La correspondance preuves-programmes découverte capitale … ignorée

Jean-Louis Krivine

IRIF, Université Paris-Cité

krivine@irif.fr

Paris, 19 mai 2025

Introduction

Dans les années 30 sont découverts deux objets logiques étranges :
la *logique combinatoire* par le Russe Moïse Schönfinkel
le *lambda-calcul* par l'Américain Alonzo Church.
Leur vraie nature ne sera comprise que 30 ans après :
ce sont des *langages de programmation* !
Dans les années 60, indépendamment et à dix ans d'intervalle
H. Curry et W. Howard les relient à un système rudimentaire de preuves :

la logique propositionnelle intuitionniste

dont le pouvoir expressif est très faible.

Introduction

Cette correspondance entre preuves et programmes
va être progressivement étendue
jusqu'à un système universel : *la théorie classique des ensembles*
qui permet d'exprimer toutes les preuves mathématiques.

C'est une histoire fort intéressante
mais il faudrait entrer dans des « détails techniques ».

Je veux parler plutôt des conséquences de cette découverte extraordinaire
dont le retentissement a été quasi nul.

La correspondance

Enoncé simple :

Les preuves mathématiques sont des programmes informatiques.

Pas la réciproque. Les programmes ne sont pas tous des preuves, loin de là.

Enoncé précis :

Les preuves mathématiques *formalisées*
sont des programmes informatiques *en langage machine*.

Mais je devrai expliquer les deux termes *formalisé* et *langage machine*.

L'énoncé simple nous suffira pour le moment
en particulier à propos des applications en physique théorique.

Conséquence immédiate

Voici un raisonnement trivial : un syllogisme !

Les démonstrations mathématiques sont des programmes informatiques.
Archimède écrivait des démonstrations mathématiques.
Donc Archimède écrivait des programmes informatiques.

J'en déduis qu'il disposait d'un ordinateur : son cerveau ?

Non, le cerveau d'Archimède *n'est pas* un ordinateur.
Mais il *en contient* un, entre mille autres choses.

Introspection

Ces programmes, Archimède, Euclide, Euler, Gauss, …
ne les inventaient pas. Ils les lisaient dans leur tête.

Rappelez-vous ce texte célèbre de Platon
où Socrate *rappelle* à un esclave
(quelle démocratie !)
qu'il *sait* que la diagonale du carré vaut $\sqrt{2}$.

Mais que font-ils donc là, dans notre tête, tous ces programmes ?
Et dans celle d'un zèbre ou d'une chauve-souris ?

L'évolution

Un zèbre « sait-il » donc que la diagonale du carré vaut $\sqrt{2}$?
En tous cas, il sait s'en servir et heureusement pour lui.
Car c'est essentiel pour sa survie.
Pour la même raison et comme vous et moi,
il sait se servir de la loi de la gravitation,
des équations de Maxwell et de la mécanique quantique.

Mais *exclusivement* pour son usage personnel.

A la différence des *homo sapiens* qui s'en servent *aussi* dans l'industrie.

Importance de cette découverte

Les domaines d'application de cette correspondance sont passionnants :

- les lois de la physique
- la nature des mathématiques
- le fonctionnement du cerveau des animaux évolués

Actuellement elle sert à l'écriture de programmes prouvés.
C'est bien mais … peut beaucoup mieux faire !

Intéressons-nous donc en premier lieu
aux lois de la physique.

Les lois de la physique

Les prétendues *lois mathématiques de la nature* :
gravitation, électromagnétisme, relativité, mécanique quantique, etc.
sont des *programmes* écrits par l'évolution (la Programmeuse)
dans le cerveau des animaux pour les aider à se reproduire.

La « déraisonnable efficacité » des mathématiques
c'est l'incroyable efficacité de la Programmeuse.

Or nous utilisons ces mêmes programmes
dans de tout autres conditions
à des fins d'expériences ou d'industrie.

L'Univers hors-la-loi

On a tendance à imaginer que l'Univers suit des lois,
qu'on découvre par des expériences ou des observations.

Non, l'Univers n'a pas de loi !

On a raison d'utiliser son intuition, c'est un bon outil de recherche.
Les biologistes ne se gênent pas pour parler
de l'apparition d'un organe en raison de son utilité.
C'est faux, ils le savent, mais c'est bien pratique.

Les lois de l'Univers, c'est pratique et tout aussi faux.

La Programmeuse

Les biologistes pensent sans arrêt à l'évolution,
c'est leur outil fondamental.

Pourquoi les physiciens n'y pensent-ils jamais ?

Sans parler des mathématiciens.

Ils ont tort, on va le voir.

Et les philosophes ?

Conséquences

Il en résulte trois principes *jamais observés par les physiciens* :

N°1. *Dans une expérience, les résultats sont destinés à l'observateur*.

