La systématicité dans les sciences formelles

Baptiste Mélès

CNRS, Archives Henri-Poincaré, Université de Lorraine

Séminaire Philmath, IHPST, 9 octobre 2023

Outline

Plan de la section

Plan de la section

Que sont les « systèmes » des sciences formelles ?

- On utilise parfois le mot informel de « système » dans les sciences formelles : « système logique », « système d'exploitation »...
  - Qu'entend-on par là ?

Polysémie du terme

- La notion de système est très polysémique :
  1. simple ensemble d'entités que l'on peut lister ;
  2. ensemble structuré par un ensemble de relations ou de fonctions (cf. la notion de « structure » en général, par exemple en mathématiques) ;
  3. doctrine philosophique complète (« le système de Kant »).

Plan de la section

Plan

1. Nous allons montrer qu'en philosophie, il existe de manière assez classique des processus de systématisation, entendue comme passage du sens 1 au sens 2.
2. Nous montrerons ensuite qu'il existe également des efforts de systématisation dans la pratique des sciences formelles :
   1. logique ;
   2. mathématiques ;
   3. informatique.

Plan de la section

Plan de la section

Aristote : une liste de catégories

- Aristote, *Catégories*, IV : classification des « énoncés sans combinaison » [Aristote(2014), 1b25-2a4] :
  1. οὐσία substance (homme, cheval) ;
  2. ποσόν quantité / combien (long de deux coudées) ;
  3. ποιόν qualité / quel (blanc, grammairien) ;
  4. πρός τι relation / par rapport à quoi (double, moitié, plus grand) ;
  5. ποῦ lieu / où (dans le Lycée, au Forum) ;
  6. ποτέ temps / quand (hier, l'an dernier) ;
  7. χεῖσθαι position (couché, assis) ;
  8. ἔχειν possession / état (chaussé, armé) ;
  9. ποιεῖν action / faire (coupe, brûle) ;
  10. πάσχειν passion / subir (coupé, brûlé).
- Aristote propose cette classification sous forme de simple liste (cf. aussi *Topiques*, I, 9, 103b21-23).

Critique par Kant de la forme « liste »

Critique dans [Kant(1944), A 81 / B 106-107] :

C'était un dessein digne d'un esprit aussi pénétrant qu'Aristote que celui de chercher ces concepts fondamentaux. Mais, comme il ne suivait aucun principe, il les recueillit avec précipitation comme ils se présentèrent à lui et en rassembla d'abord dix qu'il appela *catégories* (prédicaments). Dans la suite, il crut encore en avoir trouvé cinq autres qu'il ajouta aux premiers sous le nom de post-prédicaments. Sa table n'en resta pas moins incomplète. Du reste, on y trouve aussi quelques *modes* de la sensibilité pure (*quando, ubi, situs,* pareillement *prius, simul*) et même un mode empirique (*motus*) qui n'appartient pas du tout à ce registre généalogique de l'entendement ; on y trouve aussi des concepts dérivés mêlés à leurs concepts primitifs (*actio, passio*), et quelques-uns de ces derniers font complètement défaut.

Table kantienne des catégories

Kant propose un nouvel ensemble de catégories sous forme de table (A 80 / B 106) :

TABLE DES CATÉGORIES

1

DE LA QUANTITÉ :

Unité.
Pluralité.
Totalité.

2

DE LA QUALITÉ :

Réalité.
Négation.
Limitation.

3

DE LA RELATION :

De l'inhérence et de la subsistance (*substantia et accidens*).

De la causalité et de la dépendance (cause et effet).

De la communauté (action réciproque entre l'agent et le patient).

4

DE LA MODALITÉ :

Possibilité — Impossibilité.
Existence — Non-existence.
Nécessité — Contingence.

Systématicité des catégories kantiennes

Cette table est structurée (A 80-81 / B 106-107) :

> Cette division est tirée systématiquement d'un principe commun, à savoir, du pouvoir *de juger* (qui est la même chose que le pouvoir de penser) ; elle ne provient pas, à la façon d'une rapsodie, d'une recherche, entreprise au petit bonheur, de concepts purs, dont l'énumération ne peut jamais être certaine, puisqu'elle n'est conclue que par induction sans que jamais on pense à se demander, en agissant ainsi, pourquoi ce sont précisément ces concepts et non pas d'autres qui sont inhérents à l'entendement pur.

