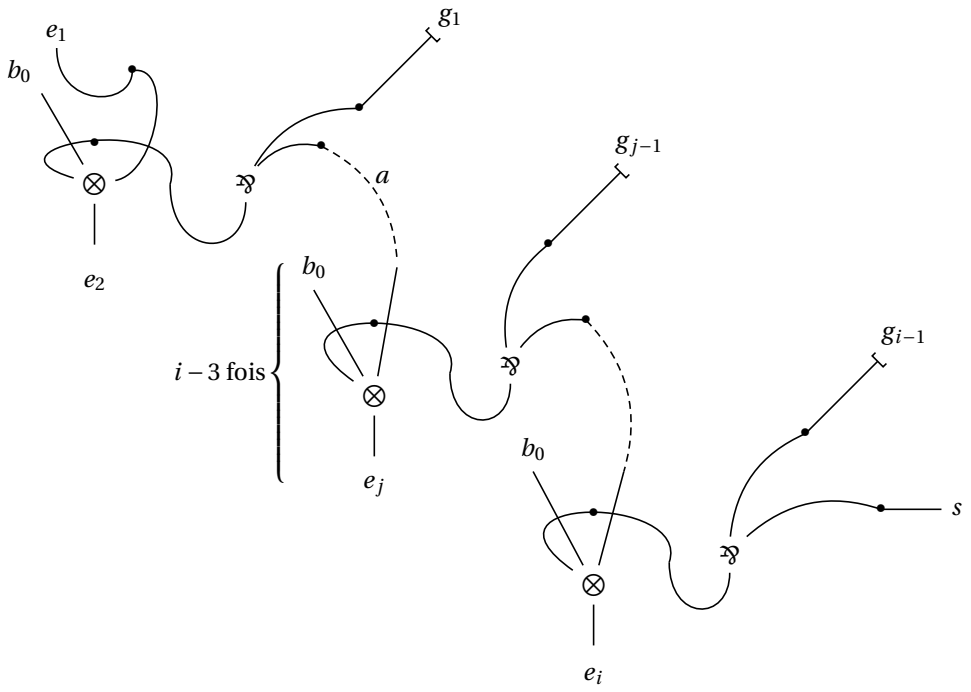
Mémoire en vue de l'obtention du M2 L.M.F.I. à l'Université Paris VII

Encadrants : Virgile Mogbil & Paulin Jacobé de Naurois
Laboratoire d'accueil : Laboratoire d'Informatique de Paris-Nord
(UMR 7030 du CNRS), Université Paris XIII (Villetaneuse)

Réseaux de preuves booléens sous-logarithmiques

Clément Aubert



Paris, septembre 2010

Introduction

L'objectif de ce mémoire est d'établir une connexion formelle entre deux modèles canoniques du calcul parallèle, à savoir les circuits booléens et les réseaux de preuves. Une machine de Turing est capable de simuler l'un et l'autre, ce qui fait que l'on a déjà un «passage» entre ces deux représentations, mais cette simulation présente un inconvénient majeur : elle linéarise le temps de calcul, nous faisant perdre tout le possible gain du calcul parallèle (pour une taxinomie des problèmes efficacement parallelisables, on peut par exemple se reporter à [Cook, 1985]).

En utilisant la correspondance preuves-programmes, il est possible de mettre en relation l'élimination des coupures dans un réseau de preuve et l'exécution parallèle de programmes fonctionnels : les réseaux sont identifiés aux circuits et l'élimination des coupures coïncide avec l'évaluation des circuits.

Un premier pas a déjà été effectué dans cette direction avec les travaux de H. Mairson et de K. Terui ([Mairson et Terui, 2003] et [Mairson, 2004]), qui ont montré que tout circuit booléen de taille s peut être simulé par une preuve en **Logique Linéaire Multiplicative** (**MLL**) – codée par un λ-terme linéaire – de taille linéaire en s. En plus d'établir une connexion entre ces deux modèles, ce résultat présente l'interêt de borner la taille de la traduction qu'on obtient. En effet, pour que la simulation d'un modèle par l'autre soit vraiment pertinente, il convient de se soucier des ressources nécessaires : on entre là dans le domaine de la complexité implicite, qui tente de borner les pouvoirs déductifs de nos outils afin de contrôler les ressources nécessaires à leur exécution.

Le problème complet pour le temps polynomial (P) canonique est *Circuit-Value* ([Ladner, 1975]) : il s'agit, étant donné un circuit booléen et une entrée, de décider quelle est la sortie du circuit. Puisqu'il existe une réduction en temps logarithmique (L) de ce problème à l'évaluation d'un réseau de preuve en **MLL**, l'élimination des coupures dans **MLL** est également un problème P-complet ([Mairson et Terui, 2003]).

Une étape supplémentaire dans la compréhension de cette correspondance a été franchie grâce à [Terui, 2004], et aux travaux de V. Mogbil et V. Rahli qui ont suivis ([Mogbil et Rahli, 2007] et [Mogbil, 2009]).

Objectifs du stage

Ce stage avait plusieurs buts, parmi lesquels «l'étude des articles définissant les relations entre les réseaux de preuves et les circuits booléens» ([Terui, 2004], [Mogbil et Rahli, 2007]), et «la caractérisation des petites classes de réseaux de preuves pour lesquelles il n'y a pas de résultats connus : uniformité et fonctions de $NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq AC^1$.»

«L'étude des classes de réseaux de preuves avec non-déterminisme explicite» était également un possible sujet d'étude, s'appuyant sur [Mogbil et Rahli, 2007] et [Mogbil, 2009]. Ce sujet n'a pas pu être approfondi, car il s'est avéré que les deux premiers objectifs étaient déjà riches et ont menés à quelques découvertes. Néanmoins, la lecture attentive de ces articles a déjà conduit à une première sensibilisation aux réseaux booléens avec non-déterminisme et aux classes de complexité correspondantes

Résultats

Notre attention s'est portée sur les résultats de K. Terui, qu'il nous semblait possible d'améliorer, ou du moins de rendre plus compréhensibles, [Terui, 2004] étant très dense et parfois elliptique. Sa simulation des réseaux booléens par des réseaux de preuves est très lourde et peu intuitive. Ainsi, la disjonction n-aire est vue comme la composition de $n-1$ disjonction binaires, alors qu'il est possible de la définir directement de façon beaucoup plus simple.

Nous avons ainsi conçu des *pièces* (définition 22, page 15), qui sont des pré-réseaux de preuves simulant les connecteurs booléens usuels de façon économe. Leur composition mène à la construction de *circuits de preuves* (définition 24), réseaux de preuves qui simulent en taille $\mathcal{O}(s^2)$ et en profondeur $\mathcal{O}(d)$ un circuit booléen de taille s et de profondeur d (théorèmes 6 et 7, page 22).

La fonction $UstCONN_2$ permet de déterminer s'il existe un chemin entre deux sommets dans un graphe non-dirigé de degré 2. Les résultats de Terui permettent de simuler avec des réseaux de preuves cette fonction, mais les réseaux ainsi obtenus sont de taille $\mathcal{O}(s^5)$. Nous affinons ainsi ces résultats tout en les adaptant aux petites classes de complexité.

Nous avons tenté de mieux cerner les classes de complexité encadrant l'espace logarithmique (c'est-à-dire NC^1 et AC^1) et plus particulièrement les classes de circuits en profondeur constante (NC^0 et AC^0). Ces classes de complexité – consommant très peu d'espace et de temps – présentent un interêt scientifique indéniable : *quels sont les calculs que nous pouvons effectuer avec un minimum de ressources ?* Nous avons ainsi laissé de côté les simulations des fonctions $MAJORITY$ et $PARITY$ que propose Terui, car ces fonctions n'appartiennent pas à AC^0 ([Furst *et al.*, 1984]).

L'article de Terui ne fait aucun cas de l'uniformité des classes de complexité qu'il étudie, nous avons tenté d'adapter ses résultats aux classes uniformes, suivant en cela [Mogbil et Rahli, 2007]. Leurs travaux prouvent que la traduction d'une famille uniforme de circuits booléens de $AC^i(USTCONN_2)$ vers une famille uniforme de réseaux booléens est *logspace*, nous précisons ce résultat en démontrant que la traduction de AC^i vers nos *circuits de preuves* est AC^0 (théorème 10, page 23).

Sommaire

1 Circuits booléens

La classe des circuits booléens est la classe des circuits arithmétiques sur l'anneau booléen. Les définitions 1 à 5 sont usuelles, on peut les trouver par exemple dans [Poizat, 1995] et dans [Greenlaw *et al.*, 1995]. Pour plus de clareté nous ne considérerons que les circuits de décision, mais prendre en compte les circuits d'arité sortante quelconque ne pose pas de problème.

Définition 1 (Fonctions booléennes)**.** Une fonction booléenne n-aire f^n est une application de $\{0,1\}^n$ dans $\{0,1\}$. Une famille de fonctions booléennes est une séquence $f = (f^n)_{n\in\mathbb{N}}$ telle que f^n est une fonction d'arité n. Une base est un ensemble de fonctions et de familles booléennes. On définit notamment $\mathfrak{B}_0 = \{\neg, \wedge^2, \vee^2\}$ et $\mathfrak{B}_1 = \{\neg, (\wedge^n)_{n\geqslant 2}, (\vee^n)_{n\geqslant 2}\}$. Étant données une base $\mathfrak{B}$ et une fonction ou une famille de fonctions f, on note $\mathfrak{B}(f)$ la base $\mathfrak{B}$ à laquelle a été ajoutée f.

Définition 2 (Circuits booléens)**.** Un circuit booléen C_n sur la base $\mathfrak{B}$ à n entrées et une sortie est un graphe orienté, fini, acyclique et étiqueté. On appele ses sommets des nœuds, et ils sont dotés d'une arité entrante et d'une arité sortante. Les nœuds d'arité entrante 0 sont les entrées du circuit booléen, ils sont étiquetés avec $x_1, \ldots, x_n$, 0, ou 1. Il n'y a qu'un nœud d'arité sortante 0, c'est la sortie du circuit booléen. Les nœuds d'arité entrante n sont étiquetés avec une fonction de $\mathfrak{B}$ d'arité n.

Un circuit booléen C_n accepte un mot $w = i_1 \ldots i_n \in \{0,1\}^n$ si lorsqu'on assigne $i_1, \ldots, i_n$ aux entrées de C_n étiquetées $x_1, \ldots, x_n$, la sortie est 1. C_n accepte un langage $X \subseteq \{0,1\}^n$ si pour tout $w \in \{0,1\}^n$ tel que $w \in X$, C_n accepte w.

Étant donné un circuit C_n, sa taille – notée $|C_n|$ – est son nombre de nœuds, sa profondeur – notée $p(C_n)$ – est la taille du plus long chemin (compté en nombre d'arêtes) entre une entrée et la sortie.

Etant donné qu'un circuit a son nombre d'entrées fixé, on doit pour calculer une fonction définir des *familles de circuits*.

Définition 3 (Famille de circuits)**.** Une famille de circuits est une séquence infinie $C = (C_n)_{n\in\mathbb{N}}$ de circuits booléens tels que C_n a n entrées.

Ainsi, deux circuits appartenant à une même famille ne calculent pas la même fonction. C'est pour cela que l'on considère le plus souvent que les circuits booléens sont des modèles non-uniformes de calcul. On impose – pour corriger cela – une condition d'uniformité aux familles de circuits : on admet en général que seules les familles uniformes peuvent être considérées comme l'implémentation d'un algorithme.

Définition 4 (Uniformité)**.** Une famille de circuits $C = (C_n)_{n\in\mathbb{N}}$ est uniforme s'il existe une machine de Turing déterministe qui étant donné n et le nom du nœud g peut déterminer toutes les informations sur le nœud g de C_n.

On ne considère que les fammiles de circuits de taille polynomiale en le nombre d'entrées du circuit.

Il existe de très nombreuses notions d'uniformité, selon les ressources qu'on alloue à cette machine de Turing : la P-uniformité ([Allender, 1989]), la L-uniformité ([Ruzzo, 1981]). Cette dernière uniformité – pourtant déjà très restrictive – est cependant encore trop coûteuse lorsqu'on étudie les classes de complexité sous-logarithmiques ([Barrington *et al.*, 1990]), c'est pourquoi nous considérerons la $DLOGTIME$-uniformité. La restriction imposée à la machine de Turing est qu'elle ne doit pas utiliser un temps supérieur à $\mathcal{O}(\log(s))$ pour donner la description d'un circuit de taille s. Les seules classes que nous considérons – sauf mention contraire – seront toujours uniformes selon le critère *ad hoc*.

Définition 5 (Classe de complexité NC^i (resp. AC^i))**.** Pour $i \in \mathbb{N}$, c'est l'ensemble des fonctions booléennes qui peuvent être calculées par des familles de circuits uniformes de profondeur $\mathcal{O}(\log^i(n))$, n étant le nombre d'entrées de C_n, sur la base $\mathfrak{B}_0$ (resp. $\mathfrak{B}_1$).

$AC^i(f)$ pour f une fonction booléenne ou une famille de fonctions booléennes correspond à AC^i sur la base $\mathfrak{B}_1 \cup \{f\}$.

De plus, $NC = \bigcup_{i\in\mathbb{N}} NC^i$ et $AC = \bigcup_{i\in\mathbb{N}} AC^i$.

Ainsi en particulier, pour AC^0 les circuits sont sur la base $\mathfrak{B}_1$, sans limite sur le nombre d'entrées, de taille polynomiale et de profondeur constante.

Les résultats suivants sont désormais bien connus :

Théorème 1 (Hierarchie)**.** *Pour tout* $k \geqslant 0$, $NC^k \subseteq AC^k \subseteq NC^{k+1}$.

$$AC^0 \subsetneq NC^1 \subseteq L \subseteq NL \subseteq AC^1$$

Afin de pouvoir manipuler les familles de circuits booléens, on définit un codage les décrivant totalement.

Définition 6 (Langage de Connexions Directes, [Vollmer, 1999])**.** Soit $C = (C_n)_{n\in\mathbb{N}}$ une famille de circuit sur la base $\mathfrak{B}$ et $bin(k)$ l'écriture en binaire de l'entier k. Le langage de connexions directes de C – noté $L_{DC}(C)$ – est l'ensemble de tous les 4-uplets $< y, g, p, b >$ où pour $|y| = n$, on a :

- g est le numéro de la porte v dans C_n,
- $p \in \{0,1\}^*$,
- si $p = \epsilon$ alors b est le numéro associé à la fonction appartenant à $\mathfrak{B}$ qui étiquète v,
- si $p = bin(k)$, alors b est le numéro de la k-ième porte prédécesseur de v (en respectant l'ordre des arêtes dans C_n).

Ainsi $L_{DC}(C)$ est un langage encodant la famille C. Si $L_{DC}(C)$ est récursif – et c'est le cas puisque les familles que nous considérons sont toujours uniformes – alors la machine acceptant ce langage est une description finie de l'objet infini C.

2 MLLu

Afin d'avoir un outil de mesure pertinent sur la mesure des réseaux, nous introduisons une variante de **MLL** où les connecteurs ne sont pas bornés, que l'on nomme **MLLu**. Nous employons ce système, qu'utilise [Terui, 2004], car les réseaux de preuves simulent de façon plus élégante et économique les circuits lorsque leurs connecteurs sont d'arités arbitraires.

2.1 Calcul des séquents

Définition 7 (Les formules de **MLLu**)**.** Pour α un atome quelconque et $n \geqslant 2$ on a en notation BNF :

$$A ::= \alpha \mid \otimes^n (A_1, \ldots, A_n) \mid \parr^n (A_n, \ldots, A_1) \mid \alpha^\perp$$

Les duaux étant définis selon les lois de De Morgan :

$$\begin{aligned} (A^\perp)^\perp &\equiv A \\ (\otimes^n(A_1,\ldots,A_n))^\perp &\equiv \parr^n(A_n^\perp,\ldots,A_1^\perp) \\ (\parr^n(A_n,\ldots,A_1))^\perp &\equiv \otimes^n(A_1^\perp,\ldots,A_n^\perp) \end{aligned}$$

Définition 8 (Les règles de **MLLu**)**.** Ce sont essentiellement les mêmes que celles de **MLL**, sauf qu'on autorise les connecteurs $\otimes$ et $\parr$ à avoir une arité arbitraire.

$$\frac{}{\vdash A, A^\perp}\, ax. \qquad \frac{\vdash \Gamma_1, A_1 \quad \ldots \quad \vdash \Gamma_n, A_n}{\vdash \Gamma_1, \ldots, \Gamma_n, \otimes^n(A_1,\ldots,A_n)}\, \otimes^n$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\, cut \qquad \frac{\vdash \Gamma, A_n, \ldots, A_1}{\vdash \Gamma, \parr^n(A_n,\ldots,A_1)}\, \parr^n$$

La règle d'échange n'est pas incluse dans ces règles, car on considère que les séquents sont des multi-ensemble de formules.

MLLu admet l'expansion des axiomes et l'élimination confluente des coupures, ce que l'on admet sans démonstration.

Convention 1. On ne notera pas toujours l'arité des connecteurs et on utilisera la notation infixée lorsque l'arité du connecteur sera de 2.

On s'autorise à employer les notations suivantes :

$$A \multimap B \equiv A^\perp \parr B \qquad \overrightarrow{A} \equiv A_1, \ldots, A_n \qquad \overleftarrow{A} \equiv A_n, \ldots, A_1$$

Dans la suite de ce travail, π designera toujours une dérivation de **MLLu**, A, B et D des formules de **MLLu**, et α un atome.

Définition 9 (Substitution)**.** On note $A[B/D]$ – ou simplement $A[D]$ si $B = \alpha$ – la formule A dans laquelle toutes les occurences de B sont remplacées par des occurences de D.

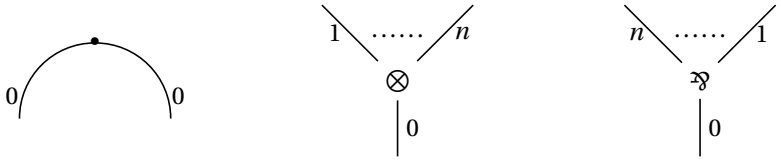


FIGURE 1 : Représentation des liens

2.2 Réseaux de preuves

À partir des preuves obtenues en calcul des séquents, on peut établir des réseaux de preuves en suivant une grammaire naturelle : une conclusion est représentée par une arête pendante, un axiome par un lien axiome, une application de la règle $⅋^n$ ayant pour formules principales $A_n,\dots,A_1$ par un lien $⅋^n$ dont les n arêtes entrantes sont connectées dans l'ordre aux arêtes reliées aux A_i et ayant une arête de sortie. Les applications de la règle $\otimes^n$ sont représentées de la même façon que les applications de la règle $⅋^n$, sauf que les n arêtes entrantes sont ordonnées dans l'autre sens.

Chaque lien a plusieurs ports : on nomme port principal un port correspondant à une conclusion, et port auxiliaire un port correspondant à une hypothèse. On associe des entiers naturels à ces ports : les liens axiomes ont deux ports principaux – que l'on numérote tous deux avec 0 – et n'ont pas de port auxiliaire. Les liens $\otimes^n$ (resp. $⅋^n$) ont n ports auxiliaires, numérotés avec $1,\dots,n$ (resp. $n,\dots,1$) et un unique port principal, numéroté avec 0.

Il n'y a pas de lien coupure : c'est une paire de ports principaux connectés l'un à l'autre par une arête.

On représente ces liens comme à la figure 1.

On pose quelques conventions de représentation des réseaux de preuves sur le plan. En prenant soin de toujours ordonner les ports des connecteurs $\otimes$ et $⅋$, et en représentant les ports principaux toujours en-dessous des liens, la numérotation des arêtes devient inutile.

Étant donné que les réseaux de preuves sont générés inductivement à partir de dérivations, il n'y a pas de critère de correction à proprement parler. Un réseau s'obtient inductivement à partir d'une dérivation, et un même réseau peut correspondre à plusieurs dérivations.

Les définitions et théorèmes de cette sous-section proviennent tous de [Terui, 2004]. On les a parfois explicités pour plus de clareté.

Définition 10 (Description d'un réseau de preuve)**.** On introduit une description séquentialisée des réseaux de preuves, que l'on définit par induction.

Il est possible d'associer à chaque port principal d'un lien une formule, que l'on numérote afin de distinguer les différentes occurences d'une même formule. On dira donc qu'un réseau est de conclusion Γ, pour Γ un multi-ensemble d'expressions du type $p : A$, où A est une formule et p un indice permettant de différencier les occurences de la formule A.

- Un nœud axiome est décrit par ax_p, p étant l'indice associé aux deux conclusions de ce réseau de preuve.
- Soient $2 \leqslant n$, $i \in \{1,\dots,n\}$ et n réseaux de preuves distincts de conclusion $\Gamma_i, p_i : A_i$ décrits par Q_i. Le réseau de preuve obtenu en reliant les arêtes des ports principaux étiquetées $p_1,\dots,p_n$ à un nœud tenseur étiqueté s est décrit par $tenseur_s^{p_1,\dots,p_n}(Q_1,\dots,Q_n)$.
- Soit un réseau de preuve de conclusion $\Gamma, p_n : A_n,\dots,p_1 : A_1$ décrit par Q. Le réseau de preuve obtenu en reliant les n arêtes des ports principaux étiquetées $p_n,\dots,p_1$ à un nœud par étiqueté s est décrit par $par_s^{p_n,\dots,p_1}(Q)$.
- Soient deux réseaux de preuves distincts de conclusions $\Gamma_1, p_1 : A$ et $\Gamma_2, p_2 : A^\perp$ décrits par Q_1 et Q_2. Le réseau de preuve obtenu en reliant les arêtes des ports principaux étiquetées p_1 et p_2 est décrit par $cut^{p_1,p_2}(Q_1,Q_2)$.

On obtient ainsi une description des réseaux de preuves qui est en bijection avec les dérivations, mais pas avec les réseaux. Par contre, il existe une injection des réseaux dans les descriptions : on note $\mathscr{D}(P)$ une des descriptions d'un réseau de preuve P.

À strictement parler, il faudrait que la fonction $\mathscr{D}(P)$ soit indicée par les numéros des ports conclusions de P. On se permet d'omettre ces indices, qui peuvent le plus souvent être facilement retrouvées depuis le contexte.

$$\frac{}{\vdash p:A, p:A^{\perp} \triangleright ax_p}\ ax. \qquad \frac{\vdash \Gamma_1, p_1:A_1 \triangleright P_1 \quad \dots \quad \vdash \Gamma_n, p_n:A_n \triangleright P_n}{\vdash \Gamma_1,\dots,\Gamma_n, s:\otimes^n(A_1,\dots,A_n) \triangleright tenseur_s^{p_1,\dots,p_n}(P_1,\dots,P_n)}\ \otimes^n$$

$$\frac{\vdash \Gamma, p:A \triangleright P \qquad \vdash \Delta, q:A^{\perp} \triangleright Q}{\vdash \Gamma,\Delta \triangleright cut^{p,q}(P,Q)}\ cut \qquad \frac{\vdash \Gamma, p_n:A_n,\dots,p_1:A_1 \triangleright P}{\vdash \Gamma, s:\parr^n(A_n,\dots,A_1) \triangleright parr_s^{p_n,\dots,p_1}(P)}\ \parr^n$$

FIGURE 2 : Les règles de typage

Définition 11 (Typage d'une dérivation, jugement)**.** Un jugement Π est de la forme $\vdash \Gamma \triangleright \mathscr{D}(P)$, où $\mathscr{D}(P)$ est la description d'un réseau P et Γ est comme décrit précedemment. Les règles de typage sont données à la figure 2.

On peut noter que les deux formules introduites par un axiome ont le même indice, tout comme les deux ports principaux d'un lien axiome portent le même numéro.

Un réseau de preuve P de type $\vdash \Gamma$ est un réseau tel que $\vdash \Gamma \triangleright \mathscr{D}(P)$ est dérivable par ces règles. Puisque les règles de typage coincident avec les règles de dérivation de **MLLu**, on est assuré que toute dérivation se type naturellement et induit ainsi un jugement. Par ailleurs, on est assuré que les réseaux décrits par jugement sont séquentialisables.

Convention 2. Afin de gagner en lisibilité, on s'autorise par exemple à remplacer

$$\frac{\dfrac{\dfrac{}{\vdash u_1:C_1, u_1:C_1^{\perp} \triangleright ax_{u_1}}\ ax. \quad \dots \quad \dfrac{}{\vdash u_n:C_n, u_n:C_n^{\perp} \triangleright ax_{u_n}}\ ax.}{\vdash q:\otimes^n(C_1,\dots,C_n), u_1:C_1^{\perp},\dots,u_n:C_n^{\perp} \triangleright tenseur_q^{u_1,\dots,u_n}(ax_{u_1},\dots,ax_{u_n})}\ \otimes^n}{\vdash q:\otimes^n(C_1,\dots,C_n), r:\parr^n(C_n^{\perp},\dots,C_1^{\perp}) \triangleright par_r^{u_n,\dots,u_1}(tenseur_q^{u_1,\dots,u_n}(ax_{u_1},\dots,ax_{u_n}))}\ \parr^n$$

par

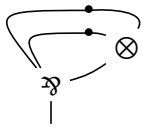
$$\frac{\dfrac{\dfrac{}{\vdash \{u_i:C_i, u_i:C_i^{\perp}\}_{i\in\{1,\dots,n\}} \triangleright ax_{\overrightarrow{u_n}}}\ ax.^n}{\vdash q:\otimes^n(\overrightarrow{C}), \{u_i:C_i^{\perp}\}_{i\in\{1,\dots,n\}} \triangleright tenseur_q^{\overrightarrow{u}}(ax_{\overrightarrow{u_n}})}\ \otimes}{\vdash q:\otimes^n(\overrightarrow{C}), r:\parr(\overleftarrow{C}) \triangleright par_r^{\overleftarrow{u}}(tenseur_q^{\overrightarrow{u}}(ax_{\overrightarrow{u_n}}))}\ \parr$$

On se servira de cette convention lors de la dérivation du jugement de la composition, à la figure 4, page 14.

Exemple 1. Il est inutile dans un jugement d'expliciter les règles d'échanges, car toutes les informations nécessaires sont données par le typage :

$$\frac{\dfrac{\dfrac{}{\vdash p:\alpha^{\perp}, p:\alpha \triangleright ax_p}\ ax. \qquad \dfrac{}{\vdash q:\alpha^{\perp}, q:\alpha \triangleright ax_q}\ ax.}{\vdash p:\alpha^{\perp}, q:\alpha^{\perp}, r:\alpha\otimes\alpha \triangleright tenseur_r^{p,q}(ax_p, ax_q)}\ \otimes^2}{\vdash s:\parr^3(\alpha^{\perp}, \alpha^{\perp}, \alpha\otimes\alpha) \triangleright par_s^{q,p,r}(tenseur_r^{p,q}(ax_p, ax_q))}\ \parr^3$$

$$\frac{\dfrac{\dfrac{}{\vdash p:\alpha^{\perp}, p:\alpha \triangleright ax_p}\ ax. \qquad \dfrac{}{\vdash q:\alpha^{\perp}, q:\alpha \triangleright ax_q}\ ax.}{\vdash p:\alpha^{\perp}, q:\alpha^{\perp}, r:\alpha\otimes\alpha \triangleright tenseur_r^{p,q}(ax_p, ax_q)}\ \otimes^2}{\vdash s:\parr^3(\alpha^{\perp}, \alpha^{\perp}, \alpha\otimes\alpha) \triangleright par_s^{p,q,r}(tenseur_r^{p,q}(ax_p, ax_q))}\ \parr^3$$

L'échange porte ici sur les deux premières prémisses du $\parr^3$ introduit par la dernière règle. Les deux réseaux induits se représentent respectivement  et  .

Remarque 1. À un même réseau peuvent bien évidemment correspondre plusieurs représentations : le graphe représentant un réseau et le réseau sont deux choses distinctes.

Cependant, nous allons associer de façon canonique à chaque réseau une représentation unique. Lever toute ambigüité quant à la représentation d'un réseau est en effet essentiel : l'une des principales astuces de K. Terui consiste à prendre en compte le fait qu'un réseau est planaire ou pas. Les réseaux représentant les valeurs booléennes 0 et 1 – qui sont en réalité les réseaux donnés à l'exemple 1 – seront de même conclusion, mais organisés différement dans l'espace.

On pourrait, pour ce faire, définir une relation d'équivalence sur les réseaux de preuves qui les rende aussi planaires que possible. Il faudrait ensuite identifier les réseaux qui ne sont qu'inessentiellement différents : nous ne sommes par exemple pas sensibles au fait qu'une arête croise une autre arête à droite ou à gauche d'un lien axiome.

On ne le fait pas car de telles définitions seraient fastidieuses et d'un interêt relatif, mais on considère désormais que si deux réseaux de preuves sont en relation d'équivalence avec un même réseau de preuve ayant aussi peu de croisements entre ses arêtes que possible, alors ces réseaux sont identiques.

On ne prend ainsi pas en considération les variations inessentielles entre deux réseaux de preuves, et on considère toujours qu'ils sont aussi planaires que possibles.

Définition 12 (Profondeurs)**.** Soient π une dérivation de $\vdash A_1, \ldots, A_n$ et P un réseau de preuve.

— La profondeur d'une formule est définie par induction :

$d(\alpha) = d(\alpha^\perp) = 1$

$d(\otimes^n(A_1, \ldots, A_n)) = d(\parr^n(A_n, \ldots, A_1)) = max(d(A_1), \ldots, d(A_n)) + 1$

— La profondeur de π – que l'on note $d(\pi)$ – est la profondeur maximale d'une formule de coupure.

— Sa profondeur totale $d'(\pi)$ est $max(d(\pi), d(A_1), \ldots, d(A_n))$.

— La profondeur $d(P)$ d'un réseau de preuve P est

$$min\{d(\pi) | \pi \text{ induit un jugement de } \vdash \Gamma \rhd \mathscr{D}(P) \text{ pour un } \Gamma \text{ quelconque}\}$$

— La profondeur totale de P est

$$min\{d'(\pi) | \pi \text{ induit un jugement de } \vdash \Gamma \rhd \mathscr{D}(P) \text{ pour un } \Gamma \text{ quelconque}\}$$

— La taille de P – que l'on note $|P|$ – est le nombre de liens dans P.

Théorème 2 ([Terui, 2004])**.** *Soient A et B des formules de **MLLu**, on a $d(A[B/\alpha]) \leqslant d(A) + d(B) - 1$*

Démonstration. Puisque B est une formule de **MLLu**, $d(B) \geqslant 1$. On raisonne par cas :

— Si α n'occure pas dans A, alors $d(A[B/\alpha]) = d(A) \leqslant d(A) + d(B) - 1$

— Si α occure dans A, prenons le pire des cas et supposons que α occure dans A à profondeur maximale. On pose $d(A) = d(\alpha) + n$ pour $n \geqslant 0$, après substitution on a $d(A[B/\alpha]) = d(B) + n = d(A) + d(B) - 1$.

□

2.3 Élimination des coupures

Définition 13 (a-coupure, m-coupure)**.** Les coupures dans un réseau de preuve ne sont créées que par le fait de relier deux liens par leurs ports principaux. Si l'un de ces deux liens est un lien axiome, on dit que la coupure est une a-coupure, sinon que c'est une m-coupure. S'il existe une m-coupure entre deux liens et que l'un de ces liens est $\otimes^n$ (resp. $\parr^n$), on est assuré que l'autre lien est $\parr^n$ (resp. $\otimes^n$).

Définition 14 ($\rightarrow_m$ et $\rightarrow_a$)**.** Soit $\circ \in \{\bullet, (\parr^n)_{n\geqslant 2}, (\otimes^n)_{n\geqslant 2}\}$, on définit à la figure 3 deux règles de réécriture sur les réseaux de preuves.

Soient P et Q deux réseaux de preuves, on écrit $P \rightarrow Q$ s'il est possible d'obtenir Q à partir de P en appliquant $\rightarrow_m$ ou $\rightarrow_a$. On définit $\rightarrow^*$ comme étant la clotûre reflexive et transitive de $\rightarrow$.

Étant donné que les coupures sont exprimées par des arêtes, on est assuré que le nombre de liens décroît strictement à chaque application de l'une de ses règles.

Théorème 3 ([Girard, 1987], [Terui, 2004])**.** *Pour tout réseau de preuve P, P se réduit en un unique réseau sans coupure en au plus $|P|$ étapes de réduction. De plus, si $\vdash \Gamma \rhd \mathscr{D}(P)$ et $P \rightarrow^* Q$, alors $\vdash \Gamma \rhd \mathscr{D}(Q)$.*

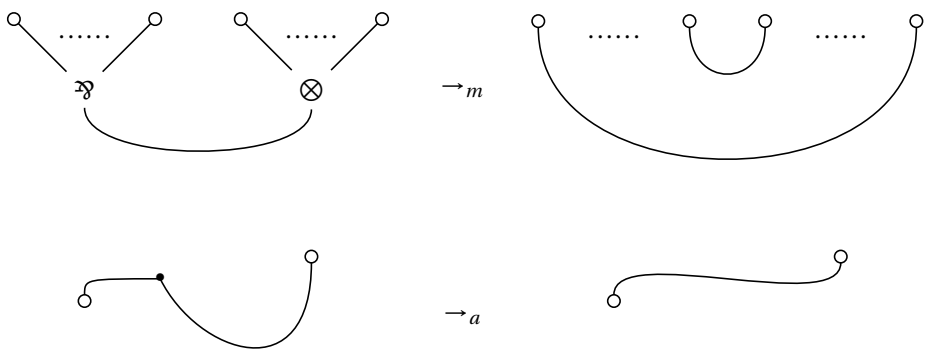


FIGURE 3 : m-réduction et a-réduction

3 Réseaux booléens

3.1 Premières définitions

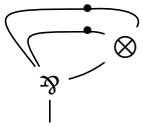
Définition 15 (Le type booléen, 0 et 1, [Terui, 2004])**.** Le type booléen **B** est défini comme étant

$$\parr^3(\alpha^\perp, \alpha^\perp, \alpha \otimes \alpha)$$

Il s'agit d'une convention inspirée du fait qu'en λ-calcul simplement typé, les valeurs booléennes sont représentées par des termes de type $\alpha \to \alpha \to \alpha$. Il n'est pas possible de prouver en **MLLu** son équivalent linéaire $(\alpha \multimap \alpha \multimap \alpha)$ en raison de l'absence d'affaiblissement.

Au renommage près, et si on ne tient pas compte des variations inessentielles dans les réseaux de preuve, il n'y a que deux réseaux de preuves de type **B**, on les nomme b_0 et b_1. On a donné leurs jugements à l'exemple 1, page 9, on les note $\vdash s : \mathbf{B} \rhd \mathscr{D}(b_0)$ et $\vdash s : \mathbf{B} \rhd \mathscr{D}(b_1)$.

$$\mathscr{D}(b_0) = par_s^{q,p,r}(tenseur_r^{p,q}(ax_p, ax_q)) \qquad b_0 \equiv$$



$$\mathscr{D}(b_1) = par_s^{p,q,r}(tenseur_r^{p,q}(ax_p, ax_q)) \qquad b_1 \equiv$$

b_0 représente 0, le faux, et b_1 1, le vrai : leur différence vient du fait que b_0 soit planaire et pas b_1.

Définition 16 (Réseau de preuve booléen, [Terui, 2004])**.** Un réseau de preuve booléen à n entrées est un réseau de preuve $P(\overrightarrow{p})$ de type

$$\vdash p_1 : \mathbf{B}^\perp[A_1], \ldots, p_n : \mathbf{B}^\perp[A_n], s : \otimes^{m+1}(\mathbf{B}[A], C_1, \ldots, C_m)$$

Étant donnés $b_{i_1}, \ldots, b_{i_n} \equiv \overrightarrow{b}$, pour $i_j \in \{0,1\}$ et $1 \leqslant j \leqslant n$, $P(\overrightarrow{b})$ est défini comme le réseau de preuve obtenu en introduisant des coupures entre b_{i_j} et p_j.

$P(\overrightarrow{b})$ se réduit *via* l'élimination des coupures à un réseau de preuve de type $\otimes^{m+1}(\mathbf{B}[A], C_1, \ldots, C_m)$. En omettant les indices et les exposants, ce réseau peut être décrit par $tenseur(\mathscr{D}(b_i), Q_1, \ldots, Q_m)$, pour $i \in \{0,1\}$. On dit alors que $P(\overrightarrow{b})$ s'évalue en b_i, et on note $P(\overrightarrow{b}) \to_{ev.} b_i$.

Un réseau de preuve booléen $P(\overrightarrow{b})$ représente une fonction booléenne $f : \{0,1\}^n \to \{0,1\}$ si pour tout $w \equiv i_1 \ldots i_n \in \{0,1\}^n$, $P(b_{i_1}, \ldots, b_{i_n})$ s'évalue en $b_{f(w)}$.

En s'inspirant des définitions 2 et 3, page 6, on définit les familles de réseaux de preuves et les conditions sous lesquelles elles acceptent un mot ou un ensemble.

Remarque 2. Puisque nous employons des réseaux de preuves en logique linéaire multiplicative, nous n'avons pas la possibilité d'effacer les liens qui ne correspondent pas au résultat de l'évaluation après l'élimination des coupures. Nous devons ainsi collecter à la fin de notre réseau booléen le résultat de notre évaluation, ainsi que tous les sous-réseaux désormais inutiles, que nous appelons les *poubelles*. C'est le rôle du tenseur étiqueté s dans la définition précédente : nous le nommons désormais le tenseur conclusion. Par convention son premier port auxiliaire est connecté au résultat attendu, et l'ordre dans lequel sont connectées ensuite les poubelles n'a pour le moment aucune importance.

De plus il ne nous est pas possible de dupliquer des formules, alors que l'arité sortante d'un nœud d'un réseau booléen peut être arbitraire. Nous comblerons cette différence en introduisant un moyen de dupliquer un résultat à la section 4.1, page 15.

Définition 17 (APN, [Terui, 2004])**.** Pour $i \in \mathbb{N}$, APN^i est la classe des langages reconnus par des familles de réseaux de preuves non-uniformes de taille polynomiale et de profondeur $\mathscr{O}(log^i(n))$.

$APN = \bigcup_{i\in\mathbb{N}} APN^i$.

Définition 18 (mBN^i, [Mogbil et Rahli, 2007])**.** La classe de complexité APN^i uniforme est appelée mBN^i.

Afin de pouvoir manipuler les réseaux avec des circuits, on définit un codage permettant de décrire univoquement une famille de réseaux. On adapte pour cela le langage de connexions directes des circuits booléens – tel que posé à la définition 6 – aux réseaux de preuves.

Définition 19 (Codage des réseaux [Vollmer, 1999])**.** Soit $P = (P_n)_{n\in\mathbb{N}}$ une famille de réseaux booléens. Le langage de connexions directes de P – noté $L_{DC}(P)$ – est l'ensemble de tous les 4-uplets $< y, g, p, b >$ où pour $|y| = n$ on a :

- g est le numéro du lien v dans P_n,
- $p \in \{0,1\}^*$,
- si $p = \epsilon$ alors b est le numéro associé au type de lien [1] qui étiquète v,
- si $p = bin(k)$, alors b est le numéro du k-ième lien prédécesseur de v (en respectant l'ordre des arêtes dans P_n) : le port principal du lien numéroté b est relié au port auxiliaire numéroté k de v. Si $k = 0$, alors b est le numéro du lien coupé avec v : leurs port principaux sont reliés.

Puisque nous n'avons pas pour le moment de contrainte d'uniformité sur les familles de réseaux de preuve, nous n'avons aucune garantie que $L_{DC}(P)$ soit récursif.

On commet un abus sans conséquence en ne distinguant pas les deux ports principaux d'un axiome : on pourra très bien, si nécessaire, modifier légèrement le codage afin de préciser quel est le port principal dont il est question.

3.2 Représentablilité selon Terui

Les outils mis en place dans l'article de K. Terui constituent une réelle innovation dont on s'inspire largement. On introduit ici quelques réseaux que nous reprendrons dans la suite de notre exposé, et notamment la façon dont est représentée la fonction conditionnelle.

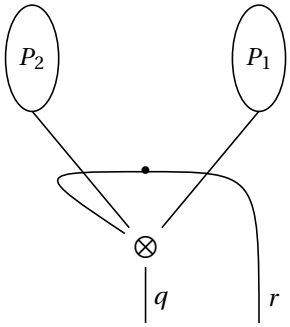
Définition 20 (COND)**.** Soient deux jugements $\vdash \Gamma, p_1 : A \triangleright \mathscr{D}(P_1)$ et $\vdash \Delta, p_2 : A \triangleright \mathscr{D}(P_2)$. On construit le jugement

$$\vdash \Gamma, \Delta, q : B[A]^{\perp}, r : A \otimes A \triangleright tenseur_q^{r,p_2,p_1}(ax_r, \mathscr{D}(P_2), \mathscr{D}(P_1))$$

On convient de nommer $cond_r^{p_1,p_2}[\mathscr{D}(P_1), \mathscr{D}(P_2)](q)$ la description de ce réseau de preuve.

On représente ainsi (en omettant les arêtes relatives à Δ et Γ) le réseau de preuve qui provient de ce jugement :

1. Il n'est pas utile de préciser l'arité dans le cas ou ce lien est un connecteur : cette information peut se retrouver à partir du nombre de liens prédecesseurs de v.



On a bien $cond[\mathscr{D}(P_1),\mathscr{D}(P_2)](b_0) \to_{ev} P_2$ et $cond[\mathscr{D}(P_1),\mathscr{D}(P_2)](b_1) \to_{ev} P_1$, ce réseau de preuve représente la fonction «conditionnel». Il convient de remarquer que les réseaux b_0 et b_1 que l'on peut couper avec l'arête étiquetée q doivent être de type $\mathbf{B}[A]$.

La profondeur totale de ce réseau de preuve est au pire $max(d'(P_1),d'(P_2))+2$, cette borne étant atteinte si $d'(P_1)=d(A)$ ou $d'(P_2)=d(A)$. Sa taille est $|P_1|+|P_2|+2$.

Le conditionnel nous servira à représenter les fonctions disjonction, conjonction, mais aussi à dupliquer des valeurs. C'est le fondement de la représentation des fonctions booléennes par des réseaux.

Cependant, comme on le voit, la formule attachée à la sortie de ce réseau est $A \otimes A$, il nous faut donc définir un moyen de composer les réseaux.

Définition 21 (Composition)**.** Soient $\Gamma \equiv p'_1 : A'_1,\dots,p'_n : A'_n$ et $\Delta \equiv q'_1 : B'_1,\dots,q'_m : B'_m$, et deux jugements :

$$\vdash \Gamma, p : \otimes^{n+1}(\mathbf{B},\overrightarrow{C}) \rhd \mathscr{D}(P(\overrightarrow{p'})) \quad \text{et} \quad \vdash q : \mathbf{B}^{\perp}[A],\Delta,r : \otimes^{m+1}(\mathbf{B},\overrightarrow{D}) \rhd \mathscr{D}(Q(\overrightarrow{q'}))$$

On montre ici comment composer ces deux réseaux de preuves afin d'obtenir un jugement

$$\vdash \Gamma[A],\Delta,s : \otimes(\mathbf{B},\overrightarrow{D},\overrightarrow{C[A]}) \rhd comp_s^{p,q,r}[\mathscr{D}(P),\mathscr{D}(Q)](\overrightarrow{p'},\overrightarrow{q'})$$

On donne le jugment de la composition en figure 4. On y a convenu que $n+m+1=e$, et il importe de remarquer qu'une substitution a été effectuée dans le jugement de $P(\overrightarrow{p'})$.

Il est nécessaire d'introduire autant d'axiomes qu'il y a de formules dans les poubelles de $P[A]$ et de Q : la composition se fait au cas par cas, il faut connaître la taille des poubelles des réseaux pour les composer. Si on reprend notre exemple, et que $\overrightarrow{D}$ est composé de m formules et $\overrightarrow{C}$ de n formules, on a le réseau de preuve donné en figure 5, sur lequel les arêtes de Γ et de Δ ne sont pas représentées.

$d'(P[A]) \leqslant d'(P)+d(A)-1$ par le théorème 2, donc ce réseau est de profondeur totale bornée par $max(d'(P)+d(A)-1,d'(Q))$ et de taille $|P|+|Q|+4+e$.

Terui utilise cette composition pour définir la conjonction et la disjonction n-aires, mais cette façon de procéder présente une certaine lourdeur : puisqu'on collecte les poubelles à la fin de chaque composition, il est nécessaire de les re-traiter au début de chaque composition.

Dans ce qui suit, nous définissions un moyen d'«internaliser» la composition, si bien que chaque *pièce* que nous allons définir prend en entrée directement la valeur booléenne attendue, envoit ses poubelles vers le tenseur conclusion au fur et à mesure, et donne en sortie uniquement la valeur booléenne résultat.

4 Circuits de preuve

Nous définissons des pièces représentant une valeur ou une fonction booléenne, ou dupliquant une valeur, qui peuvent se composer plus simplement. Les poubelles générées par ces pièces sont directement envoyées vers le tenseur conclusion.

Ainsi, on ne capture pas toutes les façons de calculer une fonction booléenne avec des réseaux de preuve, mais on capture la traduction des circuits booléens sur $\mathfrak{B}_0$ et $\mathfrak{B}_1$. On obtient un moyen aisé de raisonner sur ces réseaux et de borner leurs complexités.

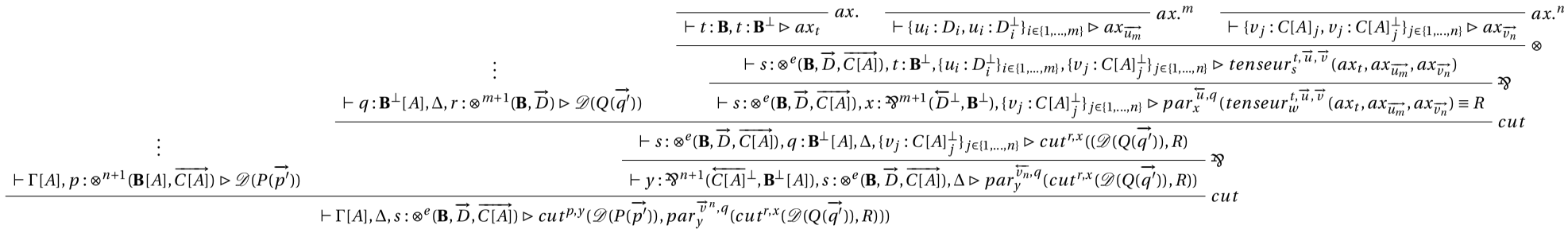


FIGURE 4 : Jugement de la composition

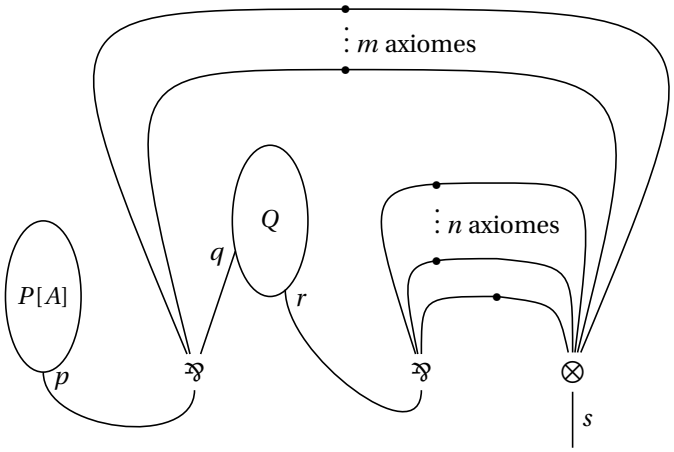


FIGURE 5 : La composition de deux réseaux de preuves booléens

Convention 3. On dessine désormais les arêtes représentant des poubelles ainsi : 

Toute arête de ce type est connectée par la droite à une arête portant un indice supérieur à 2 du tenseur conclusion. Ces arêtes sont connectées au fur et à mesure de leur apparition au tenseur conclusion, en suivant la numérotation des arêtes et en commençant par 2.

4.1 Pièces

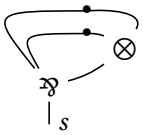
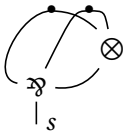
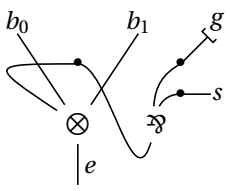
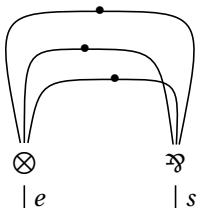
Définition 22. Une pièce $\mathscr{P}$ à $i \geqslant 0$ entrées, $j \geqslant 1$ sorties et $k \geqslant 0$ poubelles est un des pré-réseaux de preuves qui figure dans le tableau 1, où i arêtes sont étiquetées $e_1, \ldots, e_i$, j arêtes sont étiquetées $s_1, \ldots, s_j$ et k arêtes pointent vers le tenseur conclusion. Toutes ces arêtes correspondent à des ports principaux, et ce pseudo-réseau est de type

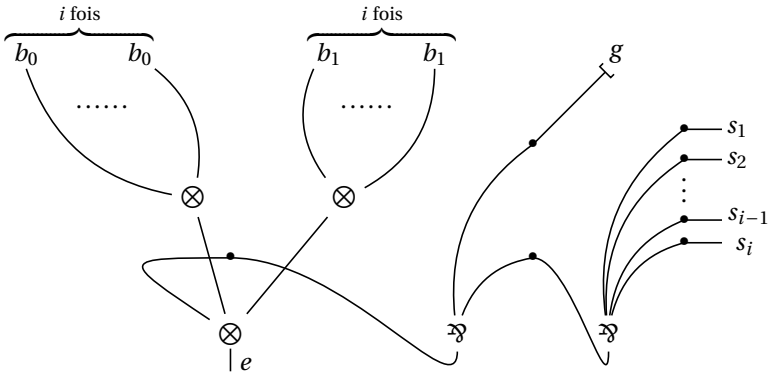
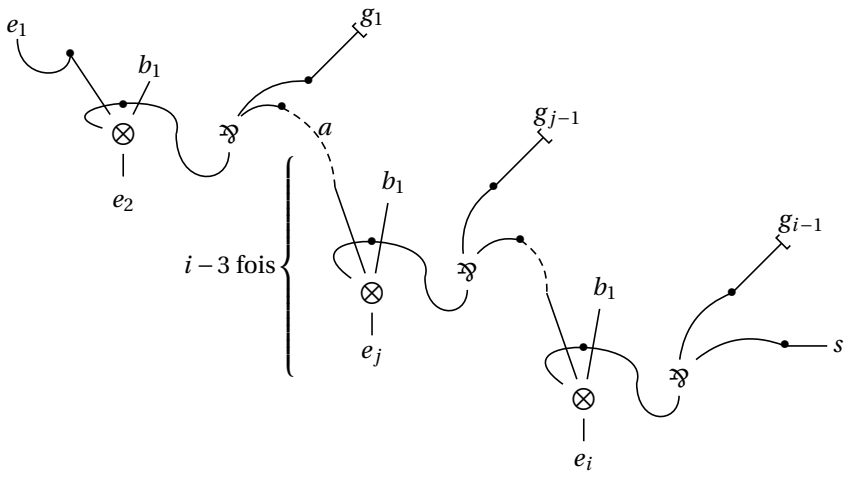
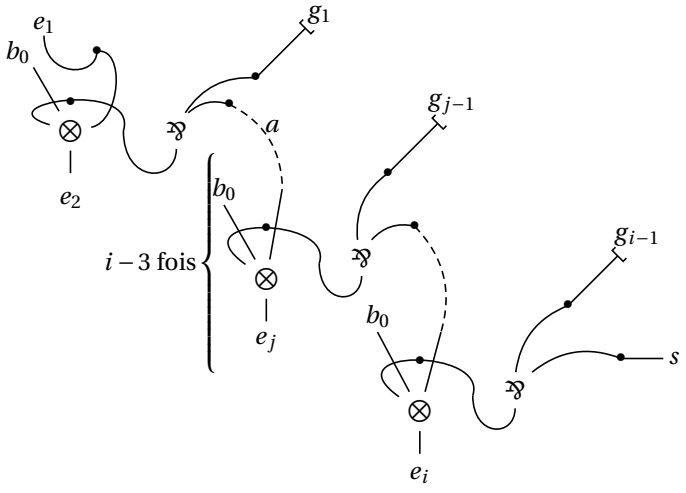
$$\vdash e_1 : \mathbf{B}^{\perp}[A_1], \ldots, e_i : \mathbf{B}^{\perp}[A_i], s_1 : \mathbf{B}[A'_1], \ldots, s_j : \mathbf{B}[A'_j], g_1 : \mathbf{B}[A''_1], \ldots, g_k : \mathbf{B}[A''_k]$$

On a donc $\mathscr{P} \in \{b_0, b_1, NEG, \{DUPL^i\}_{i\geqslant 1}, \{DISJ^i\}_{i\geqslant 2}, \{CONJ^i\}_{i\geqslant 2}\}$.

Tableau 1: Les pièces

On pose $2 \leqslant j \leqslant i$. Les arêtes étiquetées b_k – pour $k \in \{0,1\}$ – sont reliées au port s de la pièce b_k.

Nom	Usage	Réseau de preuve	Nom	Usage	Réseau de preuve
b_0	Représente la constante 0		b_1	Représente la constante 1	
$DUPL^1$	Représente la fonction identité		NEG	Représente la fonction négation	

Nom	Usage	Réseau de preuve
$DUPL^i$	Duplique une valeur i fois	
$DISJ^i$	Représente la fonction disjonction d'arité i	 Si $i = 2$, l'arête a est identique à l'arête s.
$CONJ^i$	Représente la fonction conjonction d'arité i	 Si $i = 2$, l'arête a est identique à l'arête s.

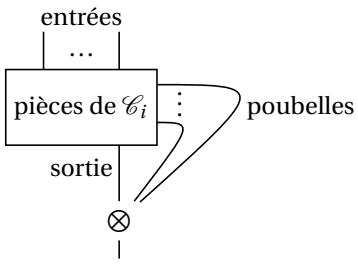
Convention 4. On s'autorise à nommer entrées (resp. sorties, poubelles) les arêtes étiquetées e (resp. s, g) d'une pièce. On dit qu'une arête est libre si elle n'est pas identifiée avec une autre arête.

4.2 Les circuits de preuve vus depuis les réseaux

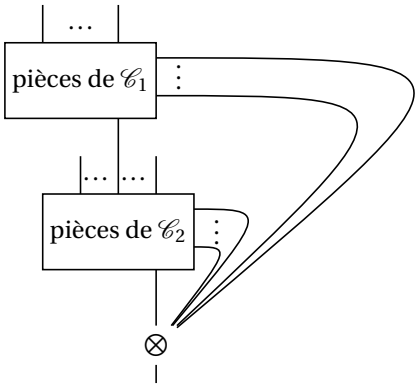
Définition 23 (Composition des pièces)**.** Pour composer deux pièces, il suffit d'identifier une des sortie d'une pièce avec une entrée d'une autre pièce. Il n'est pas possible de boucler (identifier une sortie et une entrée d'une même pièce).

Définition 24 (Circuit de preuve)**.** Un circuit de preuve $\mathscr{C}_n$ à $n \geqslant 0$ entrées et 1 sortie est un pré-réseau de preuve [2] qui ne compte qu'une seule arête sortie libre, et qui est obtenu en composant des pièces puis en étiquetant $x_1, \ldots, x_n$ les n entrées des pièces qui sont restées libres. Si les pièces employées génèrent des poubelles, il faut ajouter un tenseur conclusion, identifier l'arête de son premier port auxiliaire avec l'arête sortie et ses autres ports auxiliaires avec les arêtes poubelles. Sinon, on ajoute une pièce $DUPL^1$ – qui représente la fonction identité – à la sortie avant de gérer les poubelles [3].

Définition 25 (Composition de circuits de preuves)**.** Soient $\mathscr{C}_1$ et $\mathscr{C}_2$ deux circuits de preuves. On les représente ainsi :



Pour les composer, il suffit d'enlever le tenseur conclusion de $\mathscr{C}_1$, d'identifier la sortie de $\mathscr{C}_1$ avec l'entrée de $\mathscr{C}_2$ souhaitée, et de connecter les poubelles de $\mathscr{C}_1$ au tenseur conclusion de $\mathscr{C}_2$. On obtient alors :



On vérifie facilement que la structure de preuve ainsi obtenue est bien un circuit de preuve. Pour simplifier, si la dernière pièce de $\mathscr{C}_1$ était $DUPL^1$, alors on ôte cette pièce et on connecte son entrée à l'entrée de $\mathscr{C}_2$ souhaitée.

4.3 Traduction des circuits booléens vers les circuits de preuves

Lemme 1 (Traduction)**.** *Pour $i \in \mathbb{N}$, il est possible d'associer à tout circuit booléen C_n appartenant à une famille de circuits booléens incluse dans AC^i ou NC^i un circuit de preuve tel que le résultat de l'évaluation par élimination des coupures de ce circuit donne le réseau représentant le résultat de l'évaluation de C_n. On note $\mathscr{T}(C_n)$ le circuit de preuve ainsi obtenu, il compte au plus $2 \times |C_n|$ pièces.*

2. On montrera au théorème 5, page 20, qu'il s'agit en réalité de réseaux de preuves booléens.
3. Ainsi, le type d'un circuit de preuve sera toujours celui d'un réseau booléen.

Démonstration. La traduction est évidente : à tout nœud de C_n étiqueté par un connecteur v d'arité i de $\mathfrak{B}_0$ ou de $\mathfrak{B}_1$, on associe la pièce qui simule v d'arité i. Si l'arité sortante de ce nœud est $m > 1$, on connecte une pièce $DUPL^m$ à sa sortie. Si le circuit booléen est une entrée ou une constante niée plusieurs – ou 0 – fois, on ajoute une pièce $DUPL^1$ à la sortie de son traduit.

On n'associe aucune pièce aux nœuds correspondant aux entrées – sauf si celles-ci sont dupliquées – et le nœud sortie du circuit booléen est associé au tenseur conclusion.

En supposant qu'on maximise toutes les duplications (c'est-à-dire qu'on s'interdit de connecter une pièce $DUPL$ à une autre pièce $DUPL$), on obtient une injection des circuits booléens vers les circuits de preuve.

À tout nœud de C_n est associé au plus deux pièces, ce cas se présentant si tous les nœuds de C_n sont d'arité sortante supérieure à 1. □

On précisera aux théorèmes 6 et 7, page 22, les contraintes sur la profondeur et la taille du traduit ainsi obtenu. On verra également au théorème 10 que l'algorithme de traduction est dans AC^0. On transpose désormais la notion de profondeur d'un nœud d'un circuit booléen aux circuits de preuve.

Définition 26 (Hauteur d'une pièce)**.** Soit $\mathscr{P}$ une pièce, on note $h(\mathscr{P})$ sa hauteur, que l'on définit ainsi :

La pièce dont la sortie est connectée au tenseur conclusion est nécessairement unique, elle est de hauteur nulle.

Une pièce à une sortie connectée à l'entrée d'une pièce de hauteur n a une hauteur de $n+1$.

Une pièce $DUPL^n$ dont les sorties sont connectées à n pièces de hauteurs $i_1,\ldots,i_n$ a une hauteur de $max(i_1,\ldots,i_n)+1$.

On utilisera cette mesure notamment pour démontrer le théorème 5, page 20.

5 Contractibilité parallèle

Afin de démontrer que tout circuit de preuve est un réseau de preuve, on parallélise les règles de réécriture du critère de contractibilité, tel qu'il a été inventé par [Danos, 1990]. Le critère qu'on emploie s'inspire des *parsing boxes* telles qu'introduites par [Lafont, 1995], et reprend les améliorations proposées par [Guerrini et Masini, 2001].

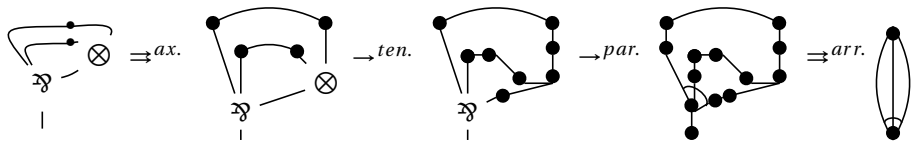
On rappelle ces règles de contraction adaptées à **MLLu** à la figure 6.

Ces règles sont des règles de réécriture sur des graphes. Ainsi, $\rightarrow^{arr.}$ convient aussi bien pour contracter les coupures : les liens se transforment toujours en points, et une arête entre deux ports principaux de liens, transformés en points, se contractera avec $\rightarrow^{arr}$.

Soit $\rightarrow^r$ une de ces règles, on note $\Rightarrow^r$ l'application en parallèle de cette règle : on applique la règle $\rightarrow^r$ autant de fois que possible et simultanément. Il n'y a pas de risque de chevauchement ou de paire critique, car ces règles sont confluentes. Par contre, il peut exister des points issus de liens par qui ne peuvent être contractés que dans un certain ordre : on explicitera cette relation de dépendance lors de la remarque 3.

Théorème 4 ([Danos, 1990])**.** *Un pré-réseau de preuve est un réseau de preuve si et seulement si il se réduit en un point par application des règles de la figure 6.*

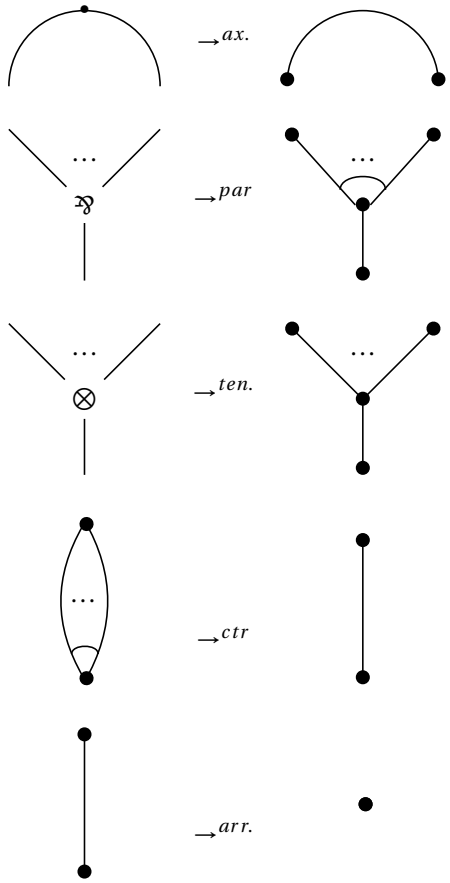
Définition 27 ($\Rightarrow^b$)**.** b_0 tout comme b_1 se réduit facilement en un point. On se contente de le vérifier pour b_0 :



Après une application de $\rightarrow^{ctr.}$ et de $\rightarrow^{arr.}$ on obtient le résultat espéré, à savoir un point.

On définit $\Rightarrow^b$ comme étant la réduction en parallèle de toutes les pièces b_0 et b_1 d'un circuit de preuve en points.

Lemme 2. *Toute pièce dont les sorties et les poubelles sont connectées à un point se réduit en un point.*



Pour les règles $\to_{ctr}$ et $\to_{arr.}$, il est nécessaire que les points soient distincts.

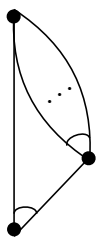
FIGURE 6 : Les règles de contractibilité

Démonstration. Afin de facilier la démonstration, on nomme $\mathscr{P}^*$ la pièce $\mathscr{P}$ où les sorties et les poubelles ont été connectées à un point. On procède par cas :

b_0 **et** b_1 Après application de $\to^b$ et de $\to^{arr.}$, b_0^* tout comme b_1^* se contracte en un point.

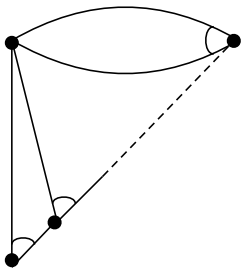
NEG Après application de $\rightrightarrows^{ax.}$, $\to^{ten.}$, $\to^{par.}$, $\rightrightarrows^{arr.}$, $\to^{ctr.}$ et $\to^{arr.}$, NEG^* se réduit en un point.

$DUPL^i$ Après application de $\rightrightarrows^b$, $\rightrightarrows^{ax.}$, $\rightrightarrows^{ten}$, $\rightrightarrows^{arr.}$, $\rightrightarrows^{par.}$ puis $\rightrightarrows^{arr.}$, $DUPL^{i*}$ se réduit en



Appliquer $\to^{ctr.}$, $\to^{arr.}$, $\to^{ctr.}$ et $\to^{arr.}$ permet d'obtenir le résultat souhaité. $DUPL^{i*}$ se réduit donc en un nombre d'étapes indépendant de i.

$CONJ^i$ **et** $DISJ^i$ Après application de $\rightrightarrows^b$, $\rightrightarrows^{ax.}$, $\rightrightarrows^{ten}$, $\rightrightarrows^{par}$ et $\rightrightarrows^{arr.}$, $CONJ^i$ aussi bien que $DISJ^i$ se réduit en



Après $i-1$ alternances de $\rightarrow^{ctr.}$ et $\rightarrow^{arr.}$ on obtient le résultat souhaité.

□

Théorème 5. *Tout circuit de preuve est un réseau booléen.*

Démonstration. Etant donné un circuit de preuve, on raisonne par induction sur la hauteur – telle que posée à la définition 26, page 18 – des pièces qui le compose. Étant donné un ensemble de pièces de hauteur n, on démontre que ces pièces et toutes les pièces de hauteur inférieure strictement à n se contractent en un point.

Si la pièce est de hauteur 0 Cette pièce est unique, et sa sortie et ses poubelles sont connectées au tenseur conclusion. D'autres poubelles sont connectées à ce tenseur, mais elles proviennent de pièces de hauteur supérieure à 0, on peut donc les ignorer ou les considérer comme étant des points reliés à ce tenseur. Après application de $\rightarrow^{ten}$ et $\rightrightarrows^{arr.}$, on se retrouve dans le cas du lemme 2 : la pièce a ses sorties et ses poubelles connectées à un point, et elle se réduit donc en un point.

Si les pièces sont de hauteur $n>0$ Par la définition 26, toutes les pièces auxquelles sont connectées les sorties de ces pièces sont de hauteur strictement inférieure à n. Par hypothèse d'induction, l'ensemble de ces pièces se réduit en un point : toutes les pièces de hauteur n ont donc leurs poubelles et leurs sorties connectées à un point. Par le lemme 2 cet ensemble de pièces se réduit en un point.

Un circuit de preuve est composé exclusivement de pièces et d'un tenseur conclusion, et on vient de démontrer que cet ensemble de liens et d'arêtes se contractait en un point. Par le théorème 4, tout circuit de preuve est donc un réseau de preuve, et il s'agit même d'un réseau booléen, car il est toujours possible de le typer avec

$$\vdash p_1 : \mathbf{B}^{\perp}[A_1], \ldots, p_n : \mathbf{B}^{\perp}[A_n], s : \otimes^{m+1}(\mathbf{B}[A], C_1, \ldots, C_m)$$

□

Remarque 3 (Chemin maximal de contraction). Pour contracter en un point une pièce $DUPL$, il faut nécessairement que toutes les pièces de hauteur inférieure auxquelles cette pièce est connectée aient été contractées. De même, pour contracter une pièce dont la sortie est reliée à l'arête étiquetée e_j d'une pièce $DISJ^i$ ou $CONJ^i$ il faut que l'ensemble des liens correspondant aux entrées et aux poubelles portant un indice supérieur à j aient été contractées en un point.

On peut ainsi définir un *chemin maximal de contraction*, qui dépend à la fois du nombre de pièces du circuit de preuve, de l'arité des fonctions qu'elles représentent, et qui peut être correctement défini car il n'existe pas de cycle dans les dépendances entre les pièces.

On conjecture que le nombre d'étapes nécessaires pour s'assurer qu'un circuit de preuve est bien un réseau de preuve est linéaire en la taille du circuit booléen dont il est le traduit.

Une preuve pourrait passer par une double induction, la première sur la hauteur des duplications et la seconde sur la hauteur des pièces. On pourrait ainsi synchroniser les contractions au moment de considérer les pièces $DUPL$.

6 Mesures de la complexité des circuits de preuve

Par le théorème 5, on sait que tout circuit de preuve peut être typé, et donc qu'il existe une dérivation de profondeur minimale qui peut lui être associé. Il est ainsi possible d'associer une formule à chaque arête d'un circuit de preuve. C'est la mesure de la profondeur de ces formules (comme posée à la définition 12, page 10) qui va nous donner la mesure de la profondeur des circuits de preuves.

Définition 28 (Propagation d'une substitution)**.** On définit la propagation d'une substitution dans une pièce d'un circuit de preuve, de l'une de ses sorties vers son (ses) entrée(s).

Si le typage nous impose qu'une sortie soit du type $\mathbf{B}[A]$ pour A une formule quelconque, on procède ainsi :

- Si la sortie n'est connectée à aucune pièce, alors c'est une entrée du circuit de preuve, il n'y a rien à faire. Le réseau de preuve qu'on y connectera devra être du type $\mathbf{B}[A]$.
- Si la sortie est connectée à b_0 ou à b_1, on effectue la substitution dans les axiomes.
- Si la sortie est connectée à NEG, on substitue dans les axiomes et on propage la substitution à la pièce connectée à l'entrée.
- Si la sortie est connectée à $DUPL^i$, admettons que ce soit la k-ième sortie qui est considérée, pour $1 \leqslant k \leqslant i$. Il suffit alors de substituer dans les axiomes des k-ièmes occurences de b_0 et de b_1, dans l'axiome de la k-ième sortie, et à l'intérieur des trois autres axiomes de cette pièce. Le type de l'entrée était
$$\mathbf{B}^\perp[\otimes^i(\mathbf{B}[A_1],\dots,\mathbf{B}[A_k],\dots,\mathbf{B}[A_i])]$$
il devient
$$\mathbf{B}^\perp[\otimes^i(\mathbf{B}[A_1],\dots,\mathbf{B}[A_k][A],\dots,\mathbf{B}[A_i])]$$
et on propage cette substitution à la pièce connectée à l'entrée.
Dans le cas où $i = 1$, le type de l'entrée passe de $\mathbf{B}^\perp[B[A_1]]$ à $\mathbf{B}^\perp[B[A_1]][A]$.
- Si la pièce est $CONJ^i$ ou $DISJ^i$, on propage la substitution $[A]$ dans les pièces qui mènent à e_1 et $[\mathbf{B}[A]]$ dans les pièces qui mènent à e_j pour $2 \leqslant j \leqslant i$. On effectue également les substitutions *ad hoc* dans les axiomes de la pièce.

Puisqu'un circuit de preuve est acyclique, on est assuré que cette propagation termine toujours.

Définition 29 (Typage d'un circuit de preuve)**.** On assigne à la sortie du circuit de preuve le type $\mathbf{B}$ et on propage la substitution vide à la pièce de hauteur minimale.

Lemme 3. *Soit $\mathscr{P}$ une pièce d'un circuit de preuve à $n \geqslant 1$ entrées et $m \geqslant 1$ sorties. Soient $A_{s_1},\dots,A_{s_m}$ les formules à sa sortie et $A_{e_1},\dots,A_{e_n}$ les formules à son entrée. On a pour $j \in \{1,\dots,n\}$*

$$d(A_{e_j}) \leqslant d(\boldsymbol{B}) + max(d(A_{s_1}),\dots,d(A_{s_m}))$$

Démonstration. Puisque $\mathscr{P}$ est dans un circuit de preuve, on sait qu'il est possible d'associer à chacune de ses arêtes une formule.

On raisonne par cas sur $\mathscr{P}$:

Si $\mathscr{P} \in \{b_0, b_1\}$, $\mathscr{P}$ ne remplit pas les hypothèses du lemme.

Si $\mathscr{P} = NEG$, alors $d(A_{e_1}) = d(A_{s_1})$.

Si $\mathscr{P} = DISJ^i$ ou $CONJ^i$, alors $d(A_{e_1}) = d(A_s)$ et $d(A_{e_j}) = d(A_s) + d(\mathbf{B})$ pour $2 \leqslant j \leqslant i$.

Si $\mathscr{P} = DUPL^i$, alors

$$\begin{aligned} d(A_{e_1}) &= d(\otimes^3[(\parr^n(A^\perp_{s_n},\dots,A^\perp_{s_1}))\parr(\parr^n(A^\perp_{s_n},\dots,A^\perp_{s_1})),\otimes^n(A_{s_1},\dots,A_{s_n}),\otimes^n(A_{s_1},\dots,A_{s_n})]) \\ &= max(d((\parr^n(A^\perp_{s_n},\dots,A^\perp_{s_1}))\parr(\parr^n(A^\perp_{s_n},\dots,A^\perp_{s_1}))),d(\otimes^n(A_{s_1},\dots,A_{s_n}))) + 1 \\ &= max(d(A_{s_1}),\dots,d(A_{s_n})) + 3 \end{aligned}$$

Étant donné que $d(\mathbf{B}) = 3$, on a bien dans tous les cas $d(A_{e_j}) \leqslant d(\mathbf{B}) + max(d(A_{s_1}),\dots,d(A_{s_m}))$.

□

On peut remarquer à cette occasion que la profondeur d'une pièce est indépendante de son arité.

Corollaire 1. *La profondeur maximale d'un circuit de preuve se situe toujours au niveau des coupures connectant les entrées au circuit.*

Démonstration. Il suffit de remarquer que les poubelles n'étant jamais coupées, elles n'entrent pas en compte dans la mesure de la profondeur d'un réseau de preuve. Le lemme précédent fait le reste. On a vu en définissant la propagation d'une substitution qu'au sein d'une même pièce les profondeurs des entrées étaient supérieures ou égales aux profondeurs des sorties. □

Théorème 6. *Soit C_n un circuit booléen, $p(C_n)$ sa profondeur, $\mathscr{T}(C_n)$ son traduit en circuit de preuve. On a $d(\mathscr{T}(C_n)) = \mathscr{O}(p(C_n))$.*

Démonstration. Soit x_i l'entrée de $\mathscr{T}(C_n)$ où la formule de coupure est la plus profonde. Le chemin de x_i à la sortie de $\mathscr{T}(C_n)$ passe par au plus $p(C_n) \times 2$ pièces : ce cas se présente si ce chemin coïncide avec le chemin le plus long de C_n et que chaque résultat est dupliqué. Par le lemme 3, la différence de profondeur logique entre une entrée et la ou les sorties de chaque pièce est d'au plus $d(\mathbf{B})$. On a donc $d(\mathscr{T}(C_n)) \leqslant p(C_n) \times 2 \times d(\mathbf{B})$. □

Théorème 7. *Soient C_n un circuit booléen, $\mathscr{T}(C_n)$ son traduit en circuit de preuve et $i \in \mathbb{N}$, on note $|C_n|$ et $|\mathscr{T}(C_n)|$ leurs tailles respectives. Si C_n appartient à une famille de circuits booléens comprise dans NC^i (resp. AC^i), $|\mathscr{T}(C_n)| = \mathscr{O}(|C_n|)$ (resp. $|\mathscr{T}(C_n)| = \mathscr{O}(|C_n|^2)$).*

Démonstration. On note k le nombre d'arêtes de C_n : si C_n appartient à une famille comprise dans NC^i (resp. AC^i), $k \leqslant 2 \times |C_n|$ (resp. $k \leqslant |C_n|^2$).

On établit par simple décompte le tableau de correspondance suivant :

Nœud du circuit booléen	Nombre de liens dans le circuit de preuve traduit
0 ou 1	4
$\neg$	5
$\wedge^i$ ou $\vee^i$	$10 + ((i-2) \times 9) = 9i - 8$

À chaque fois qu'une valeur est dupliquée i fois, $|\mathscr{T}(C_n)|$ augmente de $(i \times 4 \times 2) + i + 8 = 9i + 8$.

Un nœud de C_n d'arité entrante n et d'arité sortante m est représenté par un réseau de preuve de taille au plus $(9n-8) + (9m+8) = 9(n+m)$.

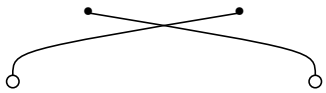
Puisque la somme des arités entrantes et sortantes de l'ensemble des nœuds de C_n ne peut dépasser k, $|\mathscr{T}(C_n)| = \mathscr{O}(k)$. □

7 Des circuits de preuve vers les circuits booléens

Le théorème 3, page 10, dû à [Girard, 1987], nous enseigne que le nombre d'étapes d'élimination des coupures dans un réseau de preuve est linéaire en la taille de ce réseau de preuve. Ce résultat est insatisfaisant du point de vue du calcul parallèle, mais K. Terui parvient à nettement l'améliorer en parallélisant de façon efficace les a-réductions.

Les a-réductions présentent en effet une resistance à la réduction en parallèle des coupures : supposons que nous voulions réduire

Si nous appliquons simultanément les deux a-réductions possibles, nous obtenons



et non, comme nous l'aurions souhaité,

Pour remédier à cela, Terui introduit une règle d'élimination des coupures (la t-réduction) permettant de réduire en une étape les «chaînes» de liens axiomes reliés par des coupures. Cette nouvelle règle lui permet de démontrer que tout réseau de preuve P_n se réduit en une forme normale en haut plus $d(P_n) \times 3$ étapes.

Un simple codage lui permet ensuite de prouver qu'il existe trois circuits booléens de taille $\mathscr{O}(|P_n|^3)$ et de profondeur constante qui peuvent – étant donnée la description d'un réseau de preuve P_n – donner en sortie la description de P_n après une des trois étapes de réductions (a-, m- ou t-réduction).

Seulement le circuit booléen capable de simuler la t-réduction utilise des portes $UstCONN_2$: cette fonction – qui identifie les chemins entre deux sommets dans un graphe non-dirigé de degré 2 – est nécessaire pour identifier les chaînes maximales de liens axiomes coupés. $UstCONN_2$ est suffisante car les liens axiomes sont effectivement des nœuds d'arité deux et qu'il n'est pas nécessaire de considérer les autres liens pour identifier ces chaînes.

7.1 La complexité de $UstCONN_2$

On ne connaît pas de classe de complexité pour laquelle $UstCONN_2$ soit complète, cependant on connaît les conséquences de l'ajout de cette fonction à une base.

Théorème 8 ([Wigderson, 1992])**.** *$UstCONN$ est un problème L-complet.*

Puisque $L \subseteq AC^1$, on a, pour $i \geqslant 0$, $AC^i(UstCONN_2) \subseteq AC^{i+1}$.

7.2 Que nos pièces ne font pas mieux que les réseaux booléens

L'élimination des coupures dans nos circuits de preuves génère des chaînes de liens axiomes coupés tout autant que les réseaux de preuves booléens de Terui. Pour simuler l'élimination des coupures dans un circuit de preuve avec un circuit booléen de façon satisfaisante, il nous est donc également nécessaire de faire appel à des nœuds étiquetés $UstCONN_2$.

Cette obligation ne nous permet donc pas d'affiner le résultat de Terui, qui est :

Théorème 9 ([Terui, 2004])**.** *Pour tout réseau booléen P_n il existe un circuit booléen de taille $\mathcal{O}(|P_n|^4)$ et de profondeur $\mathcal{O}(d(P_n))$ sur la base $\mathfrak{B}_1(UstCONN_2)$ qui accepte le même ensemble que P_n.*

8 Inclusions des classes de complexité

Définition 30 (CCP^i)**.** Pour $i \in \mathbb{N}$, CCP^i est la classe des langages pour lesquels il y a une famille uniforme de circuits de preuves à n entrées de taille polynomiale en n et de profondeur $\mathcal{O}(\log^i(n))$ qui les accepte.

Remarque 4. Pour $i \in \mathbb{N}$, $CCP^i \subseteq mBN^i$ est trivial, étant donné que tout circuit de preuve uniforme est un réseau booléen uniforme.

Définition 31 (Traduction de AC^i vers CCP^i)**.** On pose le problème suivant :

Problème :	Traduction de AC^i vers CCP^i.
Entrée :	Une description d'une famille de circuits booléens appartenant à AC^i.
Sortie :	Une description d'une famille de circuits de preuves appartenant à CCP^i tel que le résultat de leur évaluation *via* élimination des coupures coïncide avec le résultat de l'évaluation du circuit booléen.

Théorème 10. *Traduction de AC^i vers CCP^i appartient à AC^0.*

Démonstration. On construit un circuit booléen de profondeur constante sur $\mathfrak{B}_1$, qui procède comme suit :

- Un premier circuit à profondeur constante associe à chaque nœud une ou deux pièces, selon l'arité sortante du nœud.
- Un second circuit à profondeur constante relie la sortie et les arêtes poubelles créées au tenseur conclusion.

Les théorèmes 6 et 7 nous assurent que les circuits de preuves ainsi obtenus sont de taille $\mathcal{O}(|C_n|^2)$ et de profondeur $\mathcal{O}(p(C_n))$, donc que cette famille appartient à CCP^i. □

On peut démontrer de la même manière que le problème analogue Traduction de NC^i vers CCP^i appartient également à AC^0.

Ce résultat constitue une amélioration du théorème 2 de [Mogbil et Rahli, 2007], qui énonce que pour tout $i \in \mathbb{N}$, la traduction de Terui d'une famille de circuits booléens appartenant à $AC^i(UstCONN_2)$ vers une famille de réseaux booléens de mBN^i est dans L.

Notre traduction des circuits booléens vers les circuits de preuve peut s'effectuer en profondeur constante et préserve l'uniformité.

Corollaire 2. *Pour $i \in \mathbb{N}$, $AC^i \subseteq CCP^i$.*

Puisque tout circuit de preuve est un réseau de preuve, nous pouvons appliquer le théorème 9 à nos circuits de preuve : ainsi, pour $i \in \mathbb{N}$, $CCP^i \subseteq AC^i(UstCONN_2) \subseteq AC^{i+1}$.

Ces résultats améliorent l'apport majeur de Terui, qui est :

Théorème 11 ([Terui, 2004])**.** *$APN^i = AC^i(USTConn_2)$ non-uniforme pour $i \in \mathbb{N}$, et de plus $APN = NC$ non-uniforme.*

Ce résultat ne permettait pas de caractériser les circuits booléens à profondeur constante à l'aide des réseaux de preuves, car $AC^0(UstCONN_2)$ non-uniforme n'est vraissemblablement pas une classe de complexité à profondeur constante.

Notre corolaire 2 permet, lui, de caractériser précisément AC^0 uniforme, car $AC^0 \subseteq CCP^0$. Il est donc désormais possible de borner précisément les classes de complexité à profondeur constante par des classes de complexité nées de la Logique Linéaire.

Conclusion

Apports scientifiques

Notre travail présente à nos yeux deux mérites : celui de simplifier la simulation des réseaux booléens par les réseaux de preuves, et celui d'étendre un résultat à de nouvelles classes de complexité.

En simulant directement les connecteurs booléens par des pièces, nous rendons plus fidèle – et plus économe – la traduction des circuits booléens vers les réseaux de preuves. Cette présentation permet de réduire la taille des réseaux de preuves ainsi obtenus et d'appliquer les résultats de Terui aux petites classes de complexité.

Les résultats de Terui ne prenaient pas en compte – de façon étonnante – que les classes de complexités non uniformes. Nous confirmons les travaux de V. Mogbil en remarquant que ces résultats peuvent également s'appliquer aux classes uniformes, et les affinons en prenant en compte les petites classes de complexité.

Perspectives

Malgré nos efforts, nous n'avons pas trouvé de moyen d'améliorer la simulation de l'élimination des coupures dans un circuit de preuve avec les circuits booléens : cela ne signifie pas pour autant que $UstCONN_2$ soit nécessaire à la simulation efficace de l'évaluation des réseaux de preuves non déterministes.

Les classes de réseaux de preuves avec non-déterminisme explicite n'ont pas été abordées dans le cadre de ce mémoire, mais peut-être existe-t-il des extensions des résultats déjà obtenus aux petites classes de complexité.

Apports personnels du stage

Ce stage m'a tout d'abord permis d'observer le fonctionnement d'un laboratoire et de me familiariser avec la recherche. Par la lecture, le questionnement et la discussion, j'ai appris à améliorer ma méthode de travail et à mieux dialoguer avec mes encadrants. Cette première immersion a confirmé mon goût pour la recherche et mon envie de continuer en thèse mon parcours scolaire.

Ce sujet m'a posé quelques difficultés du côté de la théorie de complexité : mes premières semaines de travail furent donc consacrées à ce sujet, notamment aux circuits booléens et aux problèmes d'uniformité. Cet apprentissage effectué, j'ai pu tenter de relier mes connaissances en Logique Linéaire aux thématiques de la complexité, en commençant par des sujets plus vastes et mieux balisés.

Le thème des classes de complexité sous-logatrithmiques était particulièrement enrichissant, car il s'agissait pour une large partie de simplifier la méthode inaugurée par Terui. Ce n'était donc pas un pur exercice d'invention, mais une solide compréhension des articles et un peu d'inventivité étaient nécessaires.

La rédaction de ce mémoire m'a permis de mieux manipuler LaTeX et notamment le *package* Ti*kZ*, outils indispensables de la communauté mathématique.

Table des figures

Liste des tableaux

Table des matières

Bibliographie

[Allender, 1989] ALLENDER, E. (1989). P-uniform circuit complexity. *Journal of the ACM (JACM)*, 36(4): 912–928.

[Barrington *et al.*, 1990] BARRINGTON, D. A., IMMERMAN, N. et STRAUBING, H. (1990). On uniformity within NC1. *Journal of Computer and System Sciences*, 41(3):274–306.

[Cook, 1985] COOK, S. A. (1985). A taxinomy of problems with fast parallel algorithms. *Information and Control*, 64(1-3):2–22.

[Danos, 1990] DANOS, V. (1990). La Logique Linéaire appliquée à l'étude de divers processus de normalisation (principalement du λ-calcul). *These de doctorat, Université Paris VII*.

[Furst *et al.*, 1984] FURST, M., SAXE, J. et SIPSER, M. (1984). Parity, circuits, and the polynomial-time hierarchy. *Theory of Computing Systems*, 17(1):13–27.

[Girard, 1987] GIRARD, J.-Y. (1987). Linear logic. *Theoretical computer science*, 50(1):1–101.

[Greenlaw *et al.*, 1995] GREENLAW, R., HOOVER, H. J. et RUZZO, W. L. (1995). *Limits to Parallel Computation : P-Completeness theory*. Oxford University Press, USA.

[Guerrini et Masini, 2001] GUERRINI, S. et MASINI, A. (2001). Parsing MELL proof nets. *Theoretical Computer Science*, 254(1):317–335.

[Ladner, 1975] LADNER, R. (1975). The circuit value problem is log space complete for P. *ACM Sigact News*, 7(1):20.

[Lafont, 1995] LAFONT, Y. (1995). From Proof-Nets to Interaction Nets. *Advances in Linear Logic*, 222: 225–247.

[Mairson, 2004] MAIRSON, H. (2004). Linear lambda calculus and PTIME-completeness. *Journal of Functional Programming*, 14(6):623–633.

[Mairson et Terui, 2003] MAIRSON, H. et TERUI, K. (2003). On the computational complexity of cut-elimination in linear logic. *Theoretical Computer Science*, pages 23–36.

[Mogbil, 2009] MOGBIL, V. (2009). Non-deterministic Boolean Proof Nets. *In Proceedings of FOPARA'09*.

[Mogbil et Rahli, 2007] MOGBIL, V. et RAHLI, V. (2007). Uniform circuits, & Boolean proof nets. *In Proceedings of L.F.C.S.*, pages 401–421. Springer.

[Poizat, 1995] POIZAT, B. (1995). *Les petits cailloux*. Aléas.

[Ruzzo, 1981] RUZZO, W. (1981). On uniform circuit complexity. *Journal of Computer and System Sciences*, 22(3):365–383.

[Terui, 2004] TERUI, K. (2004). Proof Nets and Boolean Circuits. *In Proceedings of LICS'04*, pages 182–191.

[Vollmer, 1999] VOLLMER, H. (1999). *Introduction to Circuit Complexity : A Uniform Approach*. Springer Verlag.

[Wigderson, 1992] WIGDERSON, A. (1992). The complexity of graph connectivity. *Mathematical Foundations of Computer Science 1992*, pages 112–132.