Extraction de progamme à partir de preuves de la logique classique

Etude d'un exemple simple

Hugo Herbelin

November 1, 1991

Intro

Nous proposons ici d'étudier a partir d'un exemple l'obtention d'un programme a partir d'une preuve de logique classique.

Nous n'entrerons pas dans des détails concernant la réalisabilite et nous contenterons d'assimiler directement la preuve à un programme, sans se soucier de ce qui est effectivement programme et de ce qui n'a pas de valeur calculatoire.

L'exemple que nous allons étudier est un petit théorème enfantin, prouvable trivialement constructivement mais dont on va étudier une preuve un petit plus compliquée et utilisant un argument classique.

Nous regarderons diverses manières d'extraire un programme de cette preuve et observerons qu'aucune n'est vraiment satisfaisante pour l'exemple choisi. Seul le calcul des séquents symétrique laisse espérer la définition d'un procédé d'extraction convenable.

1 Le théorème binaire : quelques preuves

Ce que dans la suite nous allons appeler théorème binaire est le résultat élémentaire suivant. Ce petit exemple de travail nous a été communiqué par Stolzenberg [17], lui-même intéressé par l'obtention - au moyen de la A-traduction de Friedman - d'une preuve constructive à partir d'une preuve classique.

Théorème *Soit f une suite de $\mathcal{N}$ vers le 2-uplet $\{0,1\}$. Il existe deux indices i et $j > i$ de la suite tels que $f(i) = f(j)$.*

Ce théorème a évidemment une preuve élémentaire :
Preuve constructive : Si les deux premiers termes de la suite f(0) et f(1) sont égaux, c'est prouvé, sinon le troisième terme f(2) est égal à l'un des deux premiers. ∎

Cette preuve est strictement constructive. Elle est aussi clairement symétrique en les rôles joués par 0 et 1, et précisons le, car ce ne sera pas systématiquement le cas des programmes extraits de la preuve classique décrite plus loin, se contente de la connaissance des trois premiers termes de la suite pour conclure.

Quel est le processus sous-jacent à cette preuve ? D'abord tester l'égalité des deux premiers termes de f, fournir les indices 0 et 1 comme prouvant le théorème si $f(0) = f(1)$, sinon "on sait" que $f(0) = f(2)$ ou que $f(1) = f(2)$. Selon le cas fournir les indices 0 et 2 ou les indices 1 et 2. Ce "on sait" n'est pas très explicite. En fait il faut faire un choix pour séquentialiser la détermination de $f(0) = f(2)$ ou $f(1) = f(2)$. Nous verrons cela plus précisément plus loin.

Mais cela débouche sur des problèmes plus complexes : la transcription du raisonnement humain dans un système formel n'est pas immédiat. Les systèmes logiques — du moins les systèmes actuels — ne tolèrent ni incertitude, ni "parallélisme" dans le raisonnement.

D'autre part, nous avons rédigé la preuve ci-dessus telle qu'elle nous venait à l'esprit sans se soucier de son caractère calculatoire. Mais la formulation obtenue pour l'algorithme ne semble pas aussi naturelle que la preuve. Nous reparlerons de cela un peu plus loin.

Une preuve classique

Afin de regarder ce qu'on obtient comme programme à partir de preuves classiques, nous allons nous intéresser à une preuve un peu détournée de ce même résultat élémentaire :

Lemme *classique* : *Soit f une suite de $\mathcal{N}$ vers le 2-uplet $\{0, 1\}$. Alors il y a une infinité de 0 ou il y a une infinité de 1.*

PREUVE DU LEMME CLASSIQUE : Par l'absurde. S'il existe x_0 tel que pour tout $y \geq x_0$, $f(y) = 0$ et s'il existe x_1 tel que pour tout $y \geq x_1$, $f(y) = 1$, alors considérant le maximum m de x_0 et x_1, on a $f(m) = 0$ ou $f(m) = 1$ contredisant l'hypothèse. ■

Notre preuve détournée est alors la suivante :

PREUVE CLASSIQUE : S'il y a infinité de 0 alors il y a au moins deux 0 et donc deux termes égaux entre eux. De même s'il y a une infinité de 1. D'où le résultat. ■

Est-il possible de court-circuiter l'appel à l'infini dans cette preuve et de néanmoins en tirer un programme ?

2 Interprétation des preuves intuitionnistes

2.1 Intuitionnisme et réalisabilité

Il me semble que les premières mises en évidence d'un lien entre une preuve et un procédé de calcul (plus prosaïquement un programme, ou, en se référant à la notion théorique qu'on s'accorde à lui associer, une fonction calculable) remontent a l'intuitionnisme et à ses démêlés avec le courant classique (platonicien) de la logique. En effet, autant que j'en sache, les intuitionnistes posèrent le principe qu'un énoncé existentiel $\exists x.P(x)$ prouvable devait se ramener à l'exhibition d'un x_0 accompagné d'une preuve de la véracité de $P(x_0)$. Rejetant par là les axiomes auxquels on ne peut associer de programme comme le tiers-exclu, axiome que le courant classique de la logique reconnaissait, lui, comme valide.

C'est ensuite dans les années 40, quelques années après avoir dégagé la notion de fonction calculable (que la communauté informatique reconnait comme coïncidant avec les fonctions qu'on peut effectivement programmer, acte de foi - quoique vraisemblable - nommé thèse de Church, qu'aucun contre-exemple n'a jusque là infirmé), que Kleene [11] chercha à expliciter ce lien entre preuve intuitionniste et programme : se basant sur un systeme formel fournissant les axiomes propres à traduire un raisonnement intuitionniste (donc sans le tiers-exclu) et fournissant les entiers, l'addition, la multiplication, des axiomes décrivant leurs propriétés fondamentales, ainsi que le principe d'induction (= l'arithmétique de Heyting), il associa a chaque preuve une fonction, dite réalisation de la preuve, de telle sorte que, par exemple, des énoncés $\forall x \exists y.P(x, y)$ prouvables étaient réalisés par une fonction qui, à chaque entier x, associait un entier y vérifiant $P(x, y)$.

La connexion entre intuitionnisme et calculabilité (= la théorie des fonctions calculables) etait alors explicitée : le résultat de base étant qu'à une preuve intuitionniste, on peut effectivement associer une fonction fournissant un témoin explicite aux quantificateurs existentiels.

La formalisation de Kleene était essentiellement à but théorique : il codait les programmes par des nombres, rendant impratiquable l'utilisation concrète de ces programmes. De nombreuses autres manières d'"extraire" des programmes se firent connaître par la suite.

A la fin des années 60, on décrivit même explicitement un isomorphisme (dit de Curry-Howard) entre le calcul propositionnel intuitionniste et le λ-calcul typé (un sous-ensemble du λ-calcul, lui-même étant un des formalismes définissant les fonctions calculables), cette mise en bijection s'exprimant au mieux par le biais de la déduction naturelle, une formulation très élégante de la logique formelle.

Cette analogie entre calcul propositionnel et λ-calcul typé qui est effectivement beaucoup plus qu'une simple analogie ne se restreignit pas au niveau des propositions. Il est possible d'étendre les types du λ-calcul afin d'y interpréter le calcul des prédicats intuitionniste, l'arithmétique de Heyting et même les logiques d'ordre supérieure. De Bruijn [2] dans Automath, Martin-Löf avec son système prédicatif [14], Coquand et Huet [3, 5] avec le Calcul des Constructions suivirent cette voie, qui donna lieu dans ces trois cas à des implémentations de systèmes de preuves, et pour les deux derniers à l'implémentation d'un mécanisme d'extraction de programme à partir de ces preuves. Le système de Martin-Löf est grosso modo muni de constructeurs de type pour chacun des connecteurs logiques. Le calcul de Coquand, en revanche, se distingue par la sobriété de sa formulation. C'est un simple λ-calcul typé où les types ont été enrichis pour leur permettre de dépendre de termes et où l'introduction de l'imprédicativité permet de définir tout aussi bien les connecteurs logiques (autres que l'implication et la quantification universelle) que les structures de données usuelles (entiers naturels, listes, ...) et non usuelles. Une généralisation de ce genre de λ-calcul typé a été par la suite effectuée par Berardi et Terlow, puis Barendregt [1].

2.2 L'isomorphisme de Curry-Howard

Un puissant outil, quoique anodin en lui-même, pour jeter un pont entre preuve et programme est donc l'isomorphisme de Curry-Howard.

L'isomorphisme dit tout simplement que la preuve d'une implication $A \Rightarrow B$ est une fonction de l'ensemble des preuves de A vers l'ensemble des preuves de B. Plus précisément, c'est la fonction obtenue en abstrayant la dépendance en A dans la preuve conditionnelle de B.