Fondation sur la table des jugements

La table reçoit sa structure d'une autre table (A 70 / B 95) :

nous trouvons que la fonction de la pensée dans ce jugement peut se ramener à quatre titres dont chacun se compose de trois moments. Ils peuvent être commodément représentés dans la table suivante :

1

Quantité des jugements :

Universels.
Particuliers.
Singuliers.

2

Qualité :

Affirmatifs.
Négatifs.
Indéfinis.

3

Relation :

Catégoriques.
Hypothétiques.
Disjonctifs.

4

Modalité :

Problématiques.
Assertoriques.
Apodictiques.

Fondements de la table des jugements

- L'autre table est elle-même fondée dans la structure du jugement :
  1. quantité : quantification universelle ou existentielle ;
  2. qualité : affirmation ou négation du prédicat ;
  3. relation : logique propositionnelle (p, $\rightarrow$, $\veebar$) ;
  4. modalité : rapport à la faculté de juger ($\diamond p$, p, p).
- . . . avec la tripartition propre à la logique transcendantale (exemple : un jugement singulier relève à la fois du jugement universel et du jugement particulier).
- Les entités (pertinentes) sont ordonnées selon un principe supérieur.

Plan de la section

Notion forte de système

Le système est l'unité des connaissances sous un tout (A 832-833 / B 860-861) :

CHAPITRE III

ARCHITECTONIQUE DE LA RAISON PURE

Par *architectonique* j'entends l'art des systèmes. Comme l'unité systématique est ce qui convertit la connaissance vulgaire en science, c'est-à-dire ce qui coordonne en système un simple agrégat de ces connaissances, l'architectonique est donc la théorie de ce qu'il y a de scientifique dans notre connaissance en général et elle appartient ainsi nécessairement à la méthodologie.

Sous le gouvernement de la raison, nos connaissances en général ne sauraient former une rapsodie, mais elles doivent former un système dans lequel seul elles peuvent soutenir et favoriser les fins essentielles de la raison. Or, j'entends par système l'unité de diverses connaissances sous une idée. Cette idée est le concept rationnel de la forme d'un tout, en tant que c'est en lui que sont déterminées *a priori* la sphère des éléments divers et la position respective des parties. Le concept rationnel scientifique contient, par conséquent, la fin et la forme du tout qui concorde avec elle.

Notion forte de système

Le système est complet et ne peut grandir par accumulation :

> L'unité du but auquel se rapportent toutes les parties, en même temps qu'elle se rapportent les unes aux autres dans l'idée de ce but, fait qu'aucune partie ne peut faire défaut sans qu'on en remarque l'absence, quand on connaît les autres, et qu'aucune addition accidentelle, ni aucune grandeur indéterminée de la perfection, qui n'ait pas ses limites déterminées *a priori*, ne peuvent trouver place. Le tout est donc un système organique (*articulatio*) et non un ensemble désordonné (*coacervatio*) ; il peut, à la vérité, croître par le dedans (*per intussusceptionem*), mais non par le dehors (*per appositionem*), semblable au corps de l'animal auquel la croissance n'ajoute aucun membre, mais rend, sans rien changer aux proportions, chacun des membres plus fort et plus approprié à ses fins.

Systématisation

- Kant propose ainsi une table *systématique* des concepts fondamentaux, c'est-à-dire un ensemble justifiée par des principes philosophiques qui justifient également sa topologie (ici un arbre de 4×3 branches).
- L'énumération n'est pas la bonne façon d'exposer les concepts fondamentaux de l'entendement : on ne les comprend qu'en les situant dans un ensemble structuré.
- Le même procédé de systématisation peut être observé dans l'histoire des sciences formelles.