Et à personne d'autre ! Il a le rôle central, *qu'il le veuille ou non*.

N°2. *Si l'observateur n'est pas seul, les lois peuvent être fausses.*

Evidemment, la Programmeuse ne s'occupe que d'un seul cerveau à la fois !

N°3. *Elle ne dispose pas de capacités de calcul illimitées.*

Ce sont celles du cerveau d'une chauve-souris ou d'un gnou.

On peut donc s'attendre à des blocages.

La matière noire ? L'énergie sombre ?? La Théorie de Tout ???

L'observateur en relativité

Les dimensions d'un corps, la durée d'un évènement dépendent de l'observateur.
Pas étonnant, ces données lui sont personnellement réservées.

Il n'a pas été prévu qu'une chauve-souris et un gnou
s'échangent les résultats de leurs mesures.

Les deux jumeaux n'ont plus le même age ?
Evidemment, ils n'ont qu'à pas les comparer.

Paradoxes en mécanique quantique

Ils disparaissent si on applique les principes déjà énoncés.

1. L'information qui se transmet plus vite que la lumière.

Instantanément, en fait (expérience d'Aspect).
Il faut évidemment *deux observateurs*. Cela contredit le principe N°2.

L'explication usuelle consiste à redéfinir la notion d'information
de façon que ce qui est transmis n'en soit pas une.

Ad hoc : une clé cryptographique n'est donc pas une information ?
Demandez aux pirates.
Elle est pourtant transmise instantanément.

Paradoxes en mécanique quantique

2. L'effondrement de la fonction d'onde.

Un vieux serpent de mer, qui a suscité des explications ahurissantes,
par exemple, *des mondes multiples* qui apparaissent par millions
dès qu'un physicien (ou une fourmi) fait une expérience !

Tout ça pour ne pas appliquer le principe N°1 :

Les résultats d'une expérience sont destinés exclusivement *à l'observateur.*

Effondrement

La *fonction d'onde* d'un électron donne en chaque point
la probabilité qu'il s'y trouve.

Elle suit continûment une e.d.p. : l'équation de Schrödinger.

Jusqu'au moment où on *décide*
de connaître exactement sa position.

La probabilité devient alors *brusquement* une certitude

et l'équation de Schrödinger repart tranquillement
avec cette nouvelle donnée comme condition initiale.

Effondrement (fin)

Question :

Comment ce diable d'électron s'est-il donc aperçu
que quelqu'un avait fait une mesure ?

Réponse :

L'électron est une fiction commode qui décrit
un *programme interactif* destiné à aider l'observateur
à agir au bon moment et commandé par lui.

Moralité et expérience de pensée :

Pour interpréter correctement une expérience, l'observateur
doit croire que *sa vie dépend du résultat*.

Les fentes d'Young

Chaque soi-disant « paradoxe » de la physique
est donc, en fait, un outil fort intéressant
pour étudier le fonctionnement du cerveau de l'observateur.

Un peu comme les illusions d'optique pour la vision.

Ainsi, l'expérience des *fentes d'Young* est spécialement pertinente.

Un cas frappant du *principe d'incertitude*.

C'est à son propos que R. Feynman a écrit cette bizarrerie :
Si vous croyez avoir compris la mécanique quantique,
c'est que vous ne l'avez pas comprise.

On va voir ça !

Les fentes d'Young (suite)

Le matériel :
Une source d'électrons
un écran percé de deux fentes
un écran de réception.

L'observateur se prépare à envoyer *un seul* électron.
Que prévoit alors la mécanique quantique ?
En général, seulement des probabilités, rarement un résultat certain.

Toutefois, il y a une zone de l'écran d'arrivée
qui a des propriétés étonnantes.

Le mythe d'Orphée

Si l'observateur tourne le dos pendant l'expérience,
l'électron ne frappera certainement pas la zone en question.
Mais si l'observateur regarde par quelle fente passe l'électron,
celui-ci a une probabilité positive (calculable) de la frapper
et Eurydice retourne chez Hadès.

Moralité (obstinément refusée, malgré l'évidence) :

Dans une expérience, c'est l'observateur qui a le premier rôle.

Pas étonnant, il est l'objet de toute la sollicitude de la Programmeuse.
Et bien sûr, il doit être *unique*. Sinon, l'un pourrait regarder et l'autre pas,

et l'électron ne saurait plus à quel saint se vouer !

Les fentes d'Young (suite)

Comment donc ce programme fonctionne-t-il dans un cerveau ?
Simplement en posant la question à son supérieur :

As-tu un moyen de savoir par quelle fente passera l'électron ?

et en lui fournissant la probabilité en fonction de sa seule réponse

oui ou *non*

sans se préoccuper de la nature ou même de la réalité de ce moyen.

Quel sujet de recherche passionnant
que ce dialogue entre deux parties du cerveau :
l'application qui pose la question et le système central qui répond !

Les fentes d'Young (fin)

Je ne sais pas dans quelles circonstances
a lieu ce dialogue dans le cerveau d'un zèbre ou d'une autruche.

Mais ce doit être fréquent et vital
pour que la Programmeuse s'en soit préoccupée.

Et si le système central se trompe dans sa réponse binaire ?

Alors tant pis pour le zèbre !

Il ne se trompe donc pas souvent car il y a encore des zèbres.

(Plus pour très longtemps).

Formalisation et compilation

Enoncé précis de la correspondance preuves-programmes :

Les preuves mathématiques *formalisées*
sont des programmes informatiques *en langage machine*.

Le passage d'une preuve *lisible* à une preuve *formalisée*
s'appelle *formalisation*.

Le passage d'un programme *lisible* à un programme *en langage machine*
s'appelle *compilation*.

Il y a, en effet, deux sortes bien distinctes de langages
que ce soit pour les preuves ou pour les programmes :

Les lisibles et les illisibles.

Formalisation

Qu'est-ce donc qu'une *preuve formelle* ?

Au début du 20ème siècle, les logiciens ont découvert :

- toutes les règles de la *syntaxe* du langage mathématique
- toutes les règles de *démonstration*

et elles sont étonnamment simples.

Des résultats très importants, mais pas bien accueillis à l'époque par les mathématiciens « normaux » (voir plus loin).

Syntaxe

Les symboles sont : $\bot$ *(faux)*, $\to$ *(implique)*, $\forall$ *(quel que soit)*, $\in$ *(appartient à)*,
les lettres $a, b, \ldots, x, y, z$ qu'on appelle des *variables*
sans oublier les deux parenthèses.
C'est tout !

On construit alors des phrases appelées *formules*. Exemple :

$$\forall x \forall y \big(\forall z(z \in x \to z \in y) \to \big(\forall z(z \in y \to z \in x) \to \forall z(x \in z \to y \in z)\big)\big)$$

Remarque : cette formule est appelée *axiome d'extensionnalité*.

Axiomes et règles logiques

Ils se comptent sur les doigts de la main
et sont d'une simplicité surprenante.
Exemples :

Le *modus ponens* : de A et $A \to B$ déduire B.

La *loi de Peirce* : déduire $((A \to B) \to A) \to A$
appelée aussi *tiers exclu* ou *raisonnement par l'absurde*.

L'*élimination de* $\forall$: déduire $\forall x\, A[x] \to A[y]$.

Remarque : le fameux *syllogisme* est formé
d'une élimination de $\forall$ suivie de deux modus ponens.

ZF

Pour achever ce travail formidable, il fallait décrire les objets auxquels appliquer ce langage : les ensembles.

C'est la théorie inventée par Cantor et énoncée finalement en quelques axiomes par Zermelo-Fraenkel.

J'ai écrit plus haut l'*axiome d'extensionnalité*.

Autre exemple, l'*axiome de la paire* :

$$\forall x \forall y (\forall z (x \in z \to (y \in z \to \bot)) \to \bot)$$

Les preuves formelles

On peut traduire dans ce langage n'importe quelle démonstration.
On transforme ainsi des textes lisibles (par des spécialistes, quand même)
en des textes excessivement longs et absolument illisibles.

C'est l'opération de *formalisation*.

On comprend le désintérêt et les moqueries
des mathématiciens normaux.

Mais l'avènement de l'informatique a tout changé.

Les langages informatiques

En informatique aussi, il y a deux sortes de langages :

- *Les langages évolués* comme le langage C, Java, Python, etc.

Ils sont utilisés par les *programmeurs*.

- *Les langages machine* écrits avec les seuls caractères

binaires : $0, 1$

ou, de façon équivalente, mais plus pratique

hexadécimaux : 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Ils ne peuvent être lus que par les *processeurs*.

L'ADN est aussi écrit en langage machine avec les caractères A,C,G,T.

On ne lui connaît pas de langage évolué.

Compilation

La traduction d'un programme en langage évolué vers un langage machine s'appelle une *compilation*.

C'est impossible pour un humain.

C'est réalisable sur un ordinateur au moyen d'un programme adéquat appelé *compilateur*

et écrit, bien sûr, en langage évolué.

Devinette : comment le compilateur a-t-il été compilé ?