Il dit donc aussi qu'un argument logique comme déduire une preuve de B à partir d'une preuve de $A \Rightarrow B$ et d'une preuve de A, correspond exactement à appliquer la preuve de A à la fonction prouvant $A \Rightarrow B$.

L'expression "appliquer un argument à une fonction" peut porter à confusion, dans le sens où elle peut recouvrir deux étapes élémentaires intervenant lors de cette application. Une première étape consiste à juste indiquer que la fonction reçoit un argument. Une deuxième étape consiste à substituer effectivement l'argument là où il est attendu.

Cela introduit par exemple une distinction, si on se place dans le cadre de l'analyse, entre l'application de 3 à la fonction $x \mapsto x^2$ et le terme 3^2. Cela se justifie notamment dans le cas où l'on s'autorise à manipuler des fonctions en tant qu'elle même sans en spécifier une de particuliére et dont on ne connait donc pas le "corps".

Cela met aussi en évidence la substitution comme opération effective, et donc qui demande du travail.

Pour être plus précis sur cet isomorphisme entre preuve et fonction, nous allons introduire précisément les deux formalismes en question, la déduction naturelle et le λ-calcul typé.

L'isomorphisme peut s'étendre à un degré plus ou moins poussé, la base étant l'analogie entre l'implication et la "flèche" de la fonctionnalité. La démarche ici a été de partir de la déduction naturelle et d'étendre l'isomorphisme aux quantificateurs , à la conjonction et à la disjonction. Le λ-calcul typé que nous présentons souffre conséquemment d'un petit manque de naturel, en particulier pour la disjonction et la quantification existentielle ! Il sera néanmoins bon pour "exécuter" les preuves de la déduction naturelle.

2.2.1 La déduction naturelle

Nous considérons tout d'abord un ensemble de variables munies chacune d'une arité et qui permet de définir l'ensemble des termes comme le plus petit ensemble engendré par la règle suivante :

- si f est une variable d'arité n et $x_1, ..., x_n$ des termes alors $f(x_1, ..., x_n)$ est un terme (en particulier une variable d'arité 0 est un terme)

Nous considérons aussi un ensemble de variables de prédicats, munies elles aussi d'une arité (les variables d'arité 0 étant aussi appelées variables propositionnelles), qui permet de définir l'ensemble des propositions comme le plus petit ensemble engendré par les règles suivantes :

- si P est une variable de prédicat d'arité n et $x_1, ..., x_n$ des termes d'arité 0 alors $P(x_1, ..., x_n)$ est une proposition (en particulier une variable propositionnelle est une proposition). Une telle proposition est dite atomique.

- si A est une proposition, alors $\neg A$ est une proposition

- $\perp$ est une proposition particulière

- si A et B sont des proposition, alors $A \Rightarrow B$, $A \vee B$ et $A \wedge B$ sont des propositions

- si A est une proposition et x une variable d'arité 0, alors $\forall x A$ et $\exists x A$ sont des propositions

Nous nous donnons enfin les règles logiques décrites figure 1, page 6.

Pour chaque connecteur logique, il y a une règle d'introduction et une règle d'élimination, à l'exception de l'absurde qui n'a qu'une règle d'élimination — *ex falso quodlibet* —, propre à la version intuitionniste du système (une autre règle pour l'absurde permettra de définir un système classique).

Remarque : Nous avons ici considéré la négation comme un connecteur logique primitif, ce qui correspond à l'attitude naturelle à ce sujet. Néanmoins, nous aurions pu définir $\neg B$ comme $B \Rightarrow \perp$. Cette expression se comportant en déduction naturelle exactement comme la négation, les deux règles $\neg$-E et $\neg$-I devenant des cas particuliers des règles $\Rightarrow$-E et $\Rightarrow$-I.

La normalisation

Une importante propriété de la logique, propriété que le formalisme de la déduction naturelle permet de montrer très facilement, est que toute preuve

$$\frac{A \qquad B}{A \wedge B}\ \wedge\text{-I} \qquad\qquad \frac{A \wedge B}{A}\ \wedge\text{-E} \qquad\qquad \frac{A \wedge B}{B}\ \wedge\text{-E}$$

$$\frac{A}{A \vee B}\ \vee\text{-I} \qquad\qquad \frac{B}{A \vee B}\ \vee\text{-I} \qquad\qquad \frac{A \vee B \qquad \begin{array}{c}[A]\\ \vdots \\ C\end{array} \qquad \begin{array}{c}[A]\\ \vdots \\ C\end{array}}{C}\ \vee\text{-E}$$

$$\frac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \Rightarrow B}\ \Rightarrow\text{-I} \qquad\qquad \frac{A \qquad A \Rightarrow B}{B}\ \Rightarrow\text{-E}$$

$$\frac{\begin{array}{c}[A]\\ \vdots \\ \bot\end{array}}{\neg A}\ \neg\text{-I} \qquad\qquad \frac{A \qquad \neg A}{\bot}\ \neg\text{-E}$$

$$\frac{[y/x]A}{\forall x A}\ \forall\text{-I} \qquad\qquad \frac{\forall x A}{[t/x]A}\ \forall\text{-E}$$

$$\frac{[t/x]A}{\exists x A}\ \exists\text{-I} \qquad\qquad \frac{\exists x A \qquad \begin{array}{c}[[y/x]A]\\ \vdots \\ B\end{array}}{B}\ \exists\text{-E}$$

$$\frac{\bot}{A}\ \bot_I$$

La règle $\forall$-I a pour restriction que la variable y ne doit pas apparaître libre dans $\forall x A$, ni dans aucune hypothèse dont dépend $[y/x]A$.

La règle $\exists$-E a pour restriction que la variable y ne doit pas apparaître libre dans $\exists x A$, ni dans B, ni dans aucune hypothèse dont dépend l'occurrence supérieure de B.

Figure 1 : Règles de la déduction naturelle intuitionniste

d'une proposition utilisant éventuellement des structures arbitrairement complexes peut se ramener à une preuve n'utilisant pas de structures plus complexes que celles intervenant dans l'énoncé de la proposition.

Exprimé dans le cadre de la déduction naturelle, cette propriété s'énonce ainsi : toute preuve peut se ramener à une preuve constituée d'une suite de règles d'élimination, suivie d'une suite de règles d'introduction.

Cette propriété est, dans le cadre de la déduction naturelle, une conséquence du puissant théoréme de normalisation : toute preuve peut se ramener à une preuve sans utilisation consécutive de la règle d'introduction et de la règle d'élimination d'un même connecteur logique.

Par exemple :

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \Rightarrow B} \qquad \begin{array}{c}\vdots\\ A\end{array}}{B}$$

peut se simplifier en :

$$\begin{array}{c}\vdots\\ A\\ \vdots\\ B\end{array}$$

2.2.2 Le λ-calcul typé

Nous gardons la définition des termes de la déduction naturelle. Pour les distinguer des λ-termes ci-dessous définis, nous les appelerons aussi termes d'individu.

Nous considérons aussi un ensemble de variable de construction de types munies d'une arité. Et nous définissons l'ensemble des types du calcul comme le plus petit ensemble clos par les règles suivantes :

- si C est une variable de construction de type d'arité n et $x_1, ..., x_n$ des termes alors $C(x_1, ..., x_n)$ est un type
- **Exc** est un type spécial
- si T et U sont des types, alors $T \to U$, $T + U$ et $T * U$ sont des types
- si T est un type et x une variable, alors $\Pi x T$ et $\Sigma x T$ sont des types

Nous considérons ensuite un ensemble de variables, chacune munie d'un type, dites λ-variables typées.

Et nous considérons l'ensemble des λ-termes typés, comme le plus petit ensemble contenant les λ-variables typées et vérifiant les propriétés suivantes :

- si a est un λ-terme de type T et v une λ-variable de type U, alors $\lambda v.a$ est un λ-terme de type $U \rightarrow T$.
- si a est un λ-terme de type $U \rightarrow T$ et b un λ-terme de type U, alors $(a\ b)$ est un λ-terme de type T.
- si a est un λ-terme de type T et b un λ-terme de type U, alors $< a, b >$ est un λ-terme de type $T * U$
- si a est un λ-terme de type $T * U$, alors $\pi_1(a)$ est un λ-terme de type T
- si a est un λ-terme de type $T * U$, alors $\pi_2(a)$ est un λ-terme de type U
- si a est un λ-terme de type T et U est un type, alors $\iota_1^U(a)$ est un λ-terme de type $T + U$
- si a est un λ-terme de type T et U est un type, alors $\iota_2^U(a)$ est un λ-terme de type $U + T$
- si c est un λ-terme de type $T + U$, a un λ-terme de type $T \rightarrow V$ et b un λ-terme de type $U \rightarrow V$, alors $o(a, b, c)$ est un λ-terme de type V
- si a est un λ-terme de type T et x une variable d'arité 0, telle qu'aucun des types de λ-variables intervenant dans a n'en dépendent, alors $\lambda x.a$ est un λ-terme de type $\Pi x T$.
- si a est un λ-terme de type $\Pi x T$ et t un terme d'individu, alors $(a\ t)$ est un λ-terme de type $T[x := t]$.
- si x est une variable d'arité 0, t un terme d'individu et a un λ-terme de type $T[x := t]$, alors $< t, a >$ est un λ-terme de type $\Sigma x T$.
- si a est un λ-terme de type $\Sigma x T$ et b un λ-terme de type $T \rightarrow U$, tel que ni U, ni aucun des types des λ-variables intervenant dans b ne dépendent de x, alors $\epsilon(a, b)$ est un λ-terme de type U
- si a est un λ-terme de type **Exc** et T un type, alors *fail(a)* est un λ-terme de type T.