Plan de la section

Plan de la section

Axiomes de Russell et Whitehead

Une axiomatisation du calcul propositionnel [Whitehead et Russell(1910), p. 100-101] :

1. $A \vee A \to A$
2. $B \to A \vee B$
3. $A \vee B \to B \vee A$
4. $A \vee (B \vee C) \to B \vee (A \vee C)$
5. $(B \to C) \to (A \vee B \to A \vee C)$

Axiomes de Hilbert et Ackermann

Une autre axiomatisation du calcul propositionnel [Hilbert et Ackermann(1928), p. 23] :

1. $A \vee A \to A$
2. $A \to A \vee B$
3. $A \vee B \to B \vee A$
4. $(A \to B) \to (C \vee A \to C \vee B)$

Axiomes de David, Nour et Raffalli

Une autre axiomatisation du calcul propositionnel [David et collab.(2004)David, Nour et Raffalli, p. 59] :

1. $A \to (B \to A)$
2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
3. $A \wedge B \to A$
4. $A \wedge B \to B$
5. $A \to (B \to A \wedge B)$
6. $A \to A \vee B$
7. $B \to A \vee B$
8. $(A \to C) \to ((B \to C) \to (A \vee B \to C))$
9. $(A \to \bot) \to \neg A$
10. $\neg A \to (A \to \bot)$
11. $\bot \to A$
12. $(\neg A \to \bot) \to A$

Règle de déduction

Une seule règle de déduction, le *modus ponens* :

$$\frac{A \qquad A \to B}{B}\ \text{mp}$$

Inconvénients de ces axiomatiques

- La forme de la liste ne permet pas de savoir pourquoi ce sont précisément ces axiomes qui ont été retenus, ni s'il ne serait pas pertinent d'en rajouter d'autres.
- Ces axiomatiques sont complexes à manipuler. Par exemple, la démonstration de $A \to A$ n'a rien d'intuitif.

Démonstration de $A \to A$

1. par le schéma 2, $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
   donc $(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A))$;
2. par le schéma 1, $A \to (B \to A)$
   donc $A \to ((A \to A) \to A)$;
3. par modus ponens sur 1 et 2, $(A \to (A \to A)) \to (A \to A)$:
$$\frac{(A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A)) \qquad A \to ((A \to A)}{(A \to (A \to A)) \to (A \to A)}$$
4. par le schéma 1, $A \to (B \to A)$
   donc $A \to (A \to A)$;
5. par modus ponens sur 3 et 4, $A \to A$:
$$\frac{(A \to (A \to A)) \to (A \to A) \qquad A \to (A \to A)}{A \to A}$$
   CQFD.

Plan de la section

Déduction naturelle

[Gentzen(1955)]

La différence extérieure la plus essentielle qui distingue les *NJ*-dérivations des dérivations que l'on peut effectuer dans les systèmes de Russell, Hilbert et Heyting est la suivante : dans ces systèmes, on dérive les formules vraies d'une série de « formules logiques fondamentales » au moyen d'un nombre réduit de procédés de déduction; la déduction naturelle, par contre, ne part pas, en général, de propositions logiques fondamentales, mais d'hypothèses (voir les exemples du § 1), auxquelles viennent se rattacher des déductions logiques. Grâce à une déduction ultérieure, le résultat est alors rendu à nouveau indépendant des hypothèses.

Nous désignons les calculs du premier genre sous le nom de calculs *logistiques*.

Idée de la déduction naturelle

Gentzen (1934) propose un tout autre système :

1. un seul axiome, logiquement trivial, le principe d'identité :

$$A$$

2. des règles différentes pour chaque connecteur logique (conjonction, implication, disjonction, négation, absurde) :
   - règles d'introduction : sous quelles conditions le symbole peut être employé ;
   - règles d'élimination : ce que le symbole permet de démontrer.

