Les limites de la correspondance preuve/programme

Laurent Regnier
Institut de Mathématiques de Luminy

La *logique de la programmation* a pour but d'explorer les rapports profonds entre logique et informatique afin d'améliorer notre compréhension des mécanismes de calcul mis en œuvre dans les ordinateurs, et éventuellement de permettre la conception de nouveaux modèles de calcul. Elle est issue d'une découverte datant de la fin des années 1950 : l'*isomorphisme de Curry-Howard*. Elle a ainsi produit une très belle théorie des *langages fonctionnels*, une classe importante des langages informatiques. L'un des défis majeurs auxquels elle est confrontée aujourd'hui est d'étendre cette théorie à d'autres paradigmes de langages de programmation. En particulier, le développement des réseaux lui pose le problème d'accompagner le mouvement vers l'algorithmique répartie, c'est à dire de passer d'un modèle de calcul séquentiel à un modèle où les calculs sont divisés en sous-tâches effectués par des agents indépendants communicant entre eux.

Logique mathématique

C'est autour de 350 avant JC que les philosophes grecs, notamment Aristote, pose le principe fondateur de la logique : le raisonnement, le discours rationnel, a une *forme* qui peut se décrire et s'étudier indépendamment de l'objet auquel on l'applique ; avec pour conséquence la possibilité, dans une certaine mesure, de juger de la validité, de la pertinence du discours, en lui appliquant des critères formels définis une fois pour toute : les lois de la logique.

Disons tout de suite que ce principe a des limites, dont les grecs étaient évidemment conscients, et que l'on ne peut en général réduire la validité d'un discours à son strict respect d'un ensemble de lois données, aussi élégantes, riches, expressives, intelligemment formulées soient-elles. Après tout, cet ensemble de lois forme lui-même un discours que l'on ne va certainement pas valider à son tour, en vérifiant qu'il se respecte lui même[1] ! Il n'en reste pas moins que l'on peut étudier la forme du discours rationnel, chercher à dégager les principales lois formelles qu'il respecte, classifier ces lois, bref, faire de la logique. On verra plus loin que le discours mathématique en particulier, se prête étonnamment bien à cette approche.

On peut se livrer à une comparaison musicale, la logique étant au langage ce que l'harmonie est à la musique : la première étudie les règles que devrait suivre le plaideur pour construire son discours, ou plus justement, le savant pour construire ses théories ; la seconde celles que devrait suivre le compositeur pour construire son œuvre ; remarquons que l'analogie se pousse jusqu'aux limites et qu'il serait tout aussi absurde de réduire la justesse de tout discours à sa stricte conformité à un canon logique, que de réduire la beauté de toute musique à son strict respect des lois de l'harmonie.

Pendant des siècles la logique a été l'une des disciplines majeures de la philosophie. Elle n'est devenue une branche des mathématiques que tardivement, sous l'impulsion de ce que l'on a appelé la *crise des fondements*, dont la cause principale est la mutation profonde des mathématiques depuis la fin du XVII$^{\text{ème}}$. En l'espace de deux siècles, les mathématiques sont passées de l'état de science relativement « concrète », dont l'objet est principalement l'étude des nombres et des figures géométriques, à une construction d'une grande richesse, où la distinction entre algèbre et géométrie s'estompe, où les théories ne semblent s'enchevêtrer que pour mieux diverger, où de nouveaux objets généralisant abstraitement les concepts traditionnels apparaissent, ni numériques, ni géométriques, mais munis de propriétés opératoires utiles : l'exemple le plus emblématique de telles abstractions est l'invention des nombres complexes, qui s'est étalée sur plusieurs siècles et fut annonciatrice des mutations à venir.

Ce processus n'a pas été sans susciter de nombreuses interrogations sur la cohérence interne et l'unité des mathématiques, ce que l'on a appellé la *problématique des fondements*. Les mathématiciens, sous

[1] Cet argument informel contient en fait l'essentiel du sens du théorème d'incomplètude de Gödel

l'impulsion en particulier de Frege et Hilbert, ont alors découvert que l'on pouvait transformer ce questionnement philosophique en un problème rigoureusement posé et susceptible d'être étudié... mathématiquement, et créé une nouvelle théorie : la *logique mathématique*.

Celle-ci a vu démontrés ses grands théorèmes (complétude, incomplétude, élimination des coupures) dans la première moitié du XX$^{\text{ème}}$ siècle ; puis en mûrissant elle s'est éloignée des ses préoccupations initiales et a donné à son tour naissance à plusieurs disciplines : complexité logique, théorie des modèles, théorie des ensembles et la théorie de la démonstration qui va nous occuper ici.

Ajoutons pour clore cette partie, que contrairement aux espoirs des ses fondateurs, la logique mathématique n'a pas résolu la problématique des fondements, bien au contraire. N'importe, elle s'était déjà suffisamment développée et enrichie pour se trouver d'autres raisons d'exister. Cet article propose présenter certaines des motivations modernes d'une partie des logiciens, en particulier liées à l'informatique.

Formalisation des preuves. La logique mathématique repose sur la possibilité de *formaliser* le raisonnement mathématique. C'est à dire que l'on peut définir un système de notations (un système formel), permettant de représenter les énoncés et les raisonnements mathématiques, un peu comme le solfège définit un système de notations musicales. La propriété fondamentale de cette représentation est d'être testable effectivement : il existe des procédures permettant de vérifier automatiquement qu'un énoncé est bien formé (par exemple que toutes les parenthèses ouvertes sont fermées) et qu'un raisonnement est correct, c'est à dire qu'il ne contient pas d'erreur logique. De même on peut imaginer des programmes capables de procéder à l'analyse d'une partition et de vérifier qu'elle respecte certaines règles d'harmonie.