La plupart de ces constructions ont un sens calculatoire certain. Le $\lambda v.a$ et le $(a\ b)$ conviennent pour la "fonctionalisation" et pour l'application de l'argument a à la fonction b. Le $< a, b >$ et les π_1, π_2 conviennent pour la formation de la paire et les projections. $\lambda x.a$ où x est une variable d'individu et $(a\ t)$ où t est un terme d'individu, quoique inhabituels, peuvent être compris comme une famille de λ-termes et comme la spécialisation d'un des λ-termes de la famille. *fail(a)* convient pour représenter un "failure".

Les constructions ι_1, ι_2 sont plutôt à comprendre en oubliant le λ-terme qu'elles véhiculent, c'est-à-dire comme des valeurs booléennes. $o(a, b, c)$ apparaît alors comme le "if a then b else c". $< t, a >$ et $\epsilon(a, b)$ sont difficiles à interpréter tels quels dans un cadre informatique.

Normalisation

Dans un tel langage, certains λ-termes sont "simplifiables" (ou réductibles). Par exemple, le terme $(\lambda x.x\ y)$ exprimant l'application de y à la fonction identité peut se réduire en y, $\pi_1(< x, y >)$ peut se réduire en x, etc... A chaque “constructeur” de λ-terme est associé un “destructeur” qui décompose le constructeur.

En se restreignant à la partie la plus signifiante, calculatoirement parlant, de ce λ-calcul, réduire un λ-terme revient à exécuter le programme qu'il représente.

Que le programme s'arrête peut être montré. Effectivement, on a l'important théorème suivant : tout λ-terme, au bout d'une suite finie de simplifications, aboutit à un λ-terme, dit en forme normale, qui n'est plus réductible. C'est le théorème de normalisation. De plus (c'est la propriété de Church-Rosser), peu importe l'ordre dans lequel on réduit (au cas où plusieurs réductions sont simultanément possibles), on aboutira toujours au même λ-terme irréductible.

2.2.3 L'isomorphisme de Curry-Howard

Avec ces formulations, l'isomorphisme de Curry-Howard s'exprime très bien. Les termes d'individus des deux systèmes se correspondent. Les propositions de la déduction naturelle correspondent avec les types du λ-calcul, les règles logiques de la déduction naturelle correspondent avec les règles de formation des λ-termes.

Vu la formulation que nous avons choisi pour notre λ-calcul, les correspondances entre connecteurs logiques et les règles de constructions des types et entre les règles logiques et les règles de formation des λ-termes, sont évidentes. Remarquons juste que $\neg B$ (redondant de $B \Rightarrow \perp$) correspond avec le type $B \to \mathbf{Exc}$ du λ-calcul.

Enfin la normalisation en déduction naturelle correspond exactement à la normalisation en λ-calcul.

Notons qu'en dépit de l'isomorphisme, les choses “signifiantes” en logique et en calculabilité ne sont pas les mêmes. D'un côté, ce sont les propositions qui ont du sens. Importe la prouvabilité, mais la distinction entre deux preuves d'un même théorème n'importe pas (au point qu'avant de s'être mis à comparer la logique et la calculabilité, la question de savoir si la normalisation vérifiait ou pas la propriété de Church-Rosser ne s'était pas posée). De l'autre côté, ce sont les λ-termes en eux mêmes qui on du sens. Par exemple, si on se donne un type primitif $\mathcal{N}$ pour les entiers avec des λ-variables 0 de type $\mathcal{N}$ et S de type $\mathcal{N} \to \mathcal{N}$ permettant d'interpréter les entiers, c'est les entiers eux-mêmes qui sont intéressants et non leur type.

Ce n'est que récemment que les procédures de normalisation en logique (originairement introduites pour faciliter les preuves de cohérences) sont activement étudiées sous leur aspect calculatoire.

2.3 Exemple : interprétation de la preuve constructive du théorème binaire

Nous allons maintenant essayer de formaliser dans **DNI** la preuve constructive du théorème binaire. Pour cela, nous allons particulariser certains termes du système de telle sorte qu'il puisse exprimer de l'arithmétique. Nous isolerons ainsi une variable d'arité 0, notée **0** et une variable d'arité 1 notée **S** (et qui auront le sens de constantes, représentant intuitivement 0 et la fonction successeur, ce qui permet de représenter les entiers naturels).

Dans la suite, tous les termes du calcul seront interprétés comme des entiers naturels ($\mathbf{0}$,$\mathbf{S(0)}$,... étant interprétés comme les entiers 0,1,...).

Nous isolerons aussi un prédicat binaire d'égalité, noté de manière infixe $=$, que l'on interprètera comme le prédicat d'égalité de 2 entiers. Nous ajouterons aussi un axiome exprimant la décidabilité de l'égalité.

Si nous poursuivions l'isomorphisme de Curry-Howard pour interpréter cet axiome, $=$ étant considéré comme un constructeur de type, la décidabilité de l'égalité de x et y apparait comme un opérateur que nous nommons $Dec(x, y)$ ayant le type $x = y + (x = y \rightarrow \mathbf{Exc})$ c'est-à-dire comparable à un booléen, qui décide, vrai ou faux, (gauche du + ou droite du +) si x et y sont effectivement égaux.

Ci-suit la formulation que nous prenons pour cet axiome :

$$\frac{}{\forall x \forall y. x = y \vee \neg x = y}\,\text{Eg?}$$

Les hypothèses du théoreme se réduisent à l'axiome suivant, exprimant qu'on se donne une fonction f de $\mathcal{N}$ dans $\{0, 1\}$:

$$\frac{}{\forall n. f(n) = 0 \vee f(n) = 1}\,\text{Ax}$$

Cet axiome a évidemment un contenu calculatoire. Il est comparable lui aussi à un booléen.

Par souci de simplicité, nous formulerons de façon légérement différente l'énoncé du théorème : $\exists i \exists j > i \wedge f(i) = f(j) = 0 \vee f(i) = f(j) = 1$ et on abrègera $j > i \wedge f(i) = f(j) = 0 \vee f(i) = f(j) = 1$ par $C(i, j)$.

Nous ne serons pas toujours très méticuleux dans l'écriture des règles, par exemple on contractera en une seule introduction les introductions de $\exists i$ et $\exists j$. On passera aussi sous silence les preuves de $j > i$ qui sont ostensiblement non calculatoires.

Une première partie de la preuve consiste à montrer $(f(0) = 0 \wedge f(1) = 1) \vee (f(0) = 1 \wedge f(1) = 0)$ (que l'on abrègera en $D_{01} \vee D_{10}$) à partir de $\neg f(0) = f(1)$.

Soit p_0 la preuve suivante :

$$\dfrac{\dfrac{\dfrac{}{\forall n. f(n) = 0 \vee f(n) = 1}\,\text{Ax}}{f(1) = 0 \vee f(1) = 1} \qquad \dfrac{\dfrac{\dfrac{f(0) = 0 \qquad [f(1) = 0]}{f(0) = f(1)} \qquad \neg f(0) = f(1)}{\bot}}{f(0) = 0 \wedge f(1) = 1} \qquad \dfrac{\dfrac{f(0) = 0 \qquad [f(1) = 1]}{f(0) = 0 \wedge f(1) = 1}}{f(0) = 0 \wedge f(1) = 1}}{f(0) = 0 \wedge f(1) = 1}$$

et p_1 la preuve correspondante en changeant la valeur de $f(0)$.

Soit p la preuve suivante :

$$\dfrac{\dfrac{\dfrac{}{\forall n. f(n) = 0 \lor f(n) = 1}\,\mathrm{Ax}}{f(0) = 0 \lor f(0) = 1} \qquad \dfrac{\begin{array}{c}\neg f(0) = f(1)\ [f(0) = 0]\\ \vdots\, p_0\\ D_{01}\end{array}}{D_{01} \lor D_{10}} \qquad \dfrac{\begin{array}{c}\neg f(0) = f(1)\ [f(0) = 1]\\ \vdots\, p_1\\ D_{10}\end{array}}{D_{01} \lor D_{10}}}{D_{01} \lor D_{10}}$$

Soit alors q_0 le bout de preuve correspondant au cas $f(0) = 0 \land f(1) = 1$:

$$\dfrac{\dfrac{\dfrac{}{\forall n. f(n) = 0 \lor f(n) = 1}\,\mathrm{Ax}}{f(2) = 0 \lor f(2) = 1} \qquad \dfrac{\dfrac{\dfrac{\dfrac{f(0) = 0 \land f(1) = 1}{f(0) = 0} \quad [f(2) = 0]}{f(0) = f(2) = 0}}{\exists i \exists j C(i,j)}}{} \qquad \dfrac{\dfrac{\dfrac{f(0) = 0 \land f(1) = 1}{f(1) = 1} \quad [f(2) = 1]}{f(1) = f(2) = 1}}{\exists i \exists j C(i,j)}}{\exists i \exists j C(i,j)}$$

et q_1 le bout de preuve correspondant au cas $f(0) = 1 \land f(1) = 0$.

p_1 le bout de preuve suivant :

On définit enfin le morceau de preuve q suivant :

$$\dfrac{D_{01} \lor D_{10} \qquad \begin{array}{c}[f(0) = 0 \land f(1) = 1]\\ \vdots\, q_0\\ \exists i \exists j C(i,j)\end{array} \qquad \begin{array}{c}[f(0) = 1 \land f(1) = 0]\\ \vdots\, q_1\\ \exists i \exists j C(i,j)\end{array}}{\exists i \exists j C(i,j)}$$

alors la preuve complète est :

$$\dfrac{\dfrac{\dfrac{}{\forall x \forall y. x = y \lor \neg x = y}\,\mathrm{Eg?}}{f(0) = f(1) \lor \neg f(0) = f(1)} \qquad \dfrac{[f(0) = f(1)]}{\exists i \exists j C(i,j)} \qquad \begin{array}{c}[\neg f(0) = f(1)]\\ \vdots\, p\\ D_{01} \lor D_{10}\\ \vdots\, q\\ \exists i \exists j C(i,j)\end{array}}{\exists i \exists j C(i,j)}$$

Pour interpréter la preuve comme un programme, nous jeterons les λ-termes/preuves non pertinents, les laissant apparaître éventuellement sous le nom e, en particulier pour respecter la syntaxe $< t, a >$ (du type $\Sigma x T$).

En traduisant dans le λ-calcul, via Curry-Howard, on obtient donc ceci :

on traduira les axiomes $\mathrm{Eg?}$ et Ax par des programmes de même noms.

p se traduit alors en :

$$o(Ax(0), o(Ax(1), \iota_1(e), fail(e)), o(Ax(1), fail(e), \iota_2(e))))$$

où les λ-termes non pertinents ont été jetés.

q_0 se traduit en

$$o(Ax(2), < 0, < 2, e >>, < 1, < 2, e >>)$$

q_1 se traduit en

$$o(Ax(2), < 1, < 2, e >>, < 0, < 2, e >>)$$

q se traduit ensuite en :

$$o(p, q_0, q_1)$$

Et le programme se présente alors comme suit :

$$o(Eg?(0, 1), < 0, < 1, e >>, q)$$

On peut aussi représenter le programme par un organigramme, c'est peut-être plus lisible :

On obtient un programme qui n'est ni "naturel", ni optimal (à moins d'être majoritairement dans le cas $f(0) = f(1)$) au sens de la programmation

3 Interprétation des preuves classiques

La logique classique est longtemps restée à côté des questions de calculabilité, considérée, comme une logique non constructive (en particulier les formules existentielles prouvables ne vérifient plus la propriété de fournir un témoin effectif).

Très tôt, en 1925, Kolmogorov [12] décrit une traduction de la logique classique vers la logique intuitionniste, et en déduit que les formules décidables classiquement prouvables sont intuitionnistiquement prouvable. Gödel, Gentzen, Bernays et d'autres définirent aussi de telles traductions. Mais le premier travail d'interprétation de preuves de la logique classique en termes de fonctions calculables remontent sans doute à Kreisel. S'appuyant sur le théorème de Herbrand, il fait correspondre aux preuves, non plus des fonctions calculables simples mais des fonctionnelles “calculables”, qui affirment, par exemple pour un énoncé $\exists y \forall x . P(x, y)$ prouvable, que toute fonction f qui prétende trouver, quelque soit y, un x rendant faux $P(x, y)$, est mise en échec sur un certain y pour lequel $P(f(y), y)$ est vrai; ici, il faut voir la fonctionnelle dans la fonction qui à tout f associe un y tel que $P(f(y), y)$ est vrai. C'est l'interprétation par absence de contre-exemple (no counterexample interpretation).

Cette interprétation fut le premier moyen, sauf erreur, quoique le résultat soit facilement déductible de la première version non publiée de la preuve de Gentzen [8] de la cohérence de l'arithmétique, de montrer que les formules $\forall x \exists y . P(x, y)$ avec $P(x, y)$ décidable (= dont on a une preuve de $P(x, y) \vee \neg P(x, y)$ sans tiers-exclu) classiquement prouvables l'étaient aussi intuitionnistiquement. Plus tard, en 1978, Friedman [7] en donna une preuve purement syntaxique et par là même facilement adaptable à d'autres systèmes. Le côté purement syntaxique débouchait sur une autre perspective, celle d'obtenir de façon mécanique une preuve intuitionniste à partir de la preuve classique d'une formule $\forall x \exists y . P(x, y)$ avec $P(x, y)$ décidable, cas de la spécification d'un programme, et donc d'obtenir un programme d'une preuve classique. Ce sera le principal sujet de cette section.

Tout récemment, de nouveaux travaux ont vu le jour sur l'interprétation calculatoire de la logique classique. Un premier courant est venu du côté de l'informatique avec l'introduction par Felleisen [6] d'un opérateur de controle C dans les langages fonctionnels permettant de simuler des caractéristiques de langages impératifs. Le confrontant au λ-calcul typé, Griffin [10] a reconnu qu'il était typable par $\neg\neg B \Rightarrow B$, et en cela comparable à l'axiome de l'élimination de la double négation. Toutefois, le comportement de cet opérateur est loin d'être clair, en particulier il nécessite une discipline d'exécution bien définie pour ne pas aboutir à des résulats incohérents. Murthy [15], dans sa thèse, a montré les connexions existant entre le choix d'une discipline d'exécution (appel par nom, appel par valeur,...) dans un programme avec l'opérateur de contrôle C obtenu d'une preuve classique et la A-traduction (avec ses variantes).

Un autre courant est venu du côté de la logique avec les nouveaux calculs **LC** de Girard et **FD** de Parigot et avec l'interprétation de la prouvabilité en termes de stratégie gagnante de Coquand [4]. Ces trois derniers travaux s'appuient sur les formalismes logiques de Gentzen, et nous en reparlerons dans une section ultérieure.

3.1 La déduction naturelle classique

Nous noterons **DNC** le système **DNI** où la règle $\perp_I$ est remplacée par la règle suivante :

$$\cfrac{\begin{matrix}[\neg A] \\ \vdots \\ \perp\end{matrix}}{A}\,\perp_C$$

Cela nous donne un calcul des prédicats classique.

3.2 La A-traduction

La A-traduction a été introduite par Friedman pour prouver la l'équivalence entre prouvabilité intuitionniste et classique dans le cas des formules $\forall x \exists y.P(x,y)$ avec $P(x,y)$ décidable.

Plus spécialement, il l'utilise pour prouver la clôture de nombreux systèmes intuitionnistes (dont l'arithmt́ique de Heyting) selon la règle de Markov (i.e. que la prouvabilité de $\neg\neg\exists y.P(x,y)$ avec $P(x,y)$ décidable entraîne celle de $\exists y.P(x,y)$).

Faisant précéder la A-traduction par la traduction de Gödel (i.e. une traduction de la logique classique dans la logique intuitionniste : cf ci-dessous), ce qui donne une preuve intuitionniste de $\forall x \neg\neg\exists y.P(x,y)$, il obtient le résultat annoncé.

En fait, les deux étapes de sa démonstration peuvent être ramenée en une seule, comme cela est fait par exemple par Leivant [13], et ce que nous désignerons maintenant par A-traduction sera la combinaison d'une traduction de la logique classique vers la logique intuitionniste et de la A-traduction de Friedman à proprement parler.

3.2.1 Les traductions de la logique classique dans la logique intuitionniste

La première traduction transformant une formule prouvable classiquement en une formule équivalente prouvable intuitionnistiquement remonte à Kolmogorov [12] en 1925. Mais les travaux de Kolmogorov n'ayant pas été connus immédiatement en occident, c'est surtout la traduction de Gödel de 1932 qui est restée dans l'histoire. La même année, Gentzen introduisit une autre traduction, très proche de celle de Gödel.

L'idée de la traduction est simple, c'est de placer une double négation en des endroits bien choisis afin d' “absorber” l'utilisation des axiomes classiques. La traduction de Kolmogorov plaçait une double négation devant chaque sous-formule de la formule à traduire. La traduction de Gödel, ou plutôt celle de Gentzen qui en différait par le traitement de l'implication, se contentait de placer une double négation autour de chaque $\vee$ et chaque $\exists$.

La traduction que nous allons décrire ici est, la traduction de Gentzen (excepté qu'il définit $\neg\exists x B$ comme $\forall x \neg B$ et $\neg(B \vee C)$ comme $\neg A \wedge \neg B$). Nous nous placerons pour cela dans la déduction naturelle classique.

Définition : Soit B une proposition de la déduction naturelle. B^* se définit inductivement par les règles suivantes :

- si B est atomique alors B^* est $\neg\neg B$
- $(B \vee C)^*$ est $\neg\neg(B^* \vee C^*)$
- $(B \wedge C)^*$ est $B^* \wedge C^*$
- $(B \Rightarrow C)^*$ est $B^* \Rightarrow C^*$
- $(\neg B)^*$ est $\neg B^*$
- $(\perp)^*$ est $\perp$

- $(\exists x B)^*$ est $\neg\neg\exists x B^*$
- $(\forall x B)^*$ est $\forall x B^*$

Lemme *Dans **DNI**, on a* $\neg\neg B^* \Rightarrow B^*$

PREUVE Par induction sur la structure de B. ■

Théorème *Si B est prouvable dans **DNC**, alors B^* est prouvable dans **DNI**.*

PREUVE On prouve en fait quelque chose de plus fort : si B est prouvable dans **DNC** sous les hypothèses A_i, alors B^* est prouvable dans **DNI** sous les hypothèses A_i^*.

Cela se montre par induction sur la structure de la preuve de B. Le lemme prend en compte le cas où la dernière règle appliquée est la règle $\perp$-C. Les cas les plus compliqués, ceux où la dernière règle appliquée est $\vee$-E ou $\exists$-E, nécessitent aussi l'application du lemme. ■

Remarque : La traduction * est une traduction dans le fragment minimal de **DNC**. C'est-à-dire que la règle $\perp$-I n'est pas utilisée dans la preuve de B^*.

3.2.2 Conservation de la prouvabilité intuitionniste des formules de la forme $\forall x \exists y.P(x,y)$ dans l'extension à la logique classique

L'idée maintenant est de remplacer la négation $\neg B$ dans la traduction précédente par une "A-négation" $A \Rightarrow \perp$. Comme la traduction précédente est en fait une traduction dans le fragment minimal de **DNI** (i.e. sans la règle $\perp$-I), on a similairement :

Définition : Soit A une proposition quelconque.
La A-traduction B^A de B est définie comme la substitution systématique de $\perp$ par A et de $\neg B$ par $B \Rightarrow A$ dans B^*.

Théorème *Si B est prouvable dans **DNC**, alors B^A est prouvable dans **DNI**.*

On peut ainsi en déduire la conservation de la prouvabilité intuitionniste des formules $\forall x \exists y.P(x,y)$ dans l'extension à la logique classique :

Théorème *Si $\forall x \exists y.P(x,y)$, avec P atomique, est prouvable dans **DNC**, alors il est prouvable dans **DNI**.*

PREUVE Par le théorème précédent on obtient la prouvabilité de

$$\forall x((\exists y.((P(x,y) \Rightarrow A) \Rightarrow A) \Rightarrow A) \Rightarrow A)$$

i.e. de

$$\forall x(((\exists y.P(x,y)) \Rightarrow A) \Rightarrow A)$$

et en prenant A identique à $\exists y.P(x,y)$, on obtient

$$\forall x(((\exists y.P(x,y)) \Rightarrow \exists y.P(x,y)) \Rightarrow \exists y.P(x,y))$$

i.e

$$\forall x \exists y.P(x,y)$$

■

On peut même prouver un petit peu plus, ce que nous allons faire en vu d'appliquer ce théorème au théorème binaire :

Théorème *Si $\forall x \exists y.P(x,y)$, avec P formule sans quantificateurs, ni implication, est prouvable dans* ***DNC****, alors il est prouvable dans* ***DNI****.*

PREUVE Cela vient du fait que pour une formule P sans quantificateur universel (et a fortiori sans quantificateurs du tout), ni implication, on a

$$P^A \Rightarrow ((P \Rightarrow A) \Rightarrow A)$$

. ■

3.2.3 Une expression de la preuve classique du théorème binaire

On cherche à transcrire la preuve classique du théorème binaire donnée dans la première section de ce papier. Il est possible de transcrire le lemme classique tel qu'il est rédigé, en supposant

$$\exists x_0 \forall y_0 \neg B(x_0, y_0) \wedge \exists x_1 \forall y_1 \neg B(x_1, y_1)$$

pour aboutir ensuite à une contradiction, mais la preuve formelle de l'équivalence entre

$$\exists x_0 \forall y_0 \neg B(x_0, y_0) \wedge \exists x_1 \forall y_1 \neg B(x_1, y_1)$$

et

$$\neg(\forall \exists y_0 f(y_0) = 0 \vee \forall \exists y_1 f(y_1) = 1)$$

n'est pas aussi succincte que son pendant intuitif, ce qui nous amène à considérer une preuve plus courte (et en plus en forme normale, si l'on considère que la règle $\bot$-C ne peut se réduire d'aucune façon).

Nous considérerons comme implicite les preuves et axiomes relatives à $y_0 \geq x_0$ et à $y_1 \geq x_1$. Nous noterons m pour le maximum de x_0 et x_1, et $B_k(x,y)$ pour $y \geq x \wedge f(y) = k$.

Soit p le bout de preuve suivant :

$$\dfrac{\dfrac{\forall n.f(n)=0 \vee f(n)=1}{f(m)=0 \vee f(m)=1} \qquad \dfrac{\dfrac{f(m)=0}{\exists y_0 f(y_0)=0} \qquad \neg\exists y_0 f(y_0)=0}{\bot} \qquad \dfrac{\dfrac{f(m)=1}{\exists y_1 f(y_1)=1} \qquad \neg\exists y_1 f(y_1)=1}{\bot}}{\bot}$$

la preuve complète du lemme classique est la suivante :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}[\neg\exists y_0 f(y_0)=0]\ [\neg\exists y_1 f(y_1)=1]\\ \vdots\, p\\ \exists y_0 f(y_0)=0\end{array}}{\forall x_0\exists y_0 f(y_0)=0}}{\forall x_0\exists y_0 f(y_0)=0 \lor \forall x_1\exists y_1 f(y_1)=1} \qquad \neg(\forall x_0\exists y_0 f(y_0)=0 \lor \forall x_1\exists y_1 f(y_1)=1)}{\bot}}{\exists y_1 f(y_1)=1}}{\forall x_1\exists y_1 f(y_1)=1}}{\forall\exists y_0 f(y_0)=0 \lor \forall\exists y_1 f(y_1)=1} \qquad \neg\forall\exists y_0 f(y_0)=0 \lor \forall\exists y_1 f(y_1)=1}{\bot}}{\forall\exists y_0 f(y_0)=0 \lor \forall\exists y_1 f(y_1)=1}$$

Le reste de la preuve est le suivant, où l'on notera $C(i,j)$ pour $j > i \wedge (f(i) = f(j) = 0 \vee f(i) = f(j) = 1)$:

3.2.4 Application de la A-traduction à la preuve classique du théorème binaire

Les théorèmes énoncés ci-dessus, quoique les détails n'ait pas été donnés, décrivent de manière non ambiguë une transformation des preuves de **DNC** en preuves de **DNI**.

En particulier, en appliquant le dernier théorème énoncé à la preuve classique de $\exists i \exists j C(i,j)$, on obtient de manière non ambiguë une preuve non classique de la même formule. L'équivalence entre $C(i,j)^*$ et $\neg\neg C(i,j)$ nous permettra de traduire $C(i,j)$ plus simplement. On notera e les λ-termes non calculatoirement pertinents du programme, mais dont on a besoin pour respecter la syntaxe (en particulier celle des λ-termes de type $\Sigma x T$). Et interprétant cette preuve comme un programme via Curry-Howard, on obtient le λ-terme suivant :

Posons d'abord

$p \equiv \lambda F.(F\ \iota_1(\lambda x_0.\lambda H_0.(F\ \iota_2(\lambda x_1.\lambda H_1.o(Ax(m),(H_0\ m),(H_1\ m))))))$

ce qui correspond à la traduction du lemme classique. F est ici de type $((\Pi x_0 \Sigma y_0 B(x_0,y_0)) + (\Pi x_1 \Sigma y_1 B(x_1,y_1))) \to \Sigma i \Sigma j C(i,j)$. C'est donc une fonctionnelle qui prend comme arguments deux fonctions (du type $\lambda x. < y, e >$) et renvoie i et j.

Le programme complet est alors le suivant :

$(p\ \lambda D.(I\ o(p, \lambda g. < g(0), < g(g(0)+1), e >>, \lambda g. < g(0), < g(g(0)+1), e >>)$

où I est l'identité $\lambda a.a$ correspondant à la preuve de $A \Rightarrow A$ que l'on applique à $(A \Rightarrow A) \Rightarrow A$ dans le théorème.

En normalisant le terme, on obtient le programme suivant dont le résultat ne dépend que des valeurs de f, ce que représente l'axiome Ax :

On peut alors faire les remarques suivantes :

1. on obtient un algorithme qui nécessite dans le pire cas la connaissance de 4 termes de la suite f pour pouvoir conclure
2. l'algorithme n'est pas symétrique en échangeant les 0 et les 1
3. en revanche, la preuve telle qu'on l'a écrite informellement ne présente aucune dissymétrie entre les rôles joués par 0 et par 1

3.3 Continuations

Si on étend l'isomorphisme de Curry-Howard pour interpréter $\perp$-C par l'opérateur C de Felleisen (et en identifiant $\perp$ avec $\exists i \exists C(i,j)$ pour respecter le typage), d'après un rapide survol du programme obtenu, on peut dire les choses suivantes :

1. contrairement au cas général, l'ordre d'évaluation du programme n'influe pas sur le résultat du programme
2. le résultat du programme est le même que celui obtenu via la A-traduction

Il semble que dans l'exemple précis du théorème binaire, la manière d'évaluer la règle $\perp$-C, contrairement au cas général, ne modifie pas le résultat du programme. Il semble que l'ordre, séquentialisé, d'élimination des $\exists x_0 B(x_0, y_0)$ et

$\exists x_1 B(x_1, y_1)$ soit le seul paramètre qui détermine l'asymétrie du résultat du programme, et dans quel sens est cette asymétrie.

3.4 No counterexample interpretation

Il semble que le résultat donné par cette interprétation est liée à la façon dont sont traitées les coupures dans la preuve du théorème de Herbrand.

Autrement dit, si on considère le théorème d'Herbrand comme conséquence du Midsequent theorem, le résultat donné par l'interprétation par absence de contre-exemple dépendront de la manière d'éliminer les coupures.

Une autre question est celle-ci : au lieu de traduire la preuve sans coupure par le NCI, peut-on traduire indépendamment les différentes parties de la preuve par le NCI, puis ensuite éliminer les coupures.

4 Le calcul des séquents symétrique : LK

Nous allons voir maintenant un autre formalisme (**LK**) de la logique classique qui ne consiste pas en un simple ajout d'axiomes classiques à la logique intuitionniste. Contrairement à ce qui se passe en déduction naturelle classique, la transcription de l'argument classique dans **LK** préserve la symétrie en 0 et 1. Plus exactement, il y a rupture de symétrie du fait de choix à faire dans l'ordre d'utilisation des règles logiques, mais ces choix ne deviendront essentiels que lorsqu'on tachera de traduire vers la logique intuitionniste.

Nous parlerons en particulier de la traduction dans le calcul des séquents symétrique, mais constructif, de Girard [9], dans le système de la déduction libre de Parigot [16], ainsi que de l'interprétation de la prouvabilité en terme de stratégie gagnante par Coquand [4].

Le problème de base pour interpréter **LK** comme langage de programmation est le non déterminisme de sa règle d' "exécution", l'élimination des coupures. Girard a alors propose une restriction (**LC**) du calcul de Gentzen pour qu'on puisse lui associer une procédure de normalisation non ambiguë. Sa restriction consiste essentiellement à associer des polarités aux formules, polarités qui forceront laquelle des 2 formules impliquées dans une coupure multiple sera dedoublée. De plus ces polarités ne dépendront que des connecteurs intervenant dans la formules et de la polarité des formules atomiques. En particulier, et c'est un point fondamental, une formule quantifiée universellement sera négative et une formule quantifiée existentiellement sera positive. Il faut rapprocher les termes négatif et positif, respectivement de récepteur et émetteur.

Coquand, quant à lui, reste dans **LK**, mais suppose les formules atomiques décidables. Il associe aux formules prouvables une stratégie gagnante dans un jeu où le contexte (l'adversaire) cherche à trouver un contre-exemple à la validité de la formule et où la formule met en échec la recherche d'un contre-exemple. Les coupures s'interprètent alors comme un jeu entre deux formules duales l'une de l'autre par négation. Dans le cas des formules prouvables de la forme $\forall x \exists y. P(x, y)$ avec P décidable, la stratégie gagnante se voit comme un programme réalisant la formule. Et c'est ce programme que nous consid
ererons. Dans la pratique, au niveau des coupures, le déterminisme s'obtient comme dans **LC** : un comportement récepteur est attribué aux quantificateurs

universels et un comportement émetteur aux quantificateurs existentiels. Mais à l'opposé de **LC**, les formules sans quantificateurs n'ont pas de polarité. Etant décidables, elles sont soit vraies, soit fausses. On peut dire que le calcul de Girard donne une sémantique dénotationnelle aux preuves tandis que l'interprétation de Coquand en donne plutôt un contenu opérationnel.

Quand à Parigot, dans son système de la déduction libre, fusion de la déduction naturelle et du calcul des séquents symétrique, fait dépendre l'élimination des coupures, d'un choix de comportement entrée ou sortie pour les formules, qui force le déterminisme.

De fait, ces trois manières de faire forcent le déterminisme de l'élimination des coupures, mais fournissent des interprétations calculatoires semblable à celles obtenues, par exemple, par la A-traduction, c'est-à-dire dissymétriques. Alors que parmi les comportements posssibles de l'élimination des coupures dans le calcul original de Gentzen, on en trouve un respectant la symétrie 0/1.

4.1 Le calcul des séquents classique LK

Le calcul des séquents classique **LK** est décrit par les règles de la figure 2 de la page 21.

4.2 Une preuve symétrique dans le calcul des sequents

Notre preuve classique du théorème binaire se prête très bien à une formulation dans le calcul des séquents classique de Gentzen. La preuve telle que décrite en termes informels dans la première section est inchangée par permutation de 0 et de 1. La possibilité d'avoir plusieurs formules à droite du $\vdash$ permet de sauvegarder cette symétrie de l'argument :

La preuve du lemme classique se transcrit ainsi :

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash m \geq x_0 \qquad \dfrac{\vdash m \geq x_1 \qquad \vdash f(m) = 0, f(m) = 1}{\vdash f(m) = 0, B_1(x_1, m)}}{\vdash B_0(x_0, m), B_1(x_1, m)}}{\vdash \exists y_0 B_0(x_0, y_0), B_1(x_1, m)}}{\vdash \exists y_0 B_0(x_0, y_0), \exists y_1 B_1(x_1, y_1)}}{\vdash \forall x_0 \exists y_0 B_0(x_0, y_0), \exists y_1 B_1(x_1, y_1)}}{\vdash \forall x_0 \exists y_0 B_0(x_0, y_0), \forall x_1 \exists y_1 B_1(x_1, y_1)}$$

où m est une abréviation pour $max(x_0, x_1)$,

et $B_k(x, y)$ une abréviation pour $y \geq x \wedge f(y) = k$

On se convaincra que la manière désagréable d'obtenir $\vdash m \geq x_0 \wedge f(m) = 0, m \geq x_1 \wedge f(m) = 1$ n'est pas essentielle et tient avant tout à la binarité de l'introduction du produit.

L'autre bout de preuve, écrit pour 1 mais identique pour 0.

Règles d'identité

$$A \vdash A$$

$$\frac{\Gamma \vdash A, \Delta \qquad A, \Gamma' \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ \text{Cut}$$

Règles structurelles

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ \text{C-R} \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ \text{C-L}$$

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta}\ \text{W-R} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ \text{W-L}$$

Règles logiques

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'}\ \wedge\text{-R} \qquad \frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ \wedge\text{-L}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee\text{-R} \quad \frac{\Gamma \vdash B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee\text{-R} \qquad \frac{A, \Gamma \vdash \Delta \qquad B, \Gamma' \vdash \Delta'}{A \vee B, \Gamma, \Gamma' \vdash \Delta, \Delta'}\ \vee\text{-L}$$

$$\frac{A, \Gamma \vdash B, \Delta}{\Gamma \vdash A \supset B, \Delta}\ \Rightarrow\text{-R} \qquad \frac{\Gamma \vdash A, \Delta \qquad B, \Gamma' \vdash \Delta'}{A \supset B, \Gamma, \Gamma' \vdash \Delta, \Delta'}\ \Rightarrow\text{-L}$$

$$\frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\ \neg\text{-R} \qquad \frac{\Gamma \vdash A, \Delta}{\neg A, \Gamma \vdash \Delta}\ \neg\text{-L}$$

$$\frac{\Gamma \vdash [y/x]A, \Delta}{\Gamma \vdash \forall x A, \Delta}\ \forall\text{-R} \qquad \frac{[t/x]A, \Gamma \vdash \Delta}{\forall x A, \Gamma \vdash \Delta}\ \forall\text{-L}$$

$$\frac{\Gamma \vdash [t/x]A, \Delta}{\Gamma \vdash \exists x A, \Delta}\ \exists\text{-R} \qquad \frac{[y/x]A, \Gamma \vdash \Delta}{\exists x A, \Gamma \vdash \Delta}\ \exists\text{-L}$$

Les règles $\forall$-R et $\exists$-L ont pour restriction que la variable y ne doit pas apparaître libre dans le séquent inférieur.

Figure 2 : Règles du calcul des séquents symétrique **LK**

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{b > a \vdash b > a \qquad \dfrac{\dfrac{f(a) = 1 \vdash f(a) = 1 \qquad f(b) = 1 \vdash f(b) = 1}{f(b) = 1, f(a) = 1 \vdash f(a) = f(b) = 1}}{f(b) = 1, f(a) = 1 \vdash f(a) = f(b) = 1 \vee f(a) = f(b) = 0}}{b > a, f(b) = 1, f(a) = 1 \vdash C(a,b)}}{b > a, f(b) = 1, a \geq 0, f(a) = 1 \vdash C(a,b)}}{B_1(a+1,b), a \geq 0, f(a) = 1 \vdash C(a,b)}}{B_1(a+1,b), B_1(0,a) \vdash C(a,b)}}{B_1(a+1,b), B_1(0,a) \vdash \exists j C(a,j)}}{B_1(a+1,b), B_1(0,a) \vdash \exists i \exists j C(i,j)}}{\exists y_1 B_1(a+1,y_1), B_1(0,a) \vdash \exists i \exists j C(i,j)}}{\forall x_1 \exists y_1 B_1(x_1,y_1), B_1(0,a) \vdash \exists i \exists j C(i,j)}}{\forall x_1 \exists y_1 B_1(x_1,y_1), \exists y_1 B_1(0,y_1) \vdash \exists i \exists j C(i,j)}}{\forall x_1 \exists y_1 B_1(x_1,y_1), \forall x_1 \exists y_1 B_1(x_1,y_1) \vdash \exists i \exists j C(i,j)}}{\forall x_1 \exists y_1 B_1(x_1,y_1) \vdash \exists i \exists j C(i,j)}$$

où $C(a,b)$ est une abréviation pour $b \geq a \wedge (f(a) = f(b) = 1 \vee f(a) = f(b) = 0)$

Il y a évidemment des choix à faire dans l'ordre d'introduction des quantificateurs, mais nous verrons que les introductions de $\exists$ d'une part et de $\forall$ d'autre part, commutent de façon non essentielle.

4.3 LC : Un calcul des séquents classique à valeur calculatoire

La traduction du théorème binaire dans ce calcul conduit a un programme du même type que celle produite par la A-traduction. Ce que nous allons voir.

Nous renvoyons le lecteur au papier de Girard pour la description précise de **LC** et de la sémantique dénotationnelle associée.

LC n'étant pas spécialement préparé pour traiter les axiomes décidables, au lieu d'un axiome $\vdash f(m) = 0; f(m) = 1$, nous prendrons un axiome $\vdash; f(m) = 0 \vee f(m) = 1$, que nous utiliserons sous la forme d'une de ses conséquences : $\vdash f(m) = 0, f(m) = 1;$.

En effet :

$$\dfrac{\vdash; f(m) = 0 \vee f(m) = 1 \qquad \dfrac{\dfrac{\dfrac{\vdash \neg f(m) = 0; f(m) = 0}{\vdash \neg f(m) = 0, f(m) = 0;}}{\vdash \neg f(m) = 0, f(m) = 0, f(m) = 1;} \qquad \dfrac{\dfrac{\vdash \neg f(m) = 1; f(m) = 1}{\vdash \neg f(m) = 1, f(m) = 1;}}{\vdash \neg f(m) = 1, f(m) = 0, f(m) = 1;}}{\vdash \neg f(m) = 1 \wedge \neg f(m) = 0, f(m) = 0, f(m) = 1;}}{\vdash f(m) = 0, f(m) = 1;}$$

Nous prendrons aussi les axiomes $\vdash; m(x_0, x_1) \geq x_0$ pour réglementer les propriétés du maximum.

Par souci de simplification, nous avons modifié par rapport à **LK**, le moment d'introduction des $\exists$.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash ; m \geq x_0 \qquad \dfrac{\dfrac{\dfrac{\vdash ; m \geq x_1 \qquad \vdash f(m) = 0, f(m) = 1;}{\vdash f(m) = 0; B_1(x_1, m)}}{\vdash f(m) = 0; \exists y_1 B_1(x_1, y_1)}}{\vdash f(m) = 0, \exists y_1 B_1(x_1, y_1);}}{\vdash \exists y_1 B_1(x_1, y_1); B_0(x_0, m)}}{\vdash \exists y_1 B_1(x_1, y_1); \exists y_0 B_0(x_0, y_0)}}{\vdash \exists y_0 B_0(x_0, y_0), \exists y_1 B_1(x_1, y_1);}}{\vdash \forall x_0 \exists y_0 B_0(x_0, y_0), \exists y_1 B_1(x_1, y_1);}}{\vdash \forall x_0 \exists y_0 B_0(x_0, y_0), \forall x_1 \exists y_1 B_1(x_1, y_1);}
$$

Et l'autre bout de preuve devient dans **LC** :

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \neg b > a; b > a \qquad \dfrac{\dfrac{\vdash \neg f(a) = 1; f(a) = 1 \qquad \vdash \neg f(b) = 1; f(b) = 1}{\vdash \neg f(b) = 1, \neg f(a) = 1; f(a) = 1 \wedge f(b) = 1}}{\vdash \neg f(b) = 1, \neg f(a) = 1; f(a) = f(b) = 1 \vee f(a) = f(b) = 0}}{\vdash \neg b > a, \neg f(b) = 1, \neg f(b) = 1; C(a, b)}}{\vdash \neg b > a, \neg f(b) = 1, \neg b \geq 0, \neg f(a) = 1; C(a, b)}}{\vdash \neg B_1(a + 1, b), \neg b \geq 0, \neg f(a) = 1; C(a, b)}}{\vdash \neg B_1(a + 1, b), \neg B_1(0, a); C(a, b)}}{\vdash \neg B_1(a + 1, b), \neg B_1(0, a); \exists j C(a, j)}}{\vdash \neg B_1(a + 1, b), \neg B_1(0, a); \exists i \exists j C(i, j)}}{\vdash \forall y_1 \neg B_1(a + 1, y_1), \neg B_1(0, a); \exists i \exists j C(i, j)}}{\vdash \forall y_1 \neg B_1(a + 1, y_1), \neg B_1(0, a), \exists i \exists j C(i, j);}}{\vdash B_1(0, a), \exists i \exists j C(i, j); \exists x_1 \forall y_1 \neg B_1(x_1, y_1)}}{\vdash \neg B_1(0, a), \exists i \exists j C(i, j), \exists x_1 \forall y_1 \neg B_1(x_1, y_1);}}{\vdash \forall y_1 \neg B_1(0, y_1), \exists i \exists j C(i, j), \exists x_1 \forall y_1 \neg B_1(x_1, y_1);}}{\vdash \exists i \exists j C(i, j), \exists x_1 \forall y_1 \neg B_1(x_1, y_1); \exists x_1 \forall y_1 \neg B_1(x_1, y_1)}}{\vdash \exists i \exists j C(i, j), \exists x_1 \forall y_1 \neg B_1(x_1, y_1), \exists x_1 \forall y_1 \neg B_1(x_1, y_1);}}{\vdash \exists i \exists j C(i, j), \exists x_1 \forall y_1 \neg B_1(x_1, y_1);}
$$

Et symétriquement pour 0.

Par coupure entre ces trois bouts de preuves, nous obtenons la preuve complète du théorème binaire version classique.

Aprés élimination des coupures, on obtient un programme non symétrique relativement aux rôles joués par 0 et 1. Il est important de noter dans cet exemple que c'est l'ordre avec lequel on effectue les 2 coupures entre la preuve du lemme classique et les 2 autres morceaux de la preuve qui détermine le sens dans lequel sera orientée la dissymétrie, et non pas, comme on aurait pu penser dans une première approche, par l'ordre d'introduction des quantificateurs existentiels.

4.4 L'interprétation en terme de jeu

Imposant les mêmes comportement entrées/sorties aux formules que **LC**, le programme résultant de la preuve classique du théorème binaire est dissymétrique.

Notons que dans des cas restreints, il semble possible d'intégrer dans ce cadre des coupures simultanées, du style

$$\frac{A \vdash C \qquad \vdash A, B \qquad B \vdash D}{\vdash C, D}\, Cut'$$

avec un comportement spécifique.

4.5 La déduction libre FD

Nous ne l'avons que peu étudié, mais le procédé d'élimination des coupures que propose ce système, par le choix d'un comportement entrée ou sortie pour chaque formule, choix valable pour toutes les occurrences de la formules, impose d'obtenir en fin de compte un programme non symétrique.

Changer le comportement d'une formule, d'un endroit à l'autre de la preuve fait, d'après le géniteur de ce calcul, perdre la propriété de Church-Rosser pour le procédé de réduction.

4.6 Que se passe-t-il au niveau de l'élimination des coupures ?

Où est le hic ? Comment se fait-il qu'un exemple qui puisse s'exprimer aussi élégamment en logique classique puisse fournir une preuve aussi tordue lorsqu'on la traduit en logique intuitionniste ?

Le problème de l'élimination des coupures réside dans les différents choix que l'on peut faire pour résoudre des coupures avec contractions ou affaiblissement.

Effectivement :

$$\dfrac{\dfrac{\dfrac{\dfrac{p_1}{\vdots}}{\vdash B}}{\vdash B, A}\,\text{W-R} \qquad \dfrac{\dfrac{\dfrac{p_2}{\vdots}}{\vdash B}}{A \vdash B}\,\text{W-L}}{\dfrac{\vdash B, B}{\vdash B}\,\text{C-R}}\,\text{Cut}$$

peut se réduire de deux manières :

$$\begin{array}{c} p_1 \\ \vdots \\ \vdash B \end{array}$$

ou bien

$$\begin{array}{c} p_2 \\ \vdots \\ \vdash B \end{array}$$

Ce qui, vu du côté calculatoire, correspond à 2 programmes differents : le premier transmettant le programme p_1 de B et le deuxième le programme p_2

Ceci pour donner un exemple, car pour le théorème binaire que nous regardons ce cas là n'intervient pas. En revanche, le cas suivant, ambigu lui aussi, entre en jeu :

$$\dfrac{\dfrac{\dfrac{\vdash C,\forall xB,B'}{\vdash C,\forall xB,\forall xB}\,\forall\text{-R}}{\vdash C,\forall xB}\,\text{C-R} \qquad \dfrac{\dfrac{\forall xB,B'\vdash D}{\forall xB,\forall xB\vdash D}\,\forall\text{-L}}{\forall xB\vdash D}\,\text{C-L}}{\vdash C,D}\,\text{Cut}$$

Si l'on se place dans le cadre du calcul de Gentzen, il y a plusieurs façons de réduire cette coupure, selon la manière d'accommoder les différentes occurrences de $\forall xB$ entre elles.

Une première méthode consiste à priviligier une des occurrences d'un des côtés de la coupure :

La coupure se résoud alors en :

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash C,\forall xB,B' \qquad \dfrac{\forall xB,\forall xB\vdash D}{\forall xB\vdash D}\,\text{C-L}}{\vdash C,B',D}\,\text{Cut}}{\vdash C,\forall xB,D}\,\forall R \qquad \dfrac{\forall xB,\forall xB\vdash D}{\forall xB\vdash D}\,\text{C-L}}{\vdash C,D,D}\,\text{Cut}}{\vdash C,D}\,\text{C-R}$$

C'est cette coupure que le calcul **LC** de Girard force à utiliser : un des côtês de la coupure est duplicateur, l'autre duplique. Plus précisement, c'est le côté avec $\forall$ apparaissant à droite (c'est-à-dire récepteur) qui se duplique.

On peut aussi, symétriquement, résoudre de cette façon :

$$\dfrac{\dfrac{\dfrac{\vdash \forall xB,\forall xB,C}{\vdash \forall xBC}\,\text{C-R} \qquad \dfrac{\dfrac{\dfrac{\vdash \forall xB,\forall xB,C}{\vdash \forall xB,C}\,\text{C-R} \quad \forall xB,B'\vdash D}{B'\vdash C,D}\,\text{Cut}}{\forall xB\vdash C,D}\,\forall L}{\vdash C,C,D}\,\text{Cut}}{\vdash C,D}\,\text{C-R}$$

Les rôles entre $\forall$ et $\exists$ ont été interchangés. Cette façon de faire est, a priori possible dans **FD**, mais incompatible avec **LC**.

Une seconde méthode consiste à croiser les duplications (cross-cut) :

$$\dfrac{\dfrac{\dfrac{\vdash C,\forall xB,B' \qquad \dfrac{\forall xB,\forall xB\vdash D}{\forall xB\vdash D}\,\text{C-L}}{\vdash C,B',D}\,\text{Cut} \qquad \dfrac{\dfrac{\vdash C,\forall xB,\forall xB}{\vdash C,\forall xB}\,\text{C-R} \qquad \forall xB,B'\vdash D}{B'\vdash C,D}\,\text{Cut}}{\vdash C,C,D,D}\,Cut}{\vdash C,D}\,\text{C-R}$$

Et ce qui est remarquable, c'est que cette façon de résoudre la coupure, dans le cas du théorème binaire, fournira un programme qui est symétrique relativement aux rôles joués par 0 et 1 !

Comparé aux procédés d'élimination des coupures des différents systèmes étudiés ci-dessus et qui impose des comportements émetteur ou récepteur aux formules, valables pour la totalité de la preuve, le cross-cut est incompatible, puisque une même formule, dans une même preuve, une fois se dédouble pour recevoir son dual négatif, l'autre fois fait se dédoubler son dual.

Est-il possible d'intégrer le cross-cut dans un cadre plus général d'élimination déterministe des coupures ? Localement, on peut adapter **LC** pour qu'il élimine à la cross-cut, mais le procédé d'élimination résultant reste-t-il Church-Rosser ?

4.7 Essai d'interprétation en termes de processus communicants

L'exemple du théorème binaire se prête bien a une traduction en termes de processus communicants.

Le lemme classique pourrait s'exprimer ainsi :

Et le reste de la preuve ainsi :

Ce genre d'interprétation reste délicat, est motivé par le sens disjonctif de la partie à droite du $\vdash$ dans le calcul des séquents, mais aucune tentative réelle n'a été faite actuellement pour essayer une de généraliser le procédé.

Bibliographie

[1] H. Barendregt. *Introduction to Generalized Type System.* To appear in Proceedings of the third Italian conference on Theoretical Computer Science,

Ed. U. Moscati, Mantova, 2-4 December 1989, World Scientific Publishing Co, Singapore.

[2] N. De Bruijn. *A survey of the project Automath.* in Curry volume, Acc. Press, 1980.

[3] T. Coquand. *Une théorie des constructions.* Thèse de 3$^{\text{ème}}$ de l'université Paris 7, 1985.

[4] T. Coquand. *A semantic of evidence for classical arithmetic* Proceedings of BRA Logical Frameworks 91.

[5] T. Coquand, G. Huet. *Constructions : A higher order proof system for mechanizing mathematics.* EUROCAL85, Linz, Springer-Verlag LNCS 203, 1985

[6] M. Felleisen. *Reasoning About Continuations.* LICS 1986.

[7] H. Friedman. *Classically and Intuitionnistically Provably Recursive Functions.* In Higher Set Theory, ed. G.H. Müller and D.S. Scott. Springer Verlag Lecture Notes 669 (1978).

[8] G. Gentzen. *A consistency proof ...* in Collected papers of Gentzen, Szabo Ed. North Holland, 1969.

[9] J-Y. Girard. *A new constructive logic : classical logic.* A paraître. 1991.

[10] T. Griffin. *A Formulae-as-Types Notion of Control.* Proceedings of POPL 1990.

[11] S. Kleene. *Introduction to metamathematics.*Bibliotheca Mathematica. North Holland, 1952.

[12] A.N. Kolmogorov. *On the principle of excluded middle.* 1925. in From Frege to Gödel : a source book in mathematical logic, 1879-1931. J. Van Heijenoort. Harvard University Press, 1967.

[13] D. Leivant. *Syntactic translations and provable recursive functions.* JSL 50, 1985.

[14] P. Martin-Löf. *Intuitionnistic type theory.* Bibliopolis.

[15] C. Murthy. *Extracting Constructive Content From Classical Proofs.* Ph.D. Thesis, Cornell University, 1990.

[16] M. Parigot. *Free Deduction : an analysis of "computation" in classical logic.* A paraître. 1991.

[17] G. Stolzenberg. *A-translation : A worked example* Papier privé.

Table des matières