Conjonction

- Introduction de la conjonction

$$\frac{A \qquad B}{A \wedge B} \wedge_i$$

- Éliminations de la conjonction

$$\frac{A \wedge B}{A} \wedge_e^g$$

$$\frac{A \wedge B}{B} \wedge_e^d$$

Implication

- Introduction de l'implication

$$\frac{\begin{array}{c}[A]^n \\ \vdots \\ B\end{array}}{A \to B} \to_i^n$$

- Élimination de l'implication

$$\frac{A \qquad A \to B}{B} \to_e$$

Disjonction

- Introductions de la disjonction

$$\frac{A}{A \vee B} \vee_i^g$$

$$\frac{B}{A \vee B} \vee_i^d$$

- Élimination de la disjonction

$$\frac{A \vee B \qquad \begin{array}{c}[A]^n \\ \vdots \\ C\end{array} \qquad \begin{array}{c}[B]^n \\ \vdots \\ C\end{array}}{C} \vee_e^n$$

Négation

- Introduction de la négation

$$\dfrac{\begin{array}{c}[A]^n \\ \vdots \\ \bot\end{array}}{\neg A}\ \neg_i^n$$

- Élimination de la négation

$$\dfrac{A \qquad \neg A}{\bot}\ \neg_e$$

Absurde

- Absurde intuitionniste

$$\frac{\bot}{A}\ \bot_i$$

- Absurde classique

$$\begin{array}{c} [\neg A]^n \\ \vdots \\ \bot \end{array}$$
$$\frac{}{A}\ \bot_c^n$$

Justification des axiomes de Hilbert

- Que nous apprend la déduction naturelle sur les axiomatiques de Hilbert ?
- Elle montre la raison d'être des axiomes : introduire ou éliminer des connecteurs logiques.

Justification des axiomes de Hilbert

L'axiomatique ne devrait pas être présentée comme une simple liste :

1. $A \to (B \to A)$
2. $(A \to (B \to C)) \to ((A \to B) \to (A \to C))$
3. $A \wedge B \to A$
4. $A \wedge B \to B$
5. $A \to (B \to A \wedge B)$
6. $A \to A \vee B$
7. $B \to A \vee B$
8. $(A \to C) \to ((B \to C) \to (A \vee B \to C))$
9. $(A \to \bot) \to \neg A$
10. $\neg A \to (A \to \bot)$
11. $\bot \to A$
12. $(\neg A \to \bot) \to A$

Justification des axiomes de Hilbert

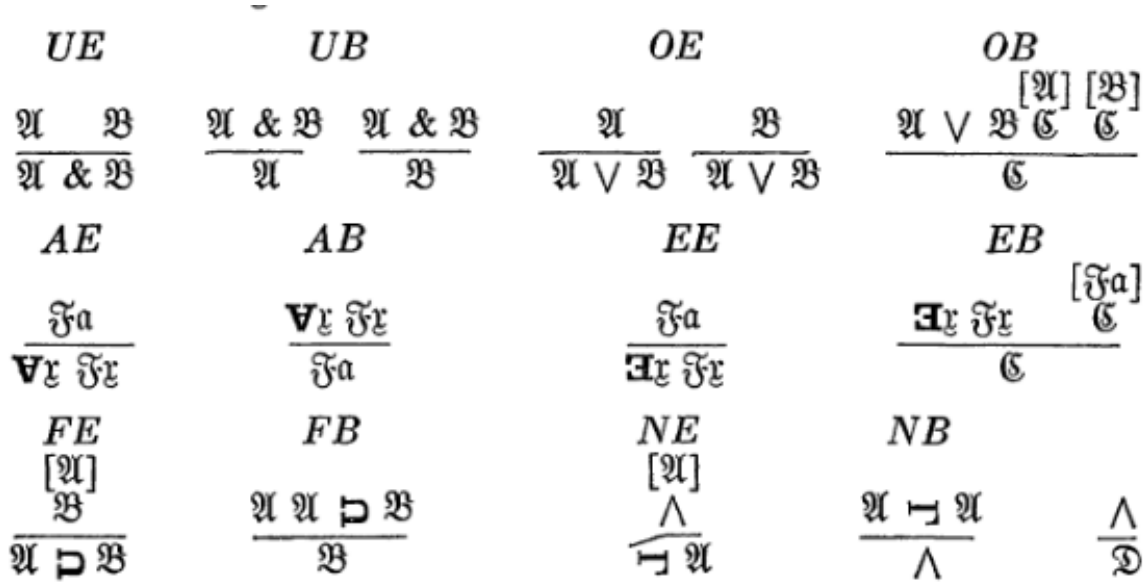
Elle devrait être, au moins en partie, présentée comme un tableau :

	Introduction	Élimination
$\wedge$	$A \to (B \to A \wedge B)$	$A \wedge B \to A$ $A \wedge B \to B$
$\vee$	$A \to A \vee B$ $B \to A \vee B$	$(A \to C) \to ((B \to C) \to (A \vee B \to C))$
...	...	...