On peut remarquer ici la situation singulière des mathématiques, dont le discours contraint par l'exigence de rigueur, est *entièrement* formalisable. Il est bien clair que tel n'est pas le cas de la musique où une partition, aussi précisément écrite soit-elle, ne peut rendre toutes les nuances d'interprétation (pour les amateurs de jazz, penser à la difficulté, voire l'impossibilité, à définir le « swing »).

Tout énoncé mathématique, toute preuve peut s'écrire dans le système de notation logique. Il ne s'agit pas là d'une constation empirique mais d'un théorème, l'un des premiers de la logique mathématique : le théorème de *complétude* de Gödel (à ne pas confondre avec ses deux théorèmes d'incomplètudes).

Insistons également sur le fait que la formalisation ne définit pas la *vérité* qui est une propriété intrinsèque des énoncés mathématiques, mais la *prouvabilité* qui est une propriété des objets formels dénotant ces énoncés. De même, le fait que l'on puisse écrire la partition d'une pièce musicale, et vérifier qu'elle est harmoniquement conforme, n'entraîne pas que celle-ci soit un chef-d'œuvre. Il existe bien entendu un relation entre prouvabilité et vérité : ce qui est prouvable est vrai. Mais la réciproque est plus que problématique comme le montre le théorème d'*incomplétude* de Gödel qui établit clairement que la prouvabilité n'est et ne restera toujours qu'une approximation de la vérité. Par exemple il n'existe pas de définition mathématiquement *complète* du concept, apparemment simple, de nombres entiers. C'est à dire que l'on ne peut trouver une définition mathématique des entiers à partir de laquelle on pourrait dériver logiquement toute propriété des entiers. On en est donc réduit à travailler avec des définitions approximatives (ce qui, en pratique, ne gêne pas grand monde puisque on en connait de très bonnes permettant de démontrer ou réfuter tout ce qu'il est raisonnable d'imaginer).

Théorie de la démonstration

Il s'agit de l'étude des systèmes formels dénotant le discours mathématique. Nous allons voir qu'elle a vécu un véritable révolution qui l'a menée, de la problématique des fondements à l'informatique, et à donner naissance à la « logique de la programmation », qui se donne pour but l'étude des langages informatiques et de leur dynamique par des moyens logiques.

Le calcul des séquents. En 1936 le logicien allemand Gerhard Gentzen démontrait un résultat qui devait devenir central en théorie de la démonstration : le Hauptsatz ou théorème *d'élimination des coupures*. Ce résultat repose sur un système formel inventé par Gentzen, appelé *calcul des séquents*. Comme tout système logique, le calcul des séquents se construit en deux étapes.

Dans un premier temps on se donne un formalisme pour représenter les énoncés mathématiques. On utilise pour cela les *formules* du *calcul des prédicats*. Il s'agit d'un système de notation en deux niveaux. On commence par se donner des symboles pour dénoter des *objets*, par exemple 0 pour dénoter l'entier zéro et 1 pour dénoter l'entier un ; puis on fixe les noms de certaines opérations qui vont nous permettre de construire des objets complexes, par exemple + pour l'addition entre entiers (qui nous permet déjà de dénoter tous les entiers en utilisant seulement les notations déjà définies, par exemple

deux est dénoté par $1+1$, trois par $1+1+1$), $\cdot$ pour la multiplication, etc. Ensuite on se donne des noms pour quelques *relations* élémentaires entre objets, par exemple $=$ pour l'égalité ; en les combinant au moyen des connecteurs logiques on peut alors construire des relations complexes. Rappelons rapidement les notations pour les connecteurs : la négation ($\neg$), la conjonction ($\wedge$), la disjonction ($\vee$), l'implication ($\rightarrow$), les quantificateurs existentiel ($\exists$) et universels ($\forall$). À titre d'exemple, dans le système de notation que l'on vient de définir voici une formule exprimant que deux entiers m et n sont premiers entre eux, c'est à dire qu'ils n'ont aucun diviseur commun autre que 1 :

$$\neg(n=0) \wedge \neg(m=0) \wedge \forall x \forall y \forall z((n = x \cdot y \wedge m = x \cdot z) \rightarrow x = 1)$$

ce qui se lit « n et m sont non nuls et pour tous entiers x, y et z, si n est égal au produit de x par y et si m est égal au produit de x par z alors x est égal à 1 ».

Dans un second temps, on se donne des *règles* c'est à dire des relations entre formules qui déterminent les conditions pour qu'une formule se dérive logiquement d'une ou plusieurs autres. L'histoire de la logique est jalonnée par différents systèmes de règles logiques. Les grecs anciens avaient leurs *syllogismes* (« Tous les hommes sont mortels, Socrate est un homme donc... »). La logique moderne est née avec ce que l'on appelle aujourd'hui les *systèmes à la Hilbert* dans lesquels la règle principale, le *modus ponens*, dit que l'on peut dériver logiquement une formule B des formules A et $A \rightarrow B$.