Compilation

Voici un minuscule programme en C :

```
#include <stdio.h>
int main(){
    printf("Hello, World!\n");
}
```

et un très court extrait d'un compilé en hexa :

A8 EB 47 68 2C B0 47 50 55 9A 55 F4 76 69 A0 1A DD 8F 51 E3 D5 F9 7E CC 94 48 BF
E9 7C A2 37 C6 F2 98 D6 A8 3D 03 60 73 82 38 58 C5 50 40 D9 6F 76 F2 44 AF 85 E0
4F 71 71 E3 84 F9 59 1C 64 84 59 E4 95 62 B2 D4 B6 E4 38 73 76 23 5C 32 17 E7 66
04 16 E8 25 0E 85 4A EA E7 D4 62 28 7F 72 A9 B3 25 E5 AF 4D 9B 37 D0 98 4A 5B 91
B0 E9 17 A2 C3 4F 1F D6 CB 30 DA 42 04 41 78 8F 00 00 00 00 3A D8 29 67 00 00 00
00 02 00 00 00 1C 01 00 00 58 37 00 00 58 19 00 00 52 53 44 53 E9 E9 69 4B D9 D8
3F 49 B2 EB 14 E7 58 86 A8 3C 01 00 00 00 43 3A 5C 6A 6E 6C 70 5C 77 6F 72 6B 73
70 61 63 6D 27 46 A4 A6 1A 5D F7 54 51 6A 80 22 C6 A8 66 75 4D 54 83 C9 5E 68 1B
AB F6 5B 64 B0 AD AD 20 BD 54 EF 46 6B 6D 02 E7 45 AE 75 F6 8E 8A B9 32 2E 8B B8
77 1C 52 4D 79 88 5F A5 3A 18 C9 50 2C 97 27 2D BF DA 00 7D 63 7F EF 32 EC 46 87
63 DB 9A 9F C3 5A 21 36 7B F9 9B D5 CA A0 6E 72 2C FC E1 F1 9E 70 4D 08 56 15 6F
90 AB FB 48 BA B9 D7 C1 82 78 85 65 2E 7F B5 C9 C0 E2 14 B7 37 11 41 06 E2 8A E0
A8 F6 E5 7E B6 78 05 01 C3 60 ED 4F 07 41 2E 0A 13 49 E3 90 9B 7F DD BD FA AC BF

Décompilation

Dans l'industrie, on peut être confronté au problème inverse :
passer du langage machine au langage évolué
pour des raisons de maintenance ou de progrès technique.

Occupation essentielle aussi des méchants *hackers*.

C'est la *décompilation*. Elle est toujours très difficile
et souvent impossible.

Elle n'est pas programmable.

La correspondance

La correspondance de Curry-Howard identifie :

le *langage formel* des mathématiques	le *langage machine* de l'informatique
la *formalisation*	la *compilation*
les *preuves formalisées*	les *programmes compilés*.

On va donc aussi identifier
le langage mathématique usuel à un *langage informatique évolué*
comme C, Python ou Java.

Décompilation (suite)

Le travail du mathématicien consiste donc
à décompiler en langage mathématique usuel
les programmes en langage machine présents dans sa tête.

Un travail ardu qui dure depuis des siècles
qu'aucun autre animal n'a pu faire.

Ces programmes, installés et perfectionnés par la Programmeuse
pour permettre la survie et la reproduction d'animaux évolués
se sont révélés d'une efficacité fantastique.

Et maintenant, on pense irrésistiblement à *L'apprenti sorcier*.

Les fonctions récursives

Quelle sorte de programme obtient-on à partir d'une preuve ?
Peut-on l'exécuter et que se passe-t-il alors ?

Commençons par les mathématiques de la physique
qui sont essentiellement les e.d.p. et les probabilités.
Les programmes associés sont les calculs
effectués dans les centres de recherche et l'industrie.
Un tel programme attend des données *numériques*, tourne un certain temps
puis s'arrête en fournissant un résultat *numérique*.
Autrement dit, ces programmes calculent des *fonctions récursives*.

Les systèmes et les réseaux

Mais tous les programmes ne sont pas de ce type :
Un *système d'exploitation* est un programme
qui n'attend aucune donnée et ne doit jamais s'arrêter.
Il maintient l'ordinateur en état de marche, tout arrêt serait fatal.

De même, les programmes qui font fonctionner les réseaux
tournent sans s'arrêter dans les *routeurs*, les *serveurs*
et les *clients* (dont vous faites partie).

Quelle est alors la branche des mathématiques
dont les démonstrations donnent de tels programmes ?

La théorie des ensembles

Un sujet de recherches passionnant :

Ecrire et faire tourner les programmes associés aux théorèmes de ZF donnerait un accès au système d'exploitation de l'ordinateur inclus dans le cerveau des zèbres (et d'Archimède).

La correspondance de Curry-Howard nous fournit les outils.

Au travail !

Références

Didier Dacunha-Castelle - Les mathématiques, le cerveau de l'âne et l'évolution - Cassini (2020)

Jean-Louis Krivine - Les décompilateurs (l'Univers en tête) - Calvage & Mounet (2024)

Voir aussi :

https://www.irif.fr/~krivine