L'axiomatique possède, au moins localement, une *structure*.
On peut justifier la présence de certains axiomes par une certaine place dans le tableau.

système de Gentzen

Chez Gentzen, la structure tabulaire est visible dans l'onomastique (*und*, *oder*, *all*, *es gibt*, *folgt*, *nicht* / *Einleitung*, *Beseitigung*) :



L'exception dans le système

L'exception de la négation dans NK :

Nous avons donné ainsi au principe du tiers-exclu, de façon purement extérieure, une place exceptionnelle, et nous l'avons fait parce que nous tenions cette formulation pour la « plus naturelle ». Il eût été parfaitement possible, au lieu d'introduire le schéma de formule fondamentale $\mathfrak{A} \vee \neg \mathfrak{A}$, d'introduire un nouveau schéma de déduction, par exemple $\dfrac{\neg\neg \mathfrak{A}}{\mathfrak{A}}$ (schéma analogue à celui qui est formé par Hilbert et par Heyting). Cependant, encore une fois, ce schéma de déduction tombe en dehors des cadres des figures de *NJ*-déduction en tant qu'il représente une nouvelle élimination de la négation, dont la validité ne découle nullement de la manière dont le signe de négation se trouve introduit dans le schéma *NE*.

Disparition de l'exception dans LK

Dans LK, on retrouve la propriété de symétrie :

> Dans le calcul classique *NK*, le principe du tiers-exclu occupait une situation exceptionnelle parmi les procédés de déduction (II, 5. 3) parce qu'il n'était pas possible de l'intégrer au système des introductions et des éliminations. Dans le calcul logistique classique *LK* que l'on va décrire dans ce qui suit cette singularité se trouve levée.

La raison d'être des axiomes du calcul propositionnel

- La déduction naturelle montre la raison d'être des axiomes du calcul propositionnel : les couples introduction/élimination.
- Ces couples introduction/élimination ne sont pas arbitraires : l'élimination annule l'introduction.
- C'est l'*élimination des coupures*.

Conjonction

$$\dfrac{\dfrac{A \qquad B}{A \wedge B}\,\wedge_i}{A}\,\wedge_e$$

$$\dfrac{\dfrac{A \qquad B}{A \wedge B}\,\wedge_i}{B}\,\wedge_e$$

Implication

$$\dfrac{A \qquad \dfrac{\begin{array}{c}[A]^n \\ \vdots \\ B\end{array}}{A \to B} \to^n_i}{B} \to_e$$

Disjonction

$$\frac{\dfrac{A}{A \vee B}\ \vee_i^g \qquad [A]^n \qquad A}{A}\ \vee_e^n$$

$$\frac{\dfrac{B}{A \vee B}\ \vee_i^d \qquad [A]^n \qquad A}{A}\ \vee_e^n$$

Remarque : rien ne nous oblige à utiliser l'hypothèse $[B]$. C'est ainsi que la déduction naturelle simule l'axiome 1 :

$$A \to (B \to A)$$

Négation

$$\dfrac{\dfrac{\dfrac{[A]^n}{\vdots}}{\dfrac{\bot}{\neg A}\,\neg_i^n} \qquad A}{\bot}\,\neg_e$$

Peut-on « éliminer » les règles d'élimination ?

- Puisque les règles d'élimination peuvent se « déduire » (ou plutôt s'induire) naturellement des règles d'introduction, pourquoi les formuler explicitement ?
- Il pourrait suffire de formuler la première colonne (règles d'introduction) puis de stipuler l'éliminabilité des coupures ; on en infère ainsi les règles d'élimination.