Le calcul des séquents lui introduit une petite nouveauté en ceci que ses règles permettent de dériver des objets un peu plus généraux que des formules : des *séquents*. Un séquent est une énoncé formel qui s'écrit $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ et qui se lit : « si on se donne les hypothèses A_1, ..., A_n alors l'une au moins des conclusions B_1, ..., B_m est démontrable ». Par exemple, si on veut dire qu'une formule B est démontrable, on écrit le séquent $\vdash B$ qui se lit « si on ne se donne aucune hypothèse, la formule B est démontrable ». Si on veut dire que deux formules A et B sont contradictoires on écrit $A, B \vdash$. En effet, l'interprétation que l'on a donnée entraîne qu'un séquent reste vrai si on lui ajoute des formules à gauche ou à droite ; donc le séquent $A, B \vdash$ reste vrai si on lui ajoute une formule C à droite, c'est à dire que si $A, B \vdash$ est vrai, alors $A, B \vdash C$ aussi, pour une formule C quelconque. Autrement dit de A et B on peut déduire n'importe quoi, ce qui montre bien qu'elles sont contradictoires.

Pour construire des preuves formelles permettant de démontrer des séquents (on dit aussi *dériver* des séquents), Gentzen a défini trois groupes de règles.

- Le groupe *structurel* exprime des propriétés générales : la logique est *commutative*, c'est à dire que l'on peut permuter à volonté l'ordre des formules à gauche ou à droite du symbole $\vdash$; ce qui signifie par exemple que si un énoncé B est démontrable sous les hypothèses A_2 et A_1 (c'est à dire si le séquent $A_2, A_1 \vdash B$ est dérivable) alors B est également démontrable sous les hypothèses A_1 et A_2.
  La logique est *idempotente* : si un séquent qui contient plusieurs fois la même formule à gauche (ou à droite) est dérivable alors celui où l'on a contracté toutes ces occurrences en une seule formule l'est aussi. Par exemple si $A, A \vdash B$ (« sous les hypothèses A et A, la formule B est démontrable ») est dérivable, alors $A \vdash B$ l'est aussi. C'est donc une généralisation du principe que l'on peut utiliser une hypothèse autant de fois que l'on veut pour démontrer un résultat.
  La logique est également *affine* c'est à dire que l'on peut toujours ajouter des hypothèses ou des conclusions inutiles à un séquent : par exemple si le séquent $\vdash B$ est dérivable alors le séquent $\vdash B, C$ l'est également ; de fait puisque B est démontrable sans hypothèses, certainement l'une au moins des formules B ou C est démontrable sans hypothèses. Remarquons qu'une telle règle semble inutile puisque savoir démontrer B est en général plus intéressant que savoir démontrer B ou C. Elle n'en exprime pas moins une propriété importante de la logique et l'on va voir qu'elle est indispensable au même titre que les autres.
- le groupe *logique* définit les connecteurs ; par exemple il y a deux règles pour la conjonction ($\wedge$). La première dit essentiellement que pour démontrer $A \wedge B$ il faut démontrer A et démontrer B tandis que la seconde dit que pour utiliser une hypothèse $A \wedge B$, il suffit de se donner les deux hypothèses A et B.
- le groupe *identité* est constitué de deux règles : la règle *axiome* exprime la tautologie fondamentale (sic), à savoir que A est démontrable sous l'hypothèse A ; et la règle la plus importante du système : la *coupure*.

Une preuve (ou une dérivation) se construit alors en partant des règles axiomes et en appliquant à ceux-ci des règles ce qui produit de nouveaux séquents auxquels on peut à nouveau appliquer des règles, etc. Par exemple si on part du séquent $A \vdash A$ qui est donné par une règle axiome, on peut lui appliquer

une règle de négation droite ce qui produit le séquent $\vdash \neg A, A$, puis une négation gauche ce qui donne $\neg\neg A \vdash A$ et enfin un implication droite résultant en $\vdash \neg\neg A \to A$. On a ainsi fabriqué une dérivation logique du principe de raisonnement par l'absurde, qui se représente comme suit :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\, ax}{\vdash \neg A, A}\, \neg\text{-}droite}{\neg\neg A \vdash A}\, \neg\text{-}gauche}{\vdash \neg\neg A \to A}\, \to\text{-}droite$$

Avant de discuter la règle de coupure il convient de relever l'originalité du calcul des séquents par rapport aux systèmes qui l'ont précédé. Comme on a vu, la logique mathématique est née de la crise des fondements et a commencé à se développer en tant que théorie du langage mathématique. Les premiers systèmes, les systèmes « à la Hilbert » devaient répondre à la double contrainte d'une part de donner une définition rigoureuse de preuve formelle, d'autre part de permettre la représentation de toute preuve mathématique. Gentzen lui même a commencé par inventer un système appelé *déduction naturelle*, appelé ainsi car il simplifiait les systèmes à la Hilbert en les rapprochant plus encore de la pratique mathématique.

Par contre même si l'on peut se convaincre que les règles du calcul des séquents permettent bien d'exprimer toute preuve mathématique, celles-ci, à l'exception notable de la règle de coupures, semblent pour le moins peu naturelles. En fait le calcul des séquents cherche à exprimer la structure même de la logique. Ainsi il met en valeur les symétries : toutes ses règles, sauf celles du groupe identité, ont une version gauche et une version droite ; les règles de conjonction sont les images miroir de celles de la disjonction. C'est également la première fois qu'apparaît la notion de propriétés *structurelles* de la logique. Et le théorème de Gentzen est en un sens un théorème de logique pure, au contraire des théorèmes de Gödel qui parle de la relation entre logique et mathématique. Le calcul des séquents est un outil d'étude de la logique elle même, et non plus du discours mathématique.

La règle de coupure. En mathématique, tout comme en montagne, le chemin le plus court menant à un théorème n'est en général pas la ligne droite. On peut choisir l'approche directe, ce qui présente l'avantage lorsque l'on ne connait pas le terrain, que le sommet est toujours en vue, donc que l'on n'hésite jamais sur la direction à prendre ; mais, pour peu que le terrain soit accidenté, cette stratégie risque de nous mener, au mieux dans une impasses, au pire dans une crevasse. Il est alors plus efficace de se munir d'une carte, s'il en existe de la région, et d'explorer (prudemment) le terrain pour essayer de s'y frayer un chemin praticable.

Le mathématicien est un peu dans la même situation, l'approche directe d'un problème est souvent vouée à l'échec, et il est bon de se munir d'une carte, c'est à dire de tout son savoir mathématique pour trouver son chemin. Tout l'art consiste alors à savoir inventer les objets adéquats, à trouver les résultats intermédiaires et les bonnes généralisations, aux énoncés parfois très éloignés du but poursuivi, mais qui y conduisent par des voies parfois mystérieuses (et pittoresques). Les mathématiques se construisent souvent ainsi, pour résoudre un problème difficile, on en construit des généralisations, on développe des théories nouvelles qui elles mêmes recèlent de nombreuses et excitantes questions et se mettent alors à vivre une vie indépendante. Parfois, mais pas toujours, elles permettent d'atteindre le but initial.

Pour formaliser une preuve qui emprunte des voies détournées ou qui utilise des théorèmes généraux, ce qui est souvent le cas, la règle de coupure est indispensable. C'est en effet la seule (dans le calcul des séquents) qui autorise l'utilisation d'un résultat intermédiaire. Observons le *modus ponens*, qui en est une version simplifiée : cette règle nous dit que de A et de $A \to B$ on peut déduire B. Autrement dit, pour démontrer B il suffit de trouver un énoncé A, typiquement une généralisation de B, tel que d'une part on puisse démontrer A, et d'autre part on puisse démontrer B en supposant A (ce qui est facile si par exemple B est un cas particulier de A). L'astuce consiste à chercher parmi tous les théorèmes de mathématiques, l'énoncé A duquel on pourra facilement déduire notre B (remarquons que ce genre d'« astuce » peut occuper des générations de mathématiciens).

C'est en ce sens que la règle de coupure est la plus importante du système de Gentzen. Les autres règles, structurelles et logiques, sont certes indispensables mais ne laissent aucune place à la créativité, elles sont de la logique pure. L'expérience consistant à prendre une preuve mathématique quelconque (mais pas trop longue) et l'écrire en calcul des séquents montre au moins une chose : la coupure est la règle la plus utilisée en mathématique, les autres ne jouent qu'un rôle marginal. Autrement dit, les mathématiques ne se réduisent pas à de la pure logique.

$$\frac{}{A \vdash A}\ ax \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma', A \vdash \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'}\ coupure$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ \textit{aff-gauche} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ \textit{aff-droite}$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ \textit{idem-gauche} \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ \textit{idem-droite}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ \neg\textit{-gauche} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\ \neg\textit{-droite}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\ \wedge\textit{-gauche} \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ \wedge\textit{-droite}$$

$$\frac{\Gamma, A \vdash \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\ \vee\textit{-gauche} \qquad \frac{\Gamma \vdash A, B \Delta}{\Gamma \vdash A \vee B, \Delta}\ \vee\textit{-droite}$$

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\ \to\textit{-gauche} \qquad \frac{\Gamma, A \vdash B \Delta}{\Gamma \vdash A \to B, \Delta}\ \to\textit{-droite}$$

Les lettre grecques Γ et Δ représentent des suites, possiblement vides, de formules dans lesquelles on ne tient pas compte de l'ordre. Une démonstration commence (quand on la lit de haut en bas) par des règles *ax*.
À titre d'exemple, voici une démonstration de la formule $(\neg B \to \neg A) \to (A \to B)$; à chaque étape on a écrit en caractères gras les formules qui servent à produire celle du dessous au moyen de la règle.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\ ax}{A \vdash \boldsymbol{A}, B}\ \textit{aff-droite}}{\neg \boldsymbol{A}, A \vdash B}\ \neg\textit{-gauche} \qquad \dfrac{\dfrac{\dfrac{}{B \vdash B}\ ax}{A, \boldsymbol{B} \vdash B}\ \textit{aff-gauche}}{A \vdash \neg \boldsymbol{B}, B}\ \neg\textit{-droite}}{\neg B \to \neg A, \boldsymbol{A} \vdash \boldsymbol{B}}\ \to\textit{-gauche}}{\dfrac{\neg \boldsymbol{B} \to \neg \boldsymbol{A} \vdash \boldsymbol{A} \to \boldsymbol{B}}{\vdash (\neg B \to \neg A) \to (A \to B)}\ \to\textit{-droite}}\ \to\textit{-droite}$$

FIG. 1 – Le calcul des séquents

Élimination des coupures. Le résultat principal de Gentzen (*Hauptsatz* en allemand) énonce que s'il existe une dérivation d'un séquent donné, alors il existe une dérivation du même séquent qui *n'utilise pas la coupure*. En fait Gentzen définit explicitement une procédure, appelée *élimination des coupures*, pour construire une telle preuve à partir de celle de départ.

Dès qu'un énoncé mathématique est prouvable, il l'est également sans utiliser la règle de coupure ! On peut ramener toute démonstration mathématique à une démonstration purement logique. La situation semble un peu paradoxale. C'est un peu comme de dire que l'on peut écrire la musique sans utiliser de notes.

La démonstration du théorème consiste à sélectionner dans la dérivation de départ une occurrence de la règle de coupure et à découper la dérivation en différentes sous-preuves qui sont connectées à cette coupure. On remplace ensuite la coupure par un petit morceau de preuve n'utilisant que des règles de coupures plus simples et on reconstruit une preuve en lui reconnectant les sous-preuves. On obtient ainsi un dérivation beaucoup plus grosse que celle de départ car on a remplacé une règle par plusieurs, mais surtout parceque l'opération de reconstruction utilise en général plusieurs fois chacune des sous-preuves. Pour peu que l'on ait trouvé la notion adéquate de « plus simple », on montre que en itérant le procédé, les coupures se simplifient jusqu'à disparation.

Sans entrer plus dans les détails, retenons un point important : selon la preuve dont on est parti le calcul de la preuve sans coupure peut être long. On peut facilement construire des preuves formelles telles que leur *normalisation*, c'est à dire le calcul de leur version sans coupure (leur *forme normale*), produise un résultat largement plus gros que la totalité des moyens de stockage d'information disponibles sur terre. Certes toute preuve mathématique peut se ramener à de la logique pure, mais au prix d'une

insupportable perte de concision et d'élégance. La question de l'utilité et du sens du théorème de Gentzen semble se poser, mais en fait cette question est liée au point de vue traditionnel définissant la logique comme science du raisonnement, analyse formelle du discours mathématique ; le théorème de Gentzen propose une autre façon de voir la logique, plus interne, mais aussi potentiellement riche en nouveaux développements.

> Voici un exemple de preuve avec coupure, aimablement communiqué par mon collègue Yves Lafont. On veut couvrir un quadrillage 8x8 avec des dominos rectangulaires ayant exactement la taille de deux cases. C'est assez facile mais beaucoup moins si l'on suppose en plus que l'on a supprimé les deux cases de part et d'autre de l'une des grandes diagonales. C'est alors impossible et c'est ce que nous allons maintenant démontrer.
> Pour cela on colorie notre quadrillage en blanc et noir à la façon d'un échiquier ; il est clair le coloriage ne change rien au problème. On remarque que chaque domino couvrant deux cases adjacentes, celles-ci sont de deux couleurs différentes, donc on a le
>
> **Lemme 1** *Toute couverture du quadrillage doit masquer exactement le même nombre de cases blanches et de cases noires.*
>
> Or on a enlevé deux cases de part et d'autre d'une diagonale ; celles-ci sont donc de même couleur, par exemple noire, si bien que le quadrillage ne compte que 30 cases noires pour 32 cases blanches. D'après le lemme, si une couverture existe alors le quadrillage doit compter le même nombre de cases blanches et noires, ce qui n'est pas le cas. Donc aucune couverture n'existe. CQFD
> Pour prouver l'impossibilité, on a donc ajouté une structure (le coloriage des cases) qui ne change pas la nature du problème, mais que l'on a astucieusement « sorti de son chapeau » et qui permet de trouver facilement le résultat. C'est un exemple typique d'utilisation de coupure.
> On peut s'interroger sur ce que serait une preuve sans coupure de ce petit théorème. Sans entrer dans les détails de la formalisation, on peut dire qu'un telle preuve consiste essentiellement à énumérer toutes les tentatives de couvertures, et vérifier pour chacune qu'elle échoue à couvrir tout le quadrillage : un travail long et fastidieux, et qui devient irréalisable, même à l'aide d'ordinateur, si l'on augmente la taille du quadrillage

FIG. 2 – Une preuve avec coupure.

L'importance du théorème de Gentzen en logique. En premier lieu le Hauptsatz affirme que toute preuve mathématique peut, au moins en théorie, se faire en utilisant uniquement les propriétés structurelles de la logique et les définitions des connecteurs ; autrement dit cet ensemble de principes très élémentaires est déjà assez riche pour permettre de tout démontrer (en théorie répétons le), techniquement on dit qu'il est *complet*. C'est un résultat conceptuellement intéressant, et qui a des applications en démonstration automatique, on y reviendra.

Observons d'abord d'un peu plus près ce qu'est une preuve sans coupure. La remarque fondamentale est que toutes les règles du calcul des séquents, à l'exception de la coupure, sont croissantes au sens où elles ne font jamais disparaître de formule lorsqu'on les lit de haut en bas. Pour chacune des règles, les formules qui sont au-dessus de la barre (dans les séquents *prémisse* de la règle) apparaissent également au-dessous (dans le séquent conclusion), éventuellement combinées au moyen de connecteur.

Par conséquent, dans une preuve sans coupure, le dernier séquent de la preuve est le plus compliqué. Autrement dit, on n'y utilise pas de résultat intermédiaire, c'est à dire que tous les énoncés qui y apparaissent sont pour ainsi dire « contenus » dans le théorème que l'on démontre ; c'est ce qu'on appelle la *propriété de la sous-formule*.

Une application très répandue en théorie de la démonstration concerne les preuves de *cohérence*. Il s'agit de prendre une théorie mathématique, définie par un ensemble d'axiomes, et de montrer qu'elle est non contradictoire, c'est à dire que ces axiomes n'ont pas de conséquence absurde (par exemple $0 = 1$). Le second théorème d'incomplètude de Gödel nous dit que l'on ne peut espérer démontrer la cohérence que en se plaçant dans une autre théorie, « plus forte ». Les preuves de cohérence forment donc une classification des théories mathématiques, et ont été pendant longtemps l'un des sujets de recherche de prédilection en théorie de la démonstration. Le théorème de Gentzen propose une méthode très puissante pour résoudre ces problèmes : en effet il n'y a pas de formule plus simple que $0 = 1$; mais une preuve

sans coupure de $0 = 1$ ne contient que des formules plus simples que $0 = 1$; donc il n'y a pas de preuve sans coupure de $0 = 1$. Si on a réussi à démontrer l'élimination des coupures pour le calcul des séquents exprimant la théorie, on en déduit qu'il n'y a pas de preuve du tout de $0 = 1$, donc que la théorie est cohérente.

Ainsi le problème de la cohérence est transformé en deux problèmes : (1) étendre le calcul des séquents pour permettre d'y exprimer tous les axiomes de la théorie considérée ; (2) montrer l'élimination des coupures pour le calcul étendu. Gentzen a ainsi utilisé cette approche pour montrer la cohérence de l'arithmétique de Peano, l'une des définitions mathématiques des entiers. Cette méthode a un aspect quantitatif intéressant : on a vu que la taille des preuves augmente au cours de la procédure ; on peut essayer de mesurer cette augmentation ce qui fournit un outil de comparaison entre théories mathématiques.

L'élimination des coupures a également beaucoup d'importance en démonstration automatique. Un ordinateur n'a pas d'imagination, c'est à dire qu'il ne peut utiliser intelligemment la règle de coupure. Par contre on peut facilement construire des algorithmes de recherches de preuves sans coupure. Un tel algorithme ne nous aidera guère à faire des mathématiques car, comme on a vu, les preuves sans coupure des théorèmes sont de taille trop élevée[2]. Mais il existe des classes de problèmes dont les preuve sans coupures restent raisonnable (à l'échelle de l'ordinateur) : un exemple important est les *preuves de programme*. On se donne un programme, par exemple celui qui pilote le métro automatisé Météor, et on le « prouve », c'est à dire que l'on construit une preuve formelle que les collisions entre trains sont impossibles. D'un point de vue mathématique le problème est inintéressant, et de plus hors de portée de moyens humain, car il s'agit plus d'une vérification longue et fastidieuse que d'une réelle démonstration, c'est à dire d'une preuve sans coupure. Un logiciel de démonstration automatique par contre est tout à fait approprié.

L'isomorphisme de Curry-Howard

L'isomorphisme de Curry-Howard est une rencontre entre la théorie de la démonstration et l'informatique qui a donné naissance à la logique de la programmation. Jusqu'alors la logique était l'étude du raisonnement, c'est à dire des relations entre *formules*, les preuves formelles n'étant que des moyens d'exprimer ces relations. L'isomorphisme de Curry-Howard déplace le centre d'intérêt vers les preuves et l'élimination des coupures.

Les preuves sont des programmes. Considérons une preuve d'un énoncé B sous l'hypothèse A, c'est à dire une preuve du séquent $A \vdash B$. Si on se donne par ailleurs une preuve sans coupure de A, c'est à dire du séquent $\vdash A$, on peut « brancher » ces deux preuves par le truchement d'une règle de coupure, et obtenir une preuve de B (du séquent $\vdash B$). Si on applique la procédure d'élimination des coupures, on va obtenir une preuve sans coupure de B. La preuve de $A \vdash B$ peut donc se comprendre comme une *fonction* qui associe à une preuve sans coupure de A une preuve sans coupure de B. Remarquons que lorsque l'on parle de fonction, ça n'est pas vraiment au sens usuel en mathématique et que l'on devrait plutôt dire un *programme*, puisqu'il y a un calcul pour arriver au résultat.

L'isomorphisme de Curry-Howard, ou comme on l'appelle aussi, la *correspondance preuve/programme*, consiste à interpréter les démonstrations comme des programmes et l'élimination des coupures comme l'exécution de ces programmes. Par exemple comme on vient de le voir, une démonstration d'un énoncé $A \to B$ peut aussi se voir comme un programme associant à toute démonstration de A une démonstration de B. La formule qui conclut une démonstration est désormais interprétée comme un *type*, c'est à dire une spécification de programme. L'isomorphisme de Curry-Howard identifie une preuve de la formule A à un programme de type A (ou vérifiant la spécification A). Les programmes étant faits pour fournir des résultats, c'est les preuves sans coupures qui vont jouer ce rôle.

Cette interprétation autorise une lecture positive du problème de la taille des preuves sans coupures. Un langage de programmation qui ne permettrait d'implémenter que des algorithmes rendant un résultat petit serait sans doute trop pauvre. En effet les programmes ont souvent la propriété que leur sortie est très grosse par rapport à leur entrée : par exemple un traitement de texte prend en entrée un fichier de quelques dizaines de milliers de caractères et produit une image comptant des millions de pixels. Le fait qu'une preuve puisse grossir pendant l'élimination des coupures doit s'interpréter comme exprimant la richesse du pouvoir expressif de la logique en tant que langage de programmation.

[2] Par contre on peut utiliser des algorithmes de démonstrations automatiques à des fins pédagogiques

Un changement de point de vue : des formules aux preuves. Insistons sur le fait que l'isomorphisme de Curry-Howard induit un changement profond dans la manière d'envisager la logique. Jusqu'alors, une démonstration formelle est une manière de représenter le cheminement qui, partant des axiomes de la théorie mène au théorème. L'accent est mis sur les formules ; les questions sont relatives aux formules : sont-elles prouvables ? sont-elles cohérentes ? La prouvabilité est en fait pensée comme une approximation constructive du concept idéal de vérité et dans cet esprit ce qui importe est l'existence ou non d'une démonstration.

Le théorème de Gentzen introduit déjà un déplacement d'intérêt vers les preuves qu'il classifie sommairement en démonstrations avec ou sans coupure et pour lesquelles il énonce des propriétés, comme celle de la sous-formule.

La correspondance preuve/programme va beaucoup plus loin dans cette direction. Désormais on va étudier les démonstrations relativement à leur comportement au travers de l'élimination des coupures sans plus s'intéresser à leur contenu mathématique. Par exemple on va voir ci-dessous l'exemple d'une tautologie, c'est à dire une formule vide de sens du point de vue mathématique mais qui peut s'interpréter comme le type des entiers. Très significativement, la logique de la programmation a inventé et étudié des systèmes logiques dans lesquels il est pour le moins malcommode de représenter des propriétés mathématiques, mais qui sont conçus pour analyser en profondeur l'élimination des coupures. La *logique linéaire* de Jean-Yves Girard est un exemple d'un tel système. Inventée en 1986, elle repose sur une analyse des propriétés structurelles de la logique (idempotence, affinité) et est aujourd'hui un outil fondamental en logique de la programmation.

Représentation des entiers par des preuves. Une fois la correspondance de Curry-Howard comprise, la première question qui se pose est celle du pouvoir expressif des démonstrations en tant que programmes : peut on représenter l'addition des entiers par une démonstration formelle ? et la division euclidienne ? et le calcul des décimales de π ?

Commençons par voir comment les démonstrations permettent de représenter les entiers ; en théorie si on peut faire cela, alors il n'y a pas de limite à ce que l'on peut programmer (après tout les ordinateurs ne connaissent que les entiers, codés sous forme de suite de 0 et de 1). En pratique d'autres questions se posent relativement à l'efficacité de ce mode de calcul.

Nous allons examiner les preuves sans coupures de la formule $(A \to A) \to (A \to A)$ que nous noterons $\mathcal{N}$. Une fois de plus insistons sur le fait que cette formule ne nous intéresse pas pour ce qu'elle dit (du reste elle ne dit rien, c'est une tautologie), mais pour les propriétés de ses preuves sans coupures.

Le calcul des séquents, et plus particulièrement les règles du connecteur $\to$, nous dit que démontrer cette formule revient à démontrer la formule A en supposant les hypothèses $A \to A$ et A, c'est à dire le séquent $A \to A, A \vdash A$. Il y a une infinité de manière de faire sans utiliser la règle de coupure selon le nombre de fois que l'on utilise l'hypothèse $A \to A$.

Plus précisément, au prix de quelques contraintes techniques, on peut assurer que pour chaque entier n il existe une preuve sans coupure de la formule $\mathcal{N}$ qui utilise exactement n fois l'hypothèse $A \to A$. Ces preuves sont des candidats naturels à représenter les entiers et l'on peut dire qu'elles « codent » les entiers dans le calcul des séquents. Voici par exemple les preuves représentant les entiers 0 et 2 :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A}\,ax}{A \to A, \boldsymbol{A} \vdash \boldsymbol{A}}\,\textit{aff-gauche}}{\boldsymbol{A \to A} \vdash \boldsymbol{A \to A}}\,\to\textit{-droite}}{\vdash (A \to A) \to (A \to A)}\,\to\textit{-droite}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash \boldsymbol{A}}\,ax \quad \dfrac{}{\boldsymbol{A} \vdash A}\,ax}{A \to A, A \vdash \boldsymbol{A}}\,\to\textit{-gauche} \quad \dfrac{\dfrac{}{A \vdash A}\,ax}{A \to A, \boldsymbol{A} \vdash A}\,\textit{aff-gauche}}{\boldsymbol{A \to A}, \boldsymbol{A \to A}, A \vdash A}\,\to\textit{-gauche}}{A \to A, \boldsymbol{A} \vdash \boldsymbol{A}}\,\textit{idem-gauche}}{\boldsymbol{A \to A} \vdash \boldsymbol{A \to A}}\,\to\textit{-droite}}{\vdash (A \to A) \to (A \to A)}\,\to\textit{-droite}$$

La première utilise 0 fois l'hypothèse $A \to A$ qui a été introduite par une règle d'affinité ; la seconde l'utilise 2 fois, et pour ce faire elle utilise une règle d'idempotence.

Pour résumer on a trouvé une classe de preuves de la même formule $\mathcal{N}$ qui représente les entiers. Une preuve du séquent $\mathcal{N} \vdash \mathcal{N}$ peut alors être comprise comme un programme qui rend un entier lorsqu'on lui donne un entier en entrée. Il s'avère que l'on peut ainsi construire des preuves qui vont représenter toutes les opérations usuelles sur les entiers.

Logique de la programmation

Nous allons finir cet article en donnant un aperçu rapide de certains défis modernes qui sont adressés à la logique de la programmation. On a pris le parti ici d'exposer prioritairement ceux qui sont en rapport avec l'informatique, et omis un certain nombre de travaux courants sur des sujets dont les motivations sont moins tournées vers les applications. On peut classifier ces défis selon les différents niveaux de la correspondance preuve/programme.

Au niveau des formules. L'isomorphisme de Curry-Howard nous dit que les formules de la logique peuvent être vues comme des types, c'est à dire des spécifications de programmes. Le théorème de Gentzen peut alors être compris comme énonçant la *correction* du programme/preuve, c'est à dire qu'il garantit que le résultat du calcul sera bien du type spécifié, donc vérifiera bien la spécification.

En d'autres termes, si on sait exprimer une spécification de programme (on dit aussi un cahier des charges) comme une formule mathématique, alors toute preuve de cette formule est un programme dont le théorème de Gentzen nous assure qu'il réalise cette spécification. On obtient ainsi une méthode générale pour construire des programmes prouvés, en un sens le programme est en même temps la preuve de sa correction.

Cette méthode pose toutefois plusieurs problèmes. En premier lieu il n'est pas toujours évident de transformer une spécification en une formule. En second lieu, même lorsque l'on réussit à passer cette première étape, le programme que l'on obtient en prouvant la formule n'est pas toujours très efficace. Plus généralement le problème est de construire des systèmes logiques suffisamment expressifs pour pouvoir d'une part exprimer des spécifications de programmes élaborées, d'autre part fournir des méthodes de preuves qui tout en restant compatibles avec le théorème de Gentzen, autorisent l'implémentation d'algorithmes performants. Ces deux problèmes font aujourd'hui l'objet de recherches actives.

On peut aussi prendre la question des spécifications à l'envers. Considérons un théorème de mathématiques. La correspondance de Curry-Howard nous dit qu'une preuve de ce théorème est un programme, mais que fait ce programme ? Autrement dit, comment comprendre un énoncé mathématique comme un type ? On a vu une réponse à cette question pour la formule $\mathcal{N}$, dont les preuves peuvent être vues comme des représentations des entiers, et pour la formule $\mathcal{N} \to \mathcal{N}$ dont les preuves implémentent des fonctions sur les entiers. Que se passe-t-il lorsque l'on prend des formules plus élaborées, de vrais théorèmes mathématiques ? Ces questions ont été posées par le logicien Jean-Louis Krivine, qui a obtenu un certain nombre de résultats remarquables, en particulier en analysant les programmes réalisant le théorème de complétude, ou certains axiomes de la théorie des ensembles.

Au niveau de l'élimination des coupures. La logique mathématique ne s'occupe pas vraiment de définir la vérité, concept qui lui échappe comme le démontre le théorème de Gödel. Se pose alors la question de justifier ses règles. On peut adopter une approche pragmatique en disant simplement qu'elles sont correctes parcequ'elles permettent la formalisation des mathématiques et que celles-ci ont amplement justifié leur pertinence par les réalisations scientifiques et technologiques qui en découlent. Mais on peut également chercher une explication plus intrinsèque. Une approche possible à ce problème est fournie par le théorème de Gentzen, qui dit : « les règles sont correctes parcequ'elles permettent l'élimination des coupures » ; cette approche fait de l'élimination des coupures l'objet primitif à partir duquel on devrait pouvoir reconstruire la logique.

Ce programme a été formulé explicitement dans la *ludique* de Jean-Yves Girard, qui le résume par le slogan évocateur : « des règles de la logique à la logique des règles ». Les « règles de la logique » sont définies par les systèmes formels et une analyse fine de leur interaction au travers de l'élimination des coupures devrait mener à « la logique des règles », c'est à dire aux lois profondes de la logique.

Au niveau des preuves. Si d'après l'isomorphisme de Curry-Howard les preuves sont des programmes, elles ne sont pas tous les programmes, tant s'en faut. L'informatique moderne a développé un grand nombre de concepts, de plus en plus abstraits, visant à faciliter la construction et la coordination de gros projets, le déboguage et la maintenance des logiciels, la mise en place de services réseau, etc. Il existe donc énormément de programmes, mais peu d'entre eux peuvent être vus comme des preuves dans un système logique adéquat.

C'est pourquoi une partie de l'activité en logique de la programmation consiste à chercher à rendre compte par des moyens logiques de divers principes informatiques. Dans les dix dernières années on a ainsi découvert que l'isomorphisme de Curry-Howard, qui originellement établissait une correspondance entre la logique *intuitionniste* et les langages de programmation *fonctionnels*, pouvait être étendu à la

logique *classique*. La logique intuitionniste est une restriction de la logique classique qui refuse le principe du tiers exclu. On a découvert que celui-ci (et d'autres équivalents) avait une interprétation calculatoire en termes de structure de contrôle, comme les exceptions ou les continuations utilisées dans les langages informatiques modernes.

Probablement l'un des problèmes les plus intéressants est celui du temps en logique. Les systèmes logiques connus aujourd'hui (logique classique, intuitionniste, linéaire...) ne permettent de représenter via l'isomorphisme de Curry-Howard, que des programmes *séquentiels*, c'est à dire des suites d'instructions s'exécutant l'une après l'autre dans un temps linéaire. Or une bonne partie de l'informatique moderne s'occupe de réseaux, c'est à dire de protocoles de communication, d'algorithmes répartis où les programmes ont tendance à se séparer en plusieurs morceaux s'exécutant indépendamment sur des machines distinctes (ou des processeurs distincts) en utilisant des procédures sophistiquées pour se synchroniser, selon un temps qui n'est plus linéaire mais est plutôt organisé comme une trame où les fils se séparent et se rejoignent.

On ne sait pas aujourd'hui si la logique est apte à prendre en compte le temps non linéaire mis en jeu dans l'algorithmique répartie, comme elle prend en compte au travers de l'isomorphisme de Curry-Howard le temps séquentiel des programmes fonctionnels. Il se peut que l'élimination des coupures soit intrinsèquement séquentielle mais cela semble douteux et la question reste ouverte.

Références

[1] Jean-Yves Girard. *Du pourquoi au comment : la théorie de la démonstration de 1950 à nos jours.* Birkhäuser, 2000.

[2] Jean-Louis Krivine. Mathématiques des programmes et programme des mathématiques. *Turbulence*, 1, 1994.

[3] Jean-Louis Krivine. Ensembles et preuves. *Quadrature*, 33, 1998.