *ax*-links with the constructors of Fig. 4. In the section 4 we shall give a logspace description which extend this one.

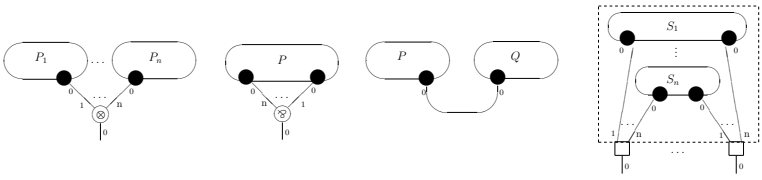

Fig. 4. Proof net constructors: $\otimes$-link, $\parr$-link, cut and sum-box

The reduction rules for MLL_u proof nets are the following one respectively called *ax*-reduction and *m*-reduction: $\forall\ 0 \leqslant i \leqslant n$

$$cut(ax(A, A^\perp), A) \ \to_{ax} \ A \quad \text{and} \quad cut(\otimes^n(\overrightarrow{A}), \parr^n(\overleftarrow{A}^\perp)) \ \to_m \ cut(A_i, A_i^\perp)$$

We give the reduction rules for the sum-boxes in Fig.5 [Mau03]: the *merge-rule* (associativity), the *down-rule* (linearity) and the down'-rule (selection) apply for an arbitrary context C and $i_0 \in I$. The down'-rule introduced in [Mau03] is redundant and useful in a sequential setting to reach a polynomial bound of reduction.

$$sum(R_0, sum_{i\in I}(R_i)) \rightarrow_{merge} sum_{i\in I\cup\{0\}}(R_i)$$

$$sum(D, C[sum_{i\in I}(R_i)]) \rightarrow_{down} sum(D, sum_{i\in I}(C[R_i]))$$

$$sum(D, C[sum_{i\in I}(R_i)]) \rightarrow_{down'} sum(C[R_{i_0}], sum(D, C[sum_{i\in I-\{i_0\}}(R_i)]))$$

Fig. 5. Sum-box reduction rules

We define various concepts of depth in a natural way: box-depth w.r.t.[1] the sum-boxes, link-depth and logical depth w.r.t. the cuts. The *box-depth* $b(l)$ of a link l is the number of sum-boxes that encapsulate it. We will say in the same way for the box-depth of a box. The box-depth $b(P)$ of a proof net P is maximal box-depth of its links. The depth or *link-depth* $d(l)$ of a link l is $1 + max\{d(p)|p \text{ premise of } l\}$. The link-depth $d(P)$ of a proof net P is the maximal link-depth of its links. The *logical depth* $\underline{c}(P)$ of a proof net P is the maximal link-depth of its cuts. We denote $c(P)$ the logical depth without counting the $\square$-links. The link-depth and the logical depth can be decomposed by the box-depths by thinking that boxes form layers. That gives partial depths denoted by indices for the box-depths as follows: $d(l) = \Sigma_{0\leqslant x\leqslant b(P)} d_x(l)$

We also define a size of proof nets and a partial size w.r.t. the box-depth. The *size* $|P|$ is the number of links in P where a box (and not the $\square$-links) is counted as 1. The partial sizes $|P|_x$ is the size of P restricted to the box-depth x: $|P| = \Sigma_{0\leqslant x\leqslant b(P)} |P|_x$.

3 Parallel cut-elimination

A critical pair arises when the reduction rules overlap to give two different proof nets. Convergence of all critical pairs insures the weak confluence but not the parallelization of the reduction. Indeed critical pairs are the conflict cases to apply reductions in parallel. It is already the case in MLL: one introduces a new reduction rule called tightening reduction [Ter04]. Next we study the parallel sum-box reductions. Notice however that every critical pair in MLL_u (only ax-reductions) are convergent then MLL_u is confluent and a cut between two sum-boxes is a convergent critical pair then $nMLL_u$ is confluent.

– Tightening reduction –

A cut between two axioms cannot be eliminated in parallel: one eliminates each maximum alternating chain of cuts/axioms in only one global step. Such step is called a tightening reduction step (t-reduction) and is denoted $\rightarrow_t$. After that there is no critical pairs in MLL_u. We denote by $\Rightarrow_t$ the parallel $\rightarrow_t$ reductions and $\Rightarrow_m$ the parallel $\rightarrow_m$ reductions. In the rest of the paper we denote $\Rightarrow$ for one parallel reduction step in MLL_u (i.e. $\Rightarrow_t$ or $\Rightarrow_m$).

[1] With relation to

– Parallel reductions of merged sum-boxes –

Critical pairs of merge rules in distinct summands of the same sum-box are not a conflict case. We can apply merge rules in parallel on them. But we cannot merge in a summand that would be merged at the same time. Because the merge-rule is confluent one can easily consider a global rewriting rule by merging sum-boxes as expected. We consider the maximal sub-graphs of a proof-net which are composed only of successive sum-boxes. Each sub-graph is a tree which one can reduced in only one step that we denote $\rightarrow_M$. We symbolically describe the corresponding rule in Fig.6 by noting only the links of one sum-box by a simple square

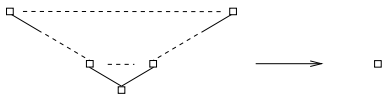

Fig. 6. $\rightarrow_M$ is the global reduction by merging

– The fixed box-depth reduction of the sum-boxes –

A cut between two sum-boxes is a critical pair for the down reduction rules but is convergent. The common reduct is a sum-box whose summands are the combination of all the summands of one sum-box with the summands of the other. However to apply in parallel the rules to this critical pair causes a conflict.

We choose a first strategy by reducing the cuts of depth zero until they become of depth one, then starting again with the following depth. For that one chooses the maximum entanglement of cut sum-boxes (as for the chains of cuts/axioms). The rule used, denoted $\rightarrow_{\mathit{fbd}}$, is described in Fig.7: for the shake of simplicity in the figure we omit the cut-free conclusions of sum-boxes in the redex and in the reduct. The result is a sum-box whose summands are all the combinations of the summands of distinct sum-boxes. Because there is no cycle of such entangled sum-boxes, each maximal entanglement is just a (rootless) tree of cut sum-boxes of same depth.

Thus we reduce in the same time all the cuts between sum-boxes of fixed depth. But it remains to reduce cuts between $\Box$-link and another link.That duplicates a part of the cut structures and in the bad case that increases the depth of the proof net ! However to reduce such cuts, a fine study of the partial sizes shows that we obtain the same result if one reduces them at the same time as $\rightarrow_{\mathit{fbd}}$ or later. Indeed it does not improve the bound on the step number to eliminate cuts at given depth. We thus generalize the $\rightarrow_{\mathit{fbd}}$ reduction rule to the case of the cuts where one premise is a sum-box s.t. the duplication is done only for the links of the fixed depth.

This strategy is so-called the fixed box-depth reduction of the sum-boxes.

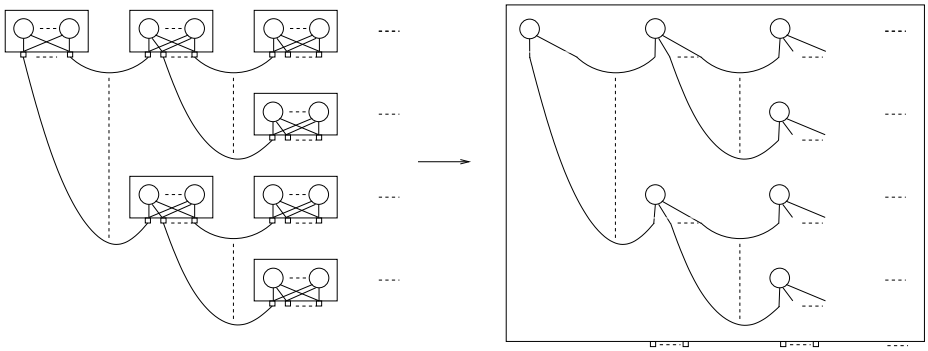

Fig. 7. $\rightarrow_{\mathit{fbd}}$ is the global fixed box-depth reduction

Let $\Rightarrow_{\mathit{fbd}_x}$ one parallel reduction step composed by all the possible $\rightarrow_{\mathit{fbd}}$ reductions at fixed box-depth x.

Lemma 1. *Let $P \in nMLL_u$ and let x be the minimal box-depth of the cuts in P. If $P \Rightarrow_x^* P' \Rightarrow_{\mathit{fbd}_x} P''$ then we have:*

- *Every cut $c \in P'$ of box-depth x is a sum-box cut, and the box-depth of c strictly increases in P'',*
- *There is no more a cut of depth x, and $b(P) = b(P') = b(P'')$.*

Proof. Minimality of x implies that after the cloture of $\Rightarrow_x$ every cut c is a sum-box cut (and $b(P) = b(P')$). By definition the $\Rightarrow_{\mathit{fbd}_x}$ reduces such a cut possibly with a lot of duplications according to whether the cut is a $\Box/\Box$-cut or not. If c is a $\Box/\Box$-cut then $\Rightarrow_{\mathit{fbd}_x}$ reduces c s.t. the link-depth of c strictly decreases in P'' without it duplicates other links of depth x. Else because combinations are made only of links at box-depth least $x+1$ and that the duplicated links are only of box-depth x, the box-depth of P' does not increase by $\Rightarrow_{\mathit{fbd}_x}$. □

Theorem 1. *There is a sequence of $O(c(P).b(P))$ parallel reductions which reduces a $nMLL_u$ proof net P in a cut-free one. In the worst case such sequence is of length $O(|P|^2)$.*

Proof. From the box-depth 0, we apply $\Rightarrow_0^*; \Rightarrow_{\mathit{fbd}_0}$ the parallel reduction sequence of lemma 1. We can apply each parallel reductions $\Rightarrow_x^*$ at most $c(P)$ time. The global sequence of reductions strictly decreases the box-depth, then we apply it $b(P)$ time. The duplicated multiplicative links are reduced in each summand in parallel in some next sequence of $\Rightarrow_x^*$. (Eventually at the end of the procedure one needs to apply one more time the $\Rightarrow_x^*$ reductions for the last multiplicative connectives) In the worst case we have $b(P) = O(|P|) = c(P)$. □

Unfortunately an example of worst case is a Boolean proof net of $nMLL_u$. Remark that we can use $\Rightarrow_{Mx}$ at the end of every round but it can decrease the box-depth of P: the next round will be made without changing box-depth.

– The parallel down reduction –

We analyze the down rule $\rightarrow_{down}$ to apply it in parallel. According to the context C, the pattern $p = sum(C[sum_{i\in I}(A_i)], D)$ can be a critical pair for the down rule. For example if $C[-] = cut(sum_{j\in J}(B_j), [-])$ then p reduces to $sum(sum_{i\in I, j\in J}(cut(A_i, B_j)), D)$ in two $\rightarrow_{down}$ steps. The same holds for a more general context C but remains convergent. As done in the previous subsection, we consider a general rule to apply the down reductions in parallel in one sum-box B.

Let B a sum-box and let B_k the sum-boxes in just one summand of B where $b(B_k) = b(B) + 1$. We reduce in one parallel reduction step this summand: the resulting summand is the combination of all the contents of the B_k and the context outside the B_k. Then the sum-boxes go down and contexts rise. This is a kind of generalization of the $\rightarrow_{fbd}$ rule to every context (i.e. not only cut sum-boxes) but only in the sum-box B. We denote this reduction by $\rightarrow_B$.

Because every summand in B are disjoined we can apply $\rightarrow_B$ in parallel on all summands of B. We denote $\Rightarrow_{Dx}$ such parallel reductions applied to every sum-boxes of the same box-depth x. We do it to have no conflict.

Lemma 2. *Let $P \in nMLL_u$. If $sum(P) \Rightarrow_{Db(P)}; \cdots; \Rightarrow_{D0} sum(P') \Rightarrow_M sum(P'')$ then we have:*

- *$b(P) = b(P')$ and the context is empty between two successive sum-boxes of P'. Moreover $c(P) = c(P')$.*
- *$b(P'') = 0$ i.e. P'' is sum-box free, and $\underline{c}(P') = \underline{c}(P'')$.*
- *For all choice of summand s_i for $i \in I$, $s_i(sum(P'')) \Rightarrow^{c(P)} P_i$ cut-free.*

The order is chosen to simplify the statement of the lemma. For another order it is enough to use the rule $\Rightarrow_M$ after every $\Rightarrow_{Dx}$ (but this may decrease de box-depth of P).

Theorem 2. *There is a sequence of $O(c(P) + b(P))$ parallel reductions which reduces a $nMLL_u$ proof net P in a cut-free one.*

Proof. Let P'' defined from P as in lemma 2. Because all the summands $\{s_i\}_{i\in I}$ are pairwise disjoint we have $\cup_{i\in I} s_i(sum(P'')) \Rightarrow^{c(P)} \cup_{i\in I} P_i$ cut-free. So $sum(P) \Rightarrow_D^{b(P)} sum(P') \Rightarrow_M^1 sum(P'') \Rightarrow^{c(P)} sum_{i\in I}(P_i)$ gives a cut-free proof net. □

The same sequence takes $O(\underline{c}(P))$ steps.

4 Boolean circuits and Boolean proof nets

In this section we briefly recall the definitions of Boolean circuits and Boolean proof nets, and several properties on them. The only novelties concern the Boolean proof nets of $nMLL_u$ that extend those of MLL_u. For further information see: [Ruz81,All89,Vol99] for uniformity and Boolean circuits, [Wol94,Par89] for non-deterministic Boolean circuits, [Ter04,MR06] for (uniform) Boolean proof nets of MLL_u.

– uniformity –

The given notions are used for Boolean circuit families and for Boolean proof net families. Here both are denoted by $F = (F_n)_{n\in\mathbb{N}}$. From an algorithmic point of view the uniformity is an important issue because only a uniform family can be regarded as an implementation of an algorithm. A family is called uniform if a description of the n'th element can be computed from 1^n. Description means all informations on the element like the sort, the predecessors and so on.

Formally, a family $F = (F_n)_{n\in\mathbb{N}}$ is so-called *L-uniform* (resp. *P-uniform*) if there is a function which computes a description of F_n from 1^n in space $O(log\ |F_n|))$ (resp. in time $|F_n|^{O(1)}$). If one works with *NC* then one could used *NC*-uniformity which is defined in the same way. Usually the description also is chosen according to the uniformity notion used.

Because we work with $NNC(poly) = NP$ the *P*-uniformity is sufficient. Nevertheless if one wants to study a property in $NNC(log\ n) = NC$ then it is necessary to use at least *NC*-uniformity. In practice we will use in this case the *L*-uniformity with the following notion of description where the links are identified by binary numbers as the sorts. Let W the set of binary words and $\overline{x}$ the binary representation of the integer x. The *direct connection language* of a proof net family $P = (P_n)$, denoted $L_{DC}(P)$, is the set of tuple $\langle \overline{y}, \overline{l}, \overline{w}, \overline{b}, \overline{s}\rangle \in W^5$ where for $n = y$ we have l is a link in P_n of sort b if $\overline{w} = \varepsilon$ else the w^{th} premise of l is the link b. We use the last bit $\overline{s}$ to say that the s^{th} sum-box contains the link l. Notice that the length of all identifiers is bounded by $log\ |P|$.

This description is given by analogy with the one of the circuits: The *direct connection language* of a Boolean circuit family $C = (C_n)_{n\in\mathbb{N}}$ over basis $\mathcal{B}$, denoted $L_{DC}(C)$, is the set of tuples $\langle \overline{y}, \overline{g}, \overline{w}, \overline{b}\rangle$, where for $n = y$ we have g is a gate in C_n labeled by the function b from $\mathcal{B}$ if $\overline{w} = \varepsilon$ else b is the w^{th} predecessor gate of g. In case of input, b is not a function but a sort (deterministic input or non-deterministic variable).

– Boolean proof nets –

Boolean values are represented with the type $\mathbf{B} = \parr^3(\alpha^\perp, \alpha^\perp, \otimes^2(\alpha, \alpha))$. The non-deterministic Boolean values are represented with the same type ! There are exactly two cut-free proof nets of MLL_u of this type and we only consider one cut-free proof net of $nMLL_u$, called resp. *false*, *true* and $b_2 \equiv sum(b_0, b_1)$ (see the Appendix):

$$b_0 \equiv par_s^{q,p,r}(tensor_r^{p,q}(ax_p, ax_q)) \text{ and } b_1 \equiv par_s^{p,q,r}(tensor_r^{p,q}(ax_p, ax_q))$$

A *Boolean proof net* with n inputs $\overrightarrow{p} = p_1, \ldots, p_n$, one output and $k(n)$ sum-boxes is a proof net $P(\overrightarrow{p})$ of $nMLL_u$ of type:

$$\vdash p_1 : \mathbf{B}^\perp[A_1], \ldots, p_n : \mathbf{B}^\perp[A_n], q : \otimes^{m+1}(\mathbf{B}, \overrightarrow{C})$$

for some $\overrightarrow{A} \equiv A_1, \ldots, A_n$ and $\overrightarrow{C} \equiv C_1, \ldots, C_m$ where we denote $\mathbf{B}[A]$ the formula $\mathbf{B}$ where all occurrences of α are substituted by A. Given $\overrightarrow{x} \equiv b_{i_1}, \ldots, b_{i_n}$, $P(\overrightarrow{x})$ denotes the proof net where we cut every b_i with p_i.

Without loss of generality we can always set $sum(P)$ for P: if a Boolean proof net P is without sum-boxes then $sum(P)$ is a sum-box with only one summand in MLL_u. This generalizes the uniform MLL_u Boolean proof nets.

For a given $\overrightarrow{x}$, a Boolean proof net $sum(P(\overrightarrow{x}))$ is of type of q and reduces in a unique cut-free proof net of the same type (e.g. by one of the reduction sequences of the previous section). We say that $sum(P(\overrightarrow{x}))$ evaluates to 1 iff one of its summands is b_1 with some *garbage* $\overrightarrow{C}$. There is an asymmetry between 1 and 0 as for non-deterministic Turing machines. To be more readable in the rest of the paper we often omit the added sum-box in $sum(P)$ and just write P as in this following definition.

An n-ary Boolean proof net $P(\overrightarrow{p})$ *computes* a function $f : \{0,1\}^n \to \{0,1\}$ (or *accepts* a set $X \subseteq \{0,1\}^n$) if $P(\overrightarrow{x})$ evaluates to $b_{f(x)}$ for every $\overrightarrow{x}$ corresponding to $x \equiv i_1 \dots i_n$.

For Boolean proof nets encoding standard Boolean functions as negation, conditional, disjunction, composition, duplication, and so on, the reader can see [Ter04] (and the Appendix): here everything is again valid.

– *NNC*() –

A *basis* is a finite set of sequences of Boolean functions. The standard basis are $\mathcal{B}_0 = \{\neg, \wedge, \vee\}$ and $\mathcal{B}_1 = \{\neg, (\wedge^n)_{n\in\mathbb{N}}, (\vee^n)_{n\in\mathbb{N}}\}$. The circuits over basis with an infinite sequence of Boolean functions (resp. without) are called *unbounded fan-in* (resp. *bounded fan-in*) circuits. In particular we extend the basis with the $stCONN_2$ gates which test the strong connectivity of the (set of) edges given in inputs and we use it to simulate the tightening cut-elimination of proof nets as in [Ter04].

A *deterministic Boolean circuit* with n inputs over a basis $\mathcal{B}$ is a directed acyclic graph with $n+1$ sources or inputs (vertices with no in-going edges) and one sink or output (a vertex with no out-going edges). Sources are labeled by literals from $\{x_1, \dots, x_n\} \cup \{1\}$ and nodes of in-degree k are labeled by one of the k-ary Boolean functions of $\mathcal{B}$. Non-inputs nodes are called *gates*, and in-degree and out-degree are called *fan-in* and *fan-out* respectively. Let F_n denote the set of all Boolean functions $f : \{0,1\}^n \to \{0,1\}$ for some $n \in \mathbb{N}$. A deterministic circuit *computes* a function in F_n (or *accepts* a set $X \subseteq \{0,1\}^n$) in a natural way.

A *non-deterministic* Boolean circuit C with n inputs over a basis $\mathcal{B}$ with k non-deterministic variables is a circuit with $n + k + 1$ sources labeled by $\{x_1, \dots, x_n\} \cup \{y_1, \dots, y_k\} \cup \{1\}$ s.t. it computes a function $f \in F_n$ as follows: for $x \in \{0,1\}^n$, $f(x) = 1$ iff $\exists y \in \{0,1\}^k$ a witness s.t. $C(x, y)$ evaluates to 1.

A *family* of circuits $C = (C_n)_{n\in\mathbb{N}}$ computes a function $f : \{0,1\}^* \to \{0,1\}$ (or accepts a set $X \in \{0,1\}^*$) if for every n the circuit C_n computes the restriction of f to F_n.

The *size* of a circuit is the number of gates and the *depth* is the length of a longest directed path. All dimensions are defined w.r.t the length of the input, which is denoted everywhere n. As usual we use abusively the words *poly* for $n^{O(1)}$ and *polylog* for $log^{O(1)}\ n$.

The *classes* NC^i and AC^i for $i \geqslant 0$ are the functions computable by uniform families of polynomial size, $O(log^i n)$ depth circuits over $\mathcal{B}_0$ and $\mathcal{B}_1$ respectively. We add $(stCONN_2)$ to the classes if the basis is extended with a $stCONN_2$ gate. The *class* $NNC^i(k(n))$ (resp. $NAC^i(k(n))$) are the functions computable by NC^i (resp. AC^i) circuit families with $O(k(n))$ non-deterministic variables. We denote NC, $NNC(k(n))$, AC and $NAC(k(n))$ the respective unions over the exponents of the depth. The hierarchy of NC and $NNC()$ is the following: $\forall i \in \mathbb{N}$ and $\forall j \in \mathbb{N}^*$

$$AC^0 \subsetneq NC^1 \subseteq L \subseteq NL \subseteq AC^1 \subseteq NC^2 \subseteq \ldots \subseteq AC = NC \subseteq P$$

$$AC^i \subseteq AC^i(stCONN_2) \subseteq AC^{i+1}$$

$$NNC^j(log\ n) = NC^j, \text{ and then } NNC(log\ n) = NC$$

$$NNC^j(poly) = NNC(poly) = NAC^i(poly) = NAC(poly) = NP$$

5 Translation and simulation

– Logspace translation –

Let $C = (C_n)_{n\in\mathbb{N}}$ be a uniform Boolean circuit family over the basis $\mathcal{B}_1(stCONN_2)$ with non-deterministic variables. We call module an intermediate proof net.

First of all without distinguish the inputs, we associate a uniform Boolean proof net family $P = (P_n)_{n\in\mathbb{N}}$ to C using the uniformity. After what we consider the non-deterministic variables. Indeed the uniformity function of Boolean circuits builds the uniformity function of Boolean proof nets as it was done in [MR06]. The main idea is already given in [Ter04]. E.g. starting from $L_{DC}(C)$:

- for each n-fan-in gate labeled $f(n)$ read in $L_{DC}(C_n)$ we give a polysize module computing $f(n)$. If the gate is a non-deterministic variable we cut the corresponding input with the proof net $b_2 \equiv sum(b_0, b_1)$,
- for each n-fan-out gate read in $L_{DC}(C_n)$ we make a polysize duplication,
- for each *edge* read in $L_{DC}(C_n)$ we glue the modules and the duplications.

Just parsing the $L_{DC}(C_n)$ for $i = 0$ to $|C_n|$ we detail a Logspace translation into $L_{DC}(P_n)$: everything is identified with a binary number

1. each $\langle n, i, \varepsilon, b\rangle$ builds the module associated to the function b of the basis of the family (or to the sort b in case of inputs). It is a subset of $L_{DC}(P_n)$ where relations between links and sorts are given.
2. if there is multiple $\langle n, i, k, j\rangle$ (i.e. j is the k-th predecessor of i) for fixed n and j then the fan-out of j is multiple. We build the corresponding duplication. It is again a subset of $L_{DC}(P_n)$.
3. each $\langle n, i, k, j\rangle$ (i.e. j is the k-th predecessor of i) builds $\langle n, a, b, c, 0\rangle$ (i.e. an edge from $(a, 0)$ to (b, c)) where a is the link associated to the output of the module (step 1) corresponding to j and b is the link associated to the c-th input of the module corresponding to i, modulo duplications added.

The novelty comes from the non-deterministic variables for which the module associated is a cut with $b_2 \equiv sum(b_0, b_1)$ but not a sort. Only here the last bit used is not zero.

Theorem 3. *For every uniform family C of unbounded fan-in Boolean circuit of size s and depth c over the basis $\mathcal{B}_1(stCONN_2)$ and with $k(n)$ non-deterministic variables, there is a uniform family of Boolean proof nets of $nMLL_u$ of size $s^{O(1)}$ and logical depth $O(c)$ and with $k(n)$ sum-boxes, which accepts the same set as C does.*

Proof. Let $C_n \in C$ and $P_n \in P$ the Boolean proof net obtained by translation. By translation $b(P_n) = 1$. Every gate is translated by a module of size $O(s^4)$ and constant depth, and only the composition of these modules increases linearly the depth [Ter04]. Let $x \in \{0,1\}^n$ an input of C_n and $\overrightarrow{x}$ corresponding to x. From the proof of theorem 2 we have: $P_n(\overrightarrow{x}) \Rightarrow^1_{D0}; \Rightarrow^1_M sum_{i\in I}(P_i) \Rightarrow^c sum_{i\in I}(Q_i)$ is an $O(c)$ steps reduction s.t. $sum_{i\in I}(Q_i)$ is cut free and there is a witness $y \in \{0,1\}^k$ s.t. $C_n(x,y)$ evaluates to 1 if and only if $P_n(\overrightarrow{x})$ evaluates to 1 (i.e. $\exists i \in I$ s.t. $Q_i = b_1$). □

Remark that the same holds with the $\Rightarrow_{ftd}$ reduction because the box-depth of a translated circuit is 1.

– Simulation of parallel cut-elimination –

Let $P = (P_n)_{n\in\mathbb{N}}$ be a uniform Boolean proof net family of $nMLL_u$ with $k(n)$ sum-boxes. We associate a uniform Boolean circuit family $C = (C_n)_{n\in\mathbb{N}}$ to P in two big steps based on the sequence of reductions of theorem 2:

- We both initialize the descriptions of the circuits and simulate all the parallel down reductions by a polysize and constant depth circuit using the uniformity,
- As done in [Ter04], we simulate all the $\Rightarrow$ reductions of all summands using $stCONN_2$ gates for $\Rightarrow_t$ simulation and finally we check the result of the last configuration.

From the description of a proof net $P_n \in P$ (or of the associated circuit in C) one built Θ_0 an initial set of boolean values representing the proof net to simulate. A *configuration* $\Theta \in Conf(P_n)$ is the set of the following Boolean values: $alive(p)$, $sort(p,s)$, $box(p,i)$ and $edge(p,0,q,i)$. Θ_0 values are initialized to 1 iff a link $p \in L$ is respectively in P_n, of sort s, contained in the i^{th} sum-box and of principal port in relation with the i^{th} port of a link $q \in L$. In pedantic terms our initial configuration is the description itself extended to alive values. Each reduction step simulation can be done with small circuits which modify a configuration to another. Everything could be done in Logspace.

Lemma 3. *There is an unbounded fan-in circuit C of size $O(|P_0|^3)$ and constant depth over $\mathcal{B}_1$ with non-deterministic variables, which computes in Logspace $\Theta \in Conf(P')$ from $\Theta_0 \in Conf(P_0)$ whenever $P_0(b_{i_1}, \ldots, b_{i_n}) \Rightarrow^{b(P)}_D \Rightarrow^1_M P'$ from given inputs $i_1, \ldots, i_n$.*

Proof. As done in the translation one can parse the configuration of $P_0(b_{i_1}, \ldots, b_{i_n})$ without taking care of sum-boxes to build partially $L_{DC}(C)$ in Logspace. From $Conf(P_0)$ we complete $L_{DC}(C)$ in Logspace and compute $Conf(P')$ as follows: The simulation of one k-ary sum-box t which corresponds to k summands/choices, uses $log\ k$ non-deterministic variables $\{G^t\}$ as done in the Fig. 8. Let l be a link of box-depth $b(l)$, i.e. l is contained in exactly $b(l)$ sum-boxes $\{t_i\}_{i\in I}$. Let k_i be the arity of the sum-box t_i. The link l depends on $\Sigma_{i\in I} log\ k_i$ non-deterministic gates. We initialize the value of the corresponding edges with the conjunction of the values of these non-deterministic gates $\cup_{i\in I}\{G^{s_i}\}$. Globally this constant depth initialization uses one conjunction gate by edge in $Conf(P_0)$ and one negation gate by non-deterministic gates. □

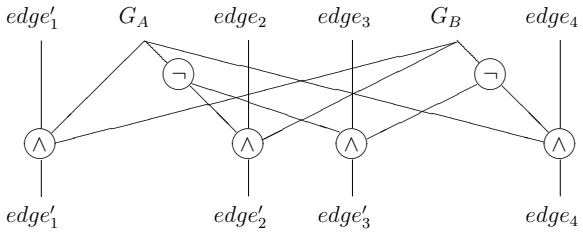

Fig. 8. 2^2-ary sum-box simulation where $edge_i$ are in the i^th summand

Lemma 4. *[Ter04] There is an unbounded fan-in circuit C over $\mathcal{B}_1(stCONN_2)$ of size $O(|P_0|^3)$ and constant depth such that whenever a configuration $\Theta \in Conf(P)$ is given as input and $P \Rightarrow P'$, C outputs a $\Theta' \in Conf(P')$.*

Theorem 4. *For every uniform family P of Boolean proof nets of size s and logical depth c, of $nMLL_u$ with $k(n)$ sum-boxes of maximal arity k, there is a uniform family of unbounded fan-in Boolean circuit over the basis $\mathcal{B}_1(stCONN_2)$ of size $s^{O(1)}$ and depth $O(c)$ and with $O(k(n).log\ k)$ non-deterministic variables, which accepts the same set as P does.*

Proof. By theorem 2 $\{i_1, \ldots, i_n\}$ is accepted by P if and only if $b_1 \in \{P_i\}_{i\in I}$ where $P(b_{i_1}, \ldots, b_{i_n})) \Rightarrow_D^{b(P)} \Rightarrow_M^1 P' \Rightarrow^{c(P)} sum_{i\in I}(P_i)$. We build a uniform polysize constant depth circuit: the lemma 3 proof gives the bound on the non-deterministic variables. For each of the $\Rightarrow^{c(P)}$ reductions we apply the Terui's lemma. Finally one easily build a polysize constant depth circuit for acceptance checking which decides if a given configuration represents b_1 or not. □

6 Proof net complexity

For $i \in \mathbb{N}$, the *class* $nmBN^i(k(n))$ and the *class* mBN^i are the functions computable by uniform families of polynomial size, $O(log^i n)$ depth Boolean proof

nets of respectively $nMLL_u$ with $O(k(n))$ sum-boxes and MLL_u. We denote $nmBN(k(n))$ and mBN the respective unions over the exponents of the depth. From the theorems 3 and 4 we obtain:

Theorem 5. *For all* $i \in \mathbb{N}$,
$NAC^i(k(n))(stCONN_2) \subseteq nmBN^i(k(n)) \subseteq NAC^i(k(n) \times log\ n)(stCONN_2)$

Proof. For a Boolean proof net of size s the arity k of a sum-box is $O(s)$ in the worst case. So $O(k(n).log\ k) = O(k(n) \times log\ n)$ because here $log\ O(s) = log\ n^{O(1)} = O(log\ n)$. □

Corollary 1. *For all* $i, j \in \mathbb{N}$,

1. $nmBN^i(poly) = NAC^i(poly)(stCONN_2) = NP$,
2. $NAC^i(log^j\ n)(stCONN_2) \subseteq nmBN^i(log^j\ n) \subseteq NAC^i(log^{j+1}\ n)(stCONN_2)$,
3. $nmBN(1) = mBN = NC$,
4. $nmBN(log\ n) \supseteq NC$,
5. $nmBN(polylog) = NNC(polylog)$,
6. $nmBN(poly) = NNC(poly) = NP$.

Proof. Point 1. $O(n^{O(1)} \times log\ n) = O(n^{O(1)})$.
Point 3. $NAC(1)(stCONN_2) = NC = NNC(log\ n) = NAC(log\ n)(stCONN_2)$.
Point 5. by union over i and j from Point 2.
Point 6. by union over i from Point 1. □

Remark that $nmBN(1) = mBN$ is what we expect: a constant number of sum-boxes corresponds to $n^{O(1)}$ summands/choices in the worst case then is simulable with a disjunction of a polynomial number of mBN circuits of same depth. I.e. it corresponds to $NNC(log\ n) = NC$.

7 Conclusion

We study the parallel reductions of the proof nets of the non-deterministic multiplicative Linear logic. We define uniform Boolean proof nets with an amount of explicit non-determinism analogously to uniform circuits of $NNC()$, the non-deterministic NC class. We apply our results to give a proof-as-programs correspondence between this two models of parallel computation, preserving both size and depth. We define in a standard way classes for families of uniform Boolean proof nets $nmBN()$ and we establish the following results:

$$\begin{array}{ccccccc} NC = nmBN(1) & \subseteq & nmBN(log\ n) & \subseteq & nmBN(polylog) & \subseteq & nmBN(poly) = NP \\ \shortparallel & & & & \shortparallel & & \shortparallel \\ mBN & & & & NNC(polylog) & & NNC(poly) \end{array}$$

Remark that the central theorems could apply for circuits without depth constraint. Such a circuit is simply called a polynomial size circuit. So there is a chain from P to NP for families of uniform polynomial size Boolean proof nets.

There is a reduction which replaces the sequence of $\Rightarrow_D$ in our theorem in only one step: it is what is simulated in circuits. The same could even be

done parsing the summands without to reduce sum-boxes (using only $k(n).log\ n$ bits) but with our theorem reduction we do fully parallel computation. Ad-hoc Boolean proof net classes can be given to have a more strictly correspondence with $NNC()$, using only binary $\Box$-links. Then the encoding are no more constant depth but $O(k(n))$ depth: one finds NC with $O(log\ n)$ sum-boxes. Remark that if we use a uniformity which is not sharp as the L-uniformity then the descriptions are more readable: e.g. for sum-boxes a relation between the $\Box$-links is sufficient.

Because in $nmBN()$ sum-boxes are not binary we only need a constant amount of sum-boxes to be equal to NC. So we are curious about the dimensions needed for the algorithm for quasigroup isomorphism (QI) problem given by [Wol94] i.e. $\exists ?i$ s.t. $QI \in nmBN^i(log\ n)$.

By showing that $NNC^i(log^j\ n) \subseteq DSPACE(log^{max(i,j)}\ n)$ in his attempt to separate NC from NP, Wolf [Wol94] gives a better intuition about $NNC()$ in term of space than of time. It could be interesting to explore this complexity classes however with the Boolean proof nets we are much closer to time: we can use more than a fixed quantity of non-deterministic variables but explicit non-determinism as functional constructor. Indeed a function can be fully non-deterministic as a choice between two functions, or it can be build with explicit non-deterministic type (nd-Bool) e.g. the function `AND` of type nd-Bool$\times$ Bool $\rightarrow$ Bool is defined by `AND(x,y)= if y then x else 0`. Such functional approach is not used in this paper but is efficient with proof nets enriched with additives. $m\&BN()$ are classes of Boolean proof nets enriched with fixed amount of $\&$ additive connectives to simulate non-determinism [MR06]. However $\&$ behavior is not strictly the non-determinism but a little more. The sum-boxes are a kind of weak additives that we can define for the Boolean proof net setting by extending additives to unary connectives: $\oplus_3\mathbf{B}$ and $\&\mathbf{B}$. The corresponding links are the usual binary $\&(\mathbf{B},\mathbf{B})$ but the $\oplus_3(\mathbf{B},\mathbf{B})$ is binary and cuts behave as for the sum-boxes. There is not an inelegant collapse of the additive neutrals as done in [Mat96] with auto-dual additive connectives (an equivalent version of nLL). An immediate advantage of Boolean proof net with unary additives is to avoid the garbage.

References

[All89] Eric W. Allender. P-uniform circuit complexity. *Journal of the Association for Computing Machinery*, 36(4):912–928, 1989.

[BS90] R. B. Boppana and M. Sipser. *The complexity of finite functions.* MIT Press, 1990.

[DR89] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28(3):181–203, 1989.

[Gir87] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–102, 1987.

[Gir96] Jean-Yves Girard. Proof-nets : the parallel syntax for proof theory. *Logic and Algebra*, 180, 1996.

[Mat96] Satoshi Matsuoka. Nondeterministic linear logic. *IPSJ signotes programming*, 12, 1996.

[Mau03] F. Maurel. Nondederministic light logics and πp-time. In *Proceedings of the 6th International Conference on Typed Lambda Calculus and Applications (TLCA'03)*, volume 2701 of *LNCS*, pages 241–255. Springer-Verlag, 2003.

[MR06] V. Mogbil and V. Rahli. Uniform circuits, & boolean proof nets. Preprint, 2006.

[Par89] I. Parberry. A note on nondeterminism in small, fast parallel computers. *IEEE Transactions on Computers*, 38(5):766–767, 1989.

[Ruz81] W. Ruzzo. On uniform circuit complexity. *J. of Computer and System Science*, 21:365–383, 1981.

[Ter04] K. Terui. Proof nets and boolean circuits. In *Proc. IEEE Logic in Comput. Sci.*, pages 182–191, 2004.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach*. Texts in Theoretical Computer Science. Springer Verlag, 1999.

[Wol94] Marty J. Wolf. Nondeterministic circuits, space complexity and quasigroups. *Theoretical Computer Science*, 125(2):295–313, 1994.

A Appendix

A.1 Functions in Boolean proof nets

The *conditional* (if-then-else) is the base of the Terui's gates translations: given two proof nets P_1 and P_2 of types $\vdash \Gamma, p_1 : A$ and $\vdash \Delta, p_2 : A$ resp., one can build a proof net $cond_r^{p_1,p_2}[P_1, P_2](q)$ of type $\vdash \Gamma, \Delta, q : \mathbf{B}[A]^{\perp}, r : A \otimes A$ (Fig.9(b)). Given a cut between b_i and q we have:

$$cond_r^{p_1,p_2}[P_1, P_2](b_1) \to^* tensor_r^{p_1,p_2}(P_1, P_2),$$
$$cond_r^{p_1,p_2}[P_1, P_2](b_0) \to^* tensor_r^{p_2,p_1}(P_2, P_1).$$

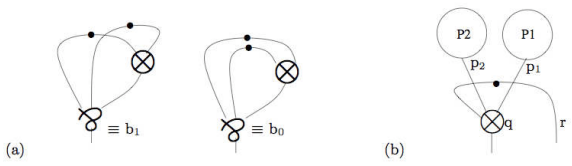

Fig. 9. (a) The Boolean b_1 and b_0 ; (b) The conditional

Disjunction, *conjunction* and *duplication* are based on the conditional: let $n \geqslant 2$ be an integer and $C \equiv \otimes(\mathbf{B}[A_1], \ldots, \mathbf{B}[A_n])$,

$or(p_1, p_2) \equiv cond[b_1, ax_{p_1}](p_2)$ of type $\vdash p_1 : \mathbf{B}^{\perp}, p_2 : \mathbf{B}[\mathbf{B}]^{\perp}, q : \mathbf{B} \otimes \mathbf{B}$,
$and(p_1, p_2) \equiv cond[ax_{p_1}, b_0](p_2)$ of type $\vdash p_1 : \mathbf{B}^{\perp}, p_2 : \mathbf{B}[\mathbf{B}]^{\perp}, q : \mathbf{B} \otimes \mathbf{B}$,
$copy^n(p) \equiv cond[tensor(\overrightarrow{b_1}), tensor(\overrightarrow{b_0})](p)$ of type $\vdash p : \mathbf{B}^{\perp}[C], q : C \otimes C$.

The *composition* of two translated circuits is defined by: let $\Gamma \equiv p'_1 : A'_1, \ldots, p'_n : A'_n$ and $\Delta \equiv q'_1 : B'_1, \ldots, q'_n : B'_m$, let $P(\overrightarrow{p'})$ and $Q(\overrightarrow{q'})$ be proof nets of type $\vdash \Gamma, p : \otimes^2(\mathbf{B}, \overrightarrow{C})$ and $\vdash q : \mathbf{B}^\perp[A], \Delta, r : \otimes^2(\mathbf{B}, \overrightarrow{D})$, respectively. Then: $comp_s^{p,q,r}[P,Q](\overrightarrow{p'}, \overrightarrow{q'})$ is of type $\vdash \Gamma[A], \Delta, s : \otimes^2(\mathbf{B}, \overrightarrow{D}, \overrightarrow{C[A]})$.

With this composition one can construct n-ary versions of conjunction and disjunction.

A.2 *nMALL* non-determinism

– $\oplus_3$ is redondant: (a) $\oplus_3$-rule (b) is derivable in nMALL –

$$\text{(a)}\ \frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \oplus_3 \qquad \text{(b)}\ \frac{\dfrac{\vdash \Gamma, A}{\vdash \Delta, A \oplus B}\ \oplus_1 \quad \dfrac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \oplus_2}{\vdash \Gamma, A \oplus B}\ sum$$

– $\oplus_3/\&$ reduction step simulation in nMALL –

$$\frac{\dfrac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \oplus_3 \quad \dfrac{\vdash \Delta, A^\perp \quad \vdash \Delta, B^\perp}{\vdash \Delta, A^\perp \& B^\perp}\ \&}{\vdash \Gamma, \Delta}\ cut$$

$$\longrightarrow_{\oplus_3/\&} \quad \frac{\dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ cut \quad \dfrac{\vdash \Gamma, B \quad \vdash \Delta, B^\perp}{\vdash \Gamma, \Delta}\ cut}{\vdash \Gamma, \Delta}\ sum$$

$$\frac{\dfrac{\dfrac{\vdash \Gamma, A}{\vdash \Delta, A \oplus B}\ \oplus_1 \quad \dfrac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\ \oplus_2}{\vdash \Gamma, A \oplus B}\ sum \quad \dfrac{\vdash \Delta, A^\perp \quad \vdash \Delta, B^\perp}{\vdash \Delta, A^\perp \& B^\perp}\ \&}{\vdash \Gamma, \Delta}\ cut$$

$$\longrightarrow_{sum/cut} \quad \frac{\dfrac{\vdash \Gamma, A}{\vdash \Delta, A \oplus B}\ \oplus_1 \quad \dfrac{\vdash \Delta, A^\perp \quad \vdash \Delta, B^\perp}{\vdash \Delta, A^\perp \& B^\perp}\ \& \quad \dfrac{\vdash \Gamma, B}{\vdash \Delta, A \oplus B}\ \oplus_2 \quad \dfrac{\vdash \Delta, A^\perp \quad \vdash \Delta, B^\perp}{\vdash \Delta, A^\perp \& B^\perp}\ \&}{\vdash \Gamma, \Delta}\ sum$$

$$\longrightarrow_{\oplus_1/\&, \oplus_2/\&} \quad \frac{\dfrac{\vdash \Gamma, A \quad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\ cut \quad \dfrac{\vdash \Gamma, B \quad \vdash \Delta, B^\perp}{\vdash \Gamma, \Delta}\ cut}{\vdash \Gamma, \Delta}\ sum$$