On peut « éliminer » les règles d'élimination

- C'est ce qui se passe dans le système Coq : les règles d'élimination sont construites automatiquement après déclaration des règles d'introduction.
- Coq est :
  - *logiquement* un assistant de démonstration ayant pour base le Calcul des Constructions Inductives ;
  - *informatiquement* un langage de programmation très puissant (avec types simples, polymorphisme, ordres supérieurs, types dépendants) et non complet.

Conjonction

Règle d'introduction déclarée à Coq

```
Inductive et (A: Prop) (B: Prop) : Prop :=
  conj: A -> B -> et A B.
```

... ce qui revient à $\dfrac{A \qquad B}{P}$ et

Règle d'élimination automatiquement inférée

```
Print et_ind.
```

forall A B P : Prop, (A -> B -> P) -> et A B -> P

... ce qui revient à $\dfrac{A \wedge B \to P \qquad A \wedge B}{P}$ et_ind

Disjonction

Règle d'introduction déclarée à Coq

```
Inductive ou (A: Prop) (B: Prop) : Prop :=
| gauche: A -> ou A B
| droite: B -> ou A B.
```

... ce qui revient à $\dfrac{A}{A \vee B}$ gauche + $\dfrac{B}{A \vee B}$ droite

Règle d'élimination automatiquement inférée par Coq

```
Print ou_ind.
```

forall A B P : Prop, (A -> P) -> (B -> P) -> ou A B -> P

... ce qui revient à $\dfrac{A \vee B \qquad A \to P \qquad B \to P}{P}$ ou_ind

Absurde

Règle d'introduction déclarée à Coq

```
Inductive faux : Prop :=
.
```

Règle d'élimination automatiquement inférée

```
Print faux_ind.
```

forall P : Prop, faux -> P

... ce qui revient à $\dfrac{\bot}{A}\ \bot_i$

C'est l'absurde intuitionniste !

Plan de la section

Prolongement en mathématiques

- Le même principe de construction produit des effets jusqu'en mathématiques, dans la présentation de l'axiomatique de l'arithmétique.

L'axiomatique de Peano (présentation historique)

[Peano(1889), p. 1] (Permuter « *a*,*b*,*c* » et « *a*,*b* » en 3 et 4.)

Signo N significatur *numerus (integer positivus).*
» 1 » *unitas.*
» $a+1$ » *sequens a,* sive *a plus* 1.
» $=$ » *est aequalis.* Hoc ut novum signum considerandum est, etsi logicae signi figuram habeat.

Axiomata.

1. $1 \in \mathrm{N}.$
2. $a \in \mathrm{N} . \supset . a = a.$
3. $a, b, c \in \mathrm{N} . \supset : a = b . = . b = a.$
4. $a, b \in \mathrm{N} . \supset \therefore a = b . b = c : \supset . a = c.$
5. $a = b . b \in \mathrm{N} : \supset . a \in \mathrm{N}.$
6. $a \in \mathrm{N} . \supset . a + 1 \in \mathrm{N}.$
7. $a, b \in \mathrm{N} . \supset : a = b . = . a + 1 = b + 1.$
8. $x \in \mathrm{N} . \supset . a + 1 - = 1.$
9. $k \in \mathrm{K} \therefore 1 \in k \therefore x \in \mathrm{N} . x \in k : \supset_x . x + 1 \in k :: \supset . \mathrm{N} \supset k.$

L'axiomatique de Peano (présentation courante)

Langage $(\mathbb{N}, 0, S)$

1. $0 \in \mathbb{N}$
2. $\forall n \in \mathbb{N}, S(n) \in \mathbb{N}$
3. $\forall nm \in \mathbb{N}, S(n) = S(m) \to n = m$
4. $\neg \exists n, Sn = 0$
5. $\forall P, P(0) \wedge (\forall n \in \mathbb{N}, P(n) \to P(S(n))) \to (\forall n \in \mathbb{N}, P(n))$

Introduction des naturels : les axiