Non-deterministic Boolean Proof Nets

Virgile Mogbil

Non-deterministic Boolean Proof Nets

(rapport interne LIPN - Novembre 2009)

Virgile Mogbil

LIPN – UMR7030, CNRS – Université Paris 13, France
virgile.mogbil@lipn.univ-paris13.fr *

Abstract. We introduce Non-deterministic Boolean proof nets to study the correspondence with Boolean circuits, a parallel model of computation. We extend the cut elimination of Non-deterministic Multiplicative Linear logic to a parallel procedure in proof nets. With the restriction of proof nets to Boolean types, we prove that the cut-elimination procedure corresponds to Non-deterministic Boolean circuit evaluation and reciprocally. We obtain implicit characterization of the complexity classes NP and NC (the efficiently parallelizable functions).

1 Introduction

The *proof nets* [Gir87,DR89] of the Linear logic (LL) are a parallel syntax for logical proofs without all the bureaucracy of 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 functional 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 connectives, also known as modalities, controlling duplication in cut-elimination process. The most known restrictions characterize P, the class of decision problems which can be solved in polynomial time. E.g. it is the case of the Intuitionistic Light Affine Logic (ILAL). By expressing non-determinism by an explicit rule to choose between proofs of the same formula, 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. With proof nets, other characterizations were given by correspondence with models of parallel computation like Boolean circuits as we shall see later. In this paper we make use of the sum rule to extend those proof nets to a non-deterministic framework.

Boolean circuits are a standard models of parallel computation [Coo85,BS90] [Vol99]. 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 Boolean circuit has a fixed input size, an infinite family of Boolean circuits is needed to do computations on arbitrary inputs. With a

* Work partially supported by the french project Complice (ANR-08-BLANC-0211-01)

uniformity property, a family can be however regarded as an implementation of an algorithm. The Boolean 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 poly-logarithmic depth.

By restricting proved formulas and with a logical depth notion, there is a proofs-as-programs correspondence between proof nets and NC such that both size and depth are preserved [Ter04]. In this setting, the proof nets allow among others to simulate gates by higher order. Here we consider a non-deterministic extension of proof nets, to give a proof-as-programs correspondence with $NNC(k(n))$, the class defined from NC Boolean circuits with $O(k(n))$ non-deterministic variables. In particular $NC = NNC(log\ n)$ and $NP = NNC(poly)$. So the Curry-Howard isomorphism for parallel model of computation gives us new tools to study theoretical implicit complexity.

In section 2 we present MLL_u, the multiplicative LL with arbitrary arity This is the smallest fragment of LL that encodes the unbounded fan-in Boolean circuits [Ter04]. We recall the reduction steps of the cut-elimination. We give its non-deterministic extension $nMLL_u$ and proof nets for it as in [Mau03]. We recall several size and depth notions used in the proofs, all are natural graph theoretic notions. In section 3 we mostly analyze and define a new cut elimination from a parallel point of view. We prove the central theorems which allow us to establish the results on the complexity classes. In section 4 we recall Boolean circuit definitions. They include uniformity of both proof net families and circuit families as well as hierarchy of NC and $NNC()$. We introduce the Boolean proof nets of $nMLL_u$, i.e with sum-boxes, which generalize the Boolean proof nets of MLL_u. In section 5 we apply the previous theorems to our Boolean proof nets with sum-boxes and we establish a proofs-as-programs correspondence with $NNC()$ Boolean circuits. They are translation and simulation theorems that preserve both size and depth of the Boolean proof nets and Boolean circuits. Finally in section 6 we summarize the obtained results via $UBPN()$, a hierarchy of proof net complexity classes defined analogously to the $NNC()$ hierarchy. The classes $UBPN(poly)$, $UBPN(polylog)$ and $UBPN(1)$ 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

2.1 Formulas, Sequent calculus and cut-elimination

We write $\overrightarrow{A}$ (respectively $\overleftarrow{A}$) for an ordered sequence of formulas $A_1, \ldots, A_n$ (resp. $A_n, \ldots, A_1$).

The *formulas* of MLL_u and $nMLL_u$ are built on literals by conjunction $\otimes^n(\overrightarrow{A})$ and disjunction $\parr^n(\overleftarrow{A})$ of Multiplicative Linear logic but with unbounded arities $n \geqslant 1$. The only difference with binary connectives is that this gives us depth-efficient proofs [DR89]. The *negation* of a non-literal formula is defined by

De Morgan's duality: $(\otimes^n(\overrightarrow{A}))^\perp = \parr^n(\overleftarrow{A^\perp})$ and $(\parr^n(\overrightarrow{A}))^\perp = \otimes^n(\overleftarrow{A^\perp})$ where the negation applies to sequences of formulas as expected.

The *sequents* of $nMLL_u$ are of the form $\vdash \Gamma$, where Γ is a multiset of formulas. The rules of the $nMLL_u$ sequent calculus are described in Fig.1. As in Linear logic sequent calculus, MLL_u and $nMLL_u$ admit a cut-elimination theorem (Hauptsatz) and implicit exchange. The cut-elimination steps of $nMLL_u$

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

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

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

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

sequent calculus are n-ary versions of the Linear logic proofs rewriting: an axiom cut (i.e. a cut rule with an axiom premise) rewrites without the axiom rule, and the multiplicative cut (i.e. a cut rule between multiplicative formulas) rewrites in sequence of cuts between premises. In Fig.2 we give the sum-rule cut-elimination step as in [Mau03].

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

Fig. 2. $nMLL_u$ sum-rule cut-elimination step

2.2 Proof nets and reduction rules

We suppose the reader familiar with proof nets and more specifically with the reduction rules corresponding to the cut-elimination steps [Gir96].

A *proof net* [Gir87,DR89] is a graph that just keeps the structure of sequent calculus proofs without some computational irrelevant part. It is a set of connected links that we build by inference from the sequent calculus rules. There

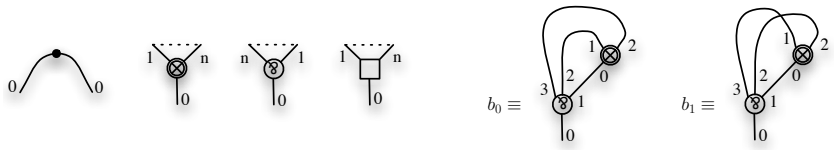


Fig. 3. *ax*-link, $\otimes$-link, $\parr$-link and $\square$-link ; small proof nets

are several *sorts* of links: *ax*-link, $\otimes$-link and $\parr$-link corresponding respectively

to MLL_u rules. Every link has several ports numbered by convention as in Fig.3. This numbering is omitted when there is no ambiguity. The *conclusion port*, also called conclusion of the link, is the port numbered 0. We use a wire between two conclusion ports rather than a cut-link that corresponds to sequent calculus cut-rule. Now we focus us on the $nMLL_u$ proof nets which extend those of MLL_u. As we only consider proof nets inferred from sequent calculus, we can inductively built them in a equivalent way from the ax-links with the constructors of Fig. 4. The sequent calculus sum-rule corresponds to a box (Fig.4), called *sum-box*, enclosing proof nets, called *sub-nets*, such that we associate one $\square$-link for each shared conclusion.

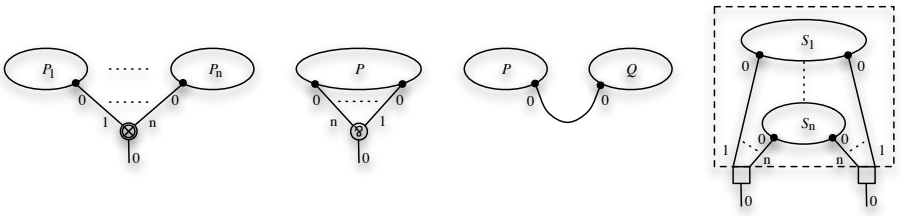


Fig. 4. Proof net constructors: $\otimes$-link, $\parr$-link, cut and sum-box

Definition 1. *A* proof net *is (i) a finite set L of links, (ii) a function $\sigma : L \to \{\bullet, \square\} \cup \{\otimes^n, \parr^n\}_{n\geqslant 1}$ giving the sort of each link, (iii) a symmetric binary relation over $L \times \mathbb{N}$ for each wire between two ports of distinct links, and (iv) a function $\tau : L \to \mathbb{N}$ which associates to each link a unique number corresponding to the sum-box that encloses it.*

A summand *of a proof net is a proof net obtained by erasing all but one sub-net in every sum-box.*

In section 4 we give a Logspace description which extends this one. We call the *conclusions* of a proof net the set of links whose conclusion ports are unrelated with other link ports. We write proof nets either with graphs or just with the link names. For instance the last proof net in Fig.4 is $sum(S_1, \ldots, S_n)$.

Remark that in every summand of a proof net the $\square$-links correspond to superfluous sum-rules with only one premise. So every summand of a proof net of $nMLL_u$ easily induces a proof net of MLL_u.

Definition 2. *The reduction rules*[1] *for MLL_u proof nets, called respectively ax-*reduction *and m-*reduction*, are defined by:*

$$cut(ax(A, A^\perp), A) \ \to_{ax} \ A \text{ and } cut(\otimes^n(\overrightarrow{A}), \parr^n(\overleftarrow{A}^\perp)) \ \to_m \ \{cut(A_i, A_i^\perp)\}_{1\leqslant i\leqslant n}$$

*For $nMLL_u$ proof nets there are moreover the reduction rules [Mau03], called respectively merge-*reduction *and down-*reduction*, defined for an arbitrary context C by:*

[1] Up to the order when they are not drawn

$$sum(sum(\overrightarrow{A}), \overrightarrow{B}) \rightarrow_{merge} sum(\overrightarrow{A}, \overrightarrow{B})$$

$$sum(C[sum(\overrightarrow{A})], \overrightarrow{B}) \rightarrow_{down} sum(sum(C[\overrightarrow{A}]), \overrightarrow{B})$$

We define various notions of depth in a natural way: the box-depth associated to sum-boxes, the depth of formulas and the logical depth associated to cuts.

Definition 3. *The* box-depth $b(l)$ *of a link* l *is the number of sum-boxes that enclose it. The box-depth of a sum-box is the box-depth of its* $\Box$*-link links. The box-depth* $b(P)$ *of a proof net* P *is the maximal box-depth of its links.*

The depth $d(A)$ *of a formula* A *in a sequent calculus derivation* π *is the length of the longest sequence of rules in* π *since axioms until the introduction of the formula* A*. The depth* $d(\pi)$ *of a* $nMLL_u$ *sequent calculus derivation* π *is the maximal depth of its cut formulas.*

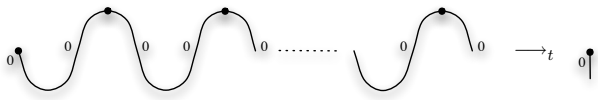
The logical depth $\underline{c}(P)$ *of a proof net* P *is* $\underline{c}(P) = min\{d(\pi) \mid P$ *is inferred by the sequent calculus derivation* $\pi\}$*. We write* $c(P)$ *the logical depth without counting the sum-rules.*

Remark that by definition all $\Box$-links of the same sum-box are at same box-depth. When a proof net is induced by a sequent calculus proof, he keeps traces of sequentialization by the sum-boxes. This corresponds to a stratification by the box-depth that we use in our reduction strategy.

The *size* $|P|$ of a proof net P is the number of links in P such that we count boxes but not the $\Box$-links.

3 Parallel reductions

Like the cut-elimination of sequent calculus the reduction of proof net is terminating. Even if the reduction of proof nets has the Church-Rosser property, reductions cannot be done in parallel as there is critical pairs. A critical pair arises when redexes of reduction rules overlap. In order to have efficient proof net reductions we consider all possible critical pairs. E.g. a cut between two distinct axioms has two overlapping redexes depending on if we consider one or the other axioms. This is solved for MLL_u in [Ter04] by introducing a *tightening reduction*: one reduces each maximal alternating chain of cuts and axioms in only one global step leading to what is expected by sequential reduction steps. Here is an example where the maximal chain starts with a cut and finishes with an axiom:



Remark that by maximality, no critical pairs can be tightening redexes. So such reductions can be done in parallel. We denote t-reduction a tightening reduction and we write it $\rightarrow_t$. We write $\Rightarrow_t$ for the simultaneous application of all the $\rightarrow_t$ reductions that can be fired in a given proof net. As $\rightarrow_m$ redexes are never

critical pairs we also write $\Rightarrow_m$ for the simultaneous $\rightarrow_m$ reductions. In the rest of the paper we write $\Rightarrow^k$ for k parallel reduction steps ($\Rightarrow_t$ or $\Rightarrow_m$) in MLL_u.

In this section we introduce new reduction rules to answer the parallelization question for $nMLL_u$ such that reduction is confluent. We will give a bound on the number of reductions to normalize a proof net (theorem 1).

3.1 Parallel reductions of merged sum-boxes

The merge rules applied to distinct sum-boxes immediately within a same sum-box are not really a conflict case even if it is a critical pair. We could merge them in parallel if the sum-box itself would not be merged at the same time. Because the merge-rule is confluent one can consider a global rewriting rule by merging sum-boxes as expected. This corresponds to a sort of innermost sequential strategy. We define a new reduction $\rightarrow_M$ whose redex is a maximal tree of successively nested sum-boxes. So the root of the tree of sum-boxes is an outermost sum-box that cannot be merged further. The $\rightarrow_M$ reduction merges this tree in the root sum-box as it is mimicked in Fig.5. We do not draw neither wires between the $\square$-links nor the sub-nets which are not sum-boxes. By maximality this reduction has no critical pairs, so we can do simultaneously $\rightarrow_M$ reductions.

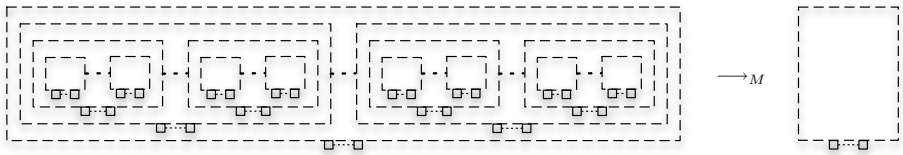


Fig. 5. $\rightarrow_M$ is the global reduction by merging

3.2 Parallel down-reductions

According to the context C, the pattern $sum(C[sum(\overrightarrow{A})], \overrightarrow{B})$ can reveal a critical pair for the down rule. E.g. within a sum-box $cut(sum(A_1, A_2), sum(B_1, B_2))$ reduces to $sum(sum(\{cut(A_i, B_j)\}_{1 \leqslant i,j \leqslant 2}))$ in four $\rightarrow_{down}$ steps. The same holds for a more general context C but remains convergent. So we define a new reduction rule based on the mathematical notion of section of indexed family of elements.

Definition 4. *Let B be a sum-box and let S be a sub-net of B, i.e. $B = sum(\overrightarrow{X}, S)$. Let $\{B_i\}_{i \in I}$ be the family of nested sum-boxes of B in S, i.e. $B_i \subset S$ and $b(B_i) = b(B) + 1$ for all $i \in I$. Let C be the context of the B_i with relation to S, i.e. $S = C[\{B_i\}_{i \in I}]$. We write $\overline{B_i}$ the family of sub-nets of B_i, i.e. $B_i = sum(\overline{B_i})$. The* section *$A \triangleleft F$ of a family $F = \{\overline{B_i}\}_{i \in I}$ is a family $A = \{a_k\}_{k \in I}$ such that for all $k \in I$ we have $a_k \in \overline{B_i}$. We define $\rightarrow_B$ the reduction of a sub-net of a sum-box B by:*

$$sum(\overrightarrow{X}, C[\{sum(\overline{B_i})\}_{i \in I}]) \quad \rightarrow_B \quad sum(\overrightarrow{X}, \{C[A] \mid A \triangleleft \{\overline{B_i}\}_{i \in I}\})$$

We abusively use a set to denote the reduct but the corresponding family is easily obtained from the implicit order associated to sections.

The intuition is to reduce in one (parallelizable) reduction step the sum-boxes B_i of one sub-net S, merging only affected sum-boxes: the result is a set of sub-nets which are the combination of all the contents of the B_i and the context outside the B_i. So the sum-boxes B_i go down whereas contexts go up. Remark that if the family of B_i is such that each B_i contains exactly one sub-net then by definition the context C applies on this set of sub-nets.

We give an example: let $B_1 = sum(a_1, a_2)$ and $B_2 = sum(b_1, b_2)$, let C be a context without other sum-boxes than B_1 and B_2. We have:
$sum(\overrightarrow{X}, C[B_1, B_2]) \rightarrow_B sum(\overrightarrow{X}, C[a_1, b_1], C[a_1, b_2], C[a_2, b_1], C[a_2, b_2])$.

Remark that the affected sum-boxes were merged by our reduction, but just to have a more readable definition. This reduction rule is a generalization of the case of context composed only with cut sum-boxes: in such case we need to reduce at fixed box-depth after all other reductions at this box-depth. But without this generalization we are not able to decrease at the same time the logical depth.

We redefine now $\rightarrow_B$ as the previous reduction extended to all sub-nets of the same sum-box B. This reduction is well defined as sub-nets are disjoints. We write $\Rightarrow_{Dx}$ such reductions applied in parallel to every sum-box at the same box-depth $x - 1$. By this way the sum-boxes at box-depth x go down. We do this in order to have no conflicts. For a given proof net P, we write $\Rightarrow_D{}^{b(P)}$ the reduction sequence $\Rightarrow_{Db(P)} \cdots \Rightarrow_{D1}$. So with our new reduction rule and our strategy we have the following remarkable properties:

Lemma 1. *Let P be a proof net such that $b(P) > 0$.*
If $sum(P) \Rightarrow_{Db(P)} sum(P')$ then $b(P') = b(P) - 1$ and $c(P') = c(P)$.

Proof. Let S be a sub-net of a sum-box B of box-depth $b(P) - 1$ in $sum(P)$. We have $b(S) = 1$. By definition each link in the reduct of S by $\rightarrow_B$ has the box-depth $b(P) - 1$. It is the same for all sub-nets of B by $\rightarrow_B$ reduction. By definition the $\Rightarrow_{Db(P)}$ reduction applies to all sum-boxes of box-depth $b(P) - 1$ in $sum(P)$. So the conclusion follows. □

The above Lemma implies the following:

Lemma 2. *Let P be a proof net such that $b(P) > 0$.*
If $sum(P) \Rightarrow_{Db(P)} \cdots \Rightarrow_{D1} \Rightarrow_M sum(P')$ then we have:

i) $c(P') = c(P)$ and $b(P') = 0$ i.e. P' is sum-box free.
ii) For all choice of summand s_i for $i \in I$, $s_i(sum(P')) \Rightarrow^{c(P)} P_i$ cut-free.

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.*

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

4 Boolean circuits and Boolean proof nets

In this section we recall the definitions and some properties of Boolean circuits and Boolean proof nets. We give also certain relationships between complexity classes associated to Boolean circuits [Coo85,All89,Wol94,Par89]. The novelties concern only the Boolean proof nets of $nMLL_u$ that extend those of MLL_u.

4.1 Boolean circuits

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. We extend the basis with a $stCONN_2$ gate to test the strong connectivity of an edge set given in input. As in [Ter04] we will use this gate to simulate the tightening reduction of proof nets.

Definition 5. *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 incoming edges) and one sink or output (a vertex with no out-going edges). Sources are labeled by literals from $\{x_1, \ldots, x_n\} \cup \{1\}$ 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 be 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 [Wol94,Par89] is a circuit with $n+k+1$ sources labeled by $\{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_k\} \cup \{1\}$ 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 $n \in \mathbb{N}$ 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.*

4.2 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 ! Let us consider the following proof nets of type $\mathbf{B}$: the non-deterministic Boolean represented by $b_2 \equiv sum(b_0, b_1)$ where b_0 and b_1 are the two cut-free proof nets of MLL_u respectively called *false* and *true* given in Fig.3.

Definition 6 ([Ter04]). *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:*

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

with $k(n)$ a function of n, and 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 $\overrightarrow{x} \equiv b_{i_1}, \ldots, b_{i_n}$ we write $P(\overrightarrow{x})$ 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 uniform MLL_u Boolean proof nets. We often omit this initial sum-box for the sake of simplicity.

Following the definition, for a given $\overrightarrow{x} \equiv b_{i_1}, \ldots, b_{i_n}$, a Boolean proof net $sum(P(\overrightarrow{x}))$ is of type $\otimes^{m+1}(\mathbf{B}, \overrightarrow{C})$ and reduces in a unique cut-free proof net of the same type (e.g. by the reduction sequence 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.

Definition 7 ([Ter04]). *A Boolean proof net $P(\overrightarrow{p})$ with n inputs* 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 $\overrightarrow{x} \equiv b_{i_1}, \ldots, b_{i_n}$ corresponding to $x \equiv i_1 \cdots i_n \in \{0,1\}^n$.*

The Boolean proof net encoding of K. Terui [Ter04] of functions as negation, conditional, disjunction, composition, duplication, and so on, is trivially valid in the extended setting of $nMLL_u$. We give some of them in the Appendix.

4.3 Uniformity and complexity classes

The given notions are used for Boolean circuit families and for $nMLL_u$ proof net families. 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.

Definition 8 ([Ruz81]). *A family $F = (F_n)_{n \in \mathbb{N}}$ is called* L-uniform *(respectively* P-uniform*) if there is a function which computes a description of F_n from 1^n (unary numeral) in space $O(log\ |F_n|)$) (respectively in time $|F_n|^{O(1)}$).*

A description means all informations about elements like sort, predecessors, ... Usually the description is also chosen according to the uniformity notion used. Because we work with *NP* the P-uniformity is sufficient [All89]. Nevertheless if we want to study a property in *NC* then we use the L-uniformity with the following notion of description where links and sorts are identified by binary numbers. Let W be the set of binary words and let $\overline{x}$ be the binary representation of the integer x.

Definition 9 ([Ruz81]). *The* direct connection language $L_{DC}(C)$ *of a Boolean circuit family $C = (C_n)_{n \in \mathbb{N}}$ over basis $\mathcal{B}$ is the set of tuples $\langle \overline{y}, \overline{g}, \overline{w}, \overline{b} \rangle \in W^4$, where for $y = n$ 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).*

This definition adapted to MLL_u proof nets [MR07] is here extended analogously to $nMLL_u$ proof nets:

Definition 10. *The* direct connection language $L_{DC}(P)$ *of a* $nMLL_u$ *proof net family* $P = (P_n)_{n\in\mathbb{N}}$ *is the set of tuple* $\langle \overline{y}, \overline{l}, \overline{w}, \overline{b}, \overline{s}\rangle \in W^5$ *where for* $y = n$ *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*. The link* l *is also enclosed by the* s^{th} *sum-box.*

Remark that the length of all identifiers of such family P is bounded by $log\ |P|$.

We finish this section recalling the complexity classes defined with uniform families of Boolean circuits ([Coo85,All89,Vol99] and [Wol94,Par89] for non-deterministic Boolean circuits). All dimensions are defined w.r.t the length of the input, which we write everywhere n. 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 the suffix $(stCONN_2)$ to classes if we extend basis 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 write NC, $NNC(k(n))$, AC and $NAC(k(n))$ the respective unions over the exponents of the depth. Relationships between complexity classes follow: $\forall i \in \mathbb{N}$ 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

5.1 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. The intermediate proof nets we build are called modules.

First of all without distinguishing the inputs, we associate a uniform Boolean proof net family $P = (P_n)_{n\in\mathbb{N}}$ to C using uniformity. After what we consider non-deterministic variables. Indeed the uniformity function of Boolean circuits builds the uniformity function of Boolean proof nets in the same way as in [MR07]. The main idea is already given in [Ter04]. 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 joint modules and 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. For each $\langle n, i, \varepsilon, b\rangle$ we build 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 relationships between links and sorts are given.
2. If there are several $\langle n, i, k, j\rangle$ (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. For each $\langle n, i, k, j\rangle$ (i.e. j is the k-th predecessor of i) we build $\langle n, a, b, c, 0\rangle$ (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 added duplications.

The novelty is the translation of non-deterministic variables into cuts with $b_2 \equiv sum(b_0, b_1)$. Only in this case the last bit of the description is not null.

Theorem 2. *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 as in definition 7. By theorem 1 proof we have: $sum(P_n(\overrightarrow{x})) \Rightarrow^1_{D1} \Rightarrow^1_M sum(\{P_i\}_{i\in I}) \Rightarrow^c sum(\{Q_i\}_{i\in I})$ is an $O(c)$ steps reduction such that $sum(\{Q_i\}_{i\in I})$ is cut free and there is a witness $y \in \{0,1\}^{k(n)}$ such that $C_n(x, y)$ evaluates to 1 if and only if $P_n(\overrightarrow{x})$ evaluates to 1 (i.e. $\exists i \in I$ such that $Q_i = b_1$). □

By translation $c = O(\underline{c})$, so we have the same result for logical depth $O(\underline{c})$.

5.2 Simulation of parallel reductions

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 reduction sequence of theorem 1:

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

In more detail, from the description of a proof net $P_n \in P$ we build 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: $alive(l)$, $sort(l, s)$,

$box(l,i)$ and $edge(l,0,l',i')$. These values are initialized to 1 respectively iff a link $l \in L$ is in P_n, l is of sort s, l is enclosed by the sum-box numbered i, and the conclusion port of l is in relation with the i'^{th} port of a link $l' \in L$. In other terms our initial configuration Θ_0 is the description itself extended to alive values. Every reduction step simulation is made by a small circuit which modifies a configuration in the other. All of this can be done in Logspace.
The parallel reduction strategy is simulated as follows: this is the tricky part

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

Proof. Like in the translation we parse the configuration of $P_n(b_{i_1},\ldots,b_{i_n})$ without taking care of sum-boxes to build partially $L_{DC}(C)$ in Logspace. From $Conf(P_n)$, 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 $\{G^t\}$. Let l be a link of box-depth $b(l)$, i.e. l is enclosed in exactly $b(l)$ sum-boxes $\{t_i\}_{i\in I}$. Let k_i be the arity of the sum-box t_i. So the link l simulation 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 $\cup_{i\in I}\{G^{t_i}\}$ like in Fig. 6. Globally this constant depth initialization uses one conjunction gate by edge in $Conf(P_n)$ and one negation gate by non-deterministic gates. □

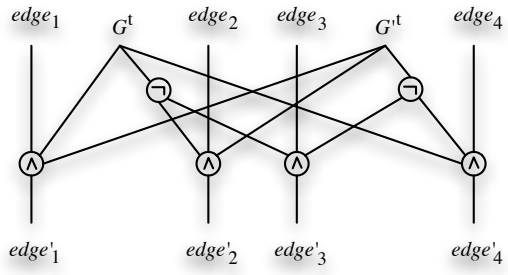


Fig. 6. 2^2-ary sum-box simulation where $edge'_i$ is in the i^{th} summand

Lemma 4. *[Ter04] There is an unbounded fan-in circuit C over $\mathcal{B}_1(stCONN_2)$ of size $O(|P|^3)$ 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 3. *For every uniform family P of Boolean proof nets of $nMLL_u$ of size s and logical depth c, 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 1, $i_1 \cdots i_n$ is accepted by P if and only if $b_1 \in \{Q_i\}_{i \in I}$ where $P_n(b_{i_1}, \ldots, b_{i_n}) \Rightarrow_D^{b(P)} \Rightarrow_M^1 P' \Rightarrow^{c(P)} sum(\{Q_i\}_{i \in I})$. By lemma 3 we build from P_n a uniform polysize constant depth circuit with $O(k(n).log(k))$ non-deterministic variables. For each of the $\Rightarrow^{c(P)}$ reductions we apply the same construction as in Terui's lemma starting from our previous configuration. At the end we easily build a polysize constant depth circuit to check if the last configuration represents b_1 or not. □

6 Proof net complexity

We define a hierarchy of complexity classes based on proof nets:

Definition 11. *For $i \in \mathbb{N}$, the* classes $UBPN^i(k(n))$ *and* $UBPN^i$ *are 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 write $UBPN(k(n))$ and $UBPN$ the respective unions over the exponents of the depth.*

From theorem 2 and theorem 3 we have:

Theorem 4. *For all $i \in \mathbb{N}$,*
$NAC^i(k(n))(stCONN_2) \subseteq UBPN^i(k(n)) \subseteq NAC^i(k(n) \times log\ n)(stCONN_2)$

Proof. For a Boolean proof net of size s the arity k of a sum-box is $O(s)$ in the worst case. By theorem 2 we have $O(s) = n^{O(1)}$. So $O(k(n).log\ k) = O(k(n) \times log\ n)$. □

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

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

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 $UBPN(1) = UBPN$ is what we expect: a constant number of sum-boxes corresponds to $n^{O(1)}$ summands/choices in the worst case, and so it can be simulated with a disjunction of a polynomial number of circuits of the same depth. I.e. it corresponds to $NNC(log\ n) = NC$.

7 Conclusion

Our study generalizes the work of K. Terui [Ter04] to non-deterministic extension as follows. We have defined new parallel reductions of proof nets of Non-deterministic Multiplicative Linear logic, $nMLL_u$. We have defined the uniform Boolean proof nets with an amount of explicit non-determinism analogously to the uniform Boolean circuits of $NNC()$, the non-deterministic NC class. By the way we give a proof-as-programs correspondence between this model of parallel computation and the uniform Boolean proof nets, preserving both size and depth. We define in a standard way the classes of uniform Boolean proof nets families $UBPN()$ and we establish the following results:

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

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

There is a reduction which replaces a sequence of $\Rightarrow_D$ in our theorem in only one step: our circuit simulating parallel down reductions made it in lemma 3. The same thing could be done by parsing the sub-nets without reducing sum-boxes (only $k(n).log(n)$ bits are used) 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: it corresponds to NC with $O(log\ n)$ sum-boxes. Remark that if we use a bigger uniformity notion then the descriptions are easier: e.g. for sum-boxes a relationship between the $\Box$-links is sufficient.

Acknowledgments. I would like to thank the reviewers for their remarks and the improvements which they suggested.

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.

[Coo85] Stephen A. Cook. A taxonomy of problems with fast parallel algorithms. *Inf. Control*, 64(1-3):2–22, 1985.

[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.

[Mau03] F. Maurel. Nonderministic light logics and NP-time. In *Proceedings of International Conference on Typed Lambda Calculus and Applications*, volume 2701 of *LNCS*, pages 241–255. Springer-Verlag, 2003.

[MR07] V. Mogbil and V. Rahli. Uniform circuits, & boolean proof nets. In *Proceedings International Symposium on Logical Foundations of Computer Science*, volume 4514 of *LNCS*, pages 401–421. Springer, 2007.

[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 *Proceedings IEEE Logic in Computer Science*, 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: functions in Boolean proof nets

The ports of the proof nets are labeled by letters rather than numbers, so that proof nets described only by the name of their links are more readable. For a given link l, we denote $l_q^{p_1,\ldots,p_n}$ if its conclusion port is q and the other ports are respectively $p_1, \ldots, p_n$ in this order. Axiom link are more simply denoted ax_p.

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.7(b)). Given a cut between b_i and q we have:

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

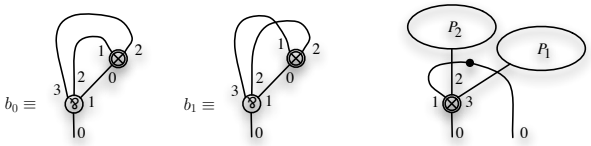


Fig. 7. (a) The Boolean b_0, b_1 and (b) The conditional Boolean proof net

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

$or(p_1, p_2) \equiv cond[b_1, ax_{p_1}](p_2)$ 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 $\vdash p_1 : \mathbf{B}^\perp, p_2 : \mathbf{B}[\mathbf{B}]^\perp, q : \mathbf{B} \otimes \mathbf{B}$,

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

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 we have:

$$comp_s^{p,q,r}[P, Q](\overrightarrow{p'}, \overrightarrow{q'})$$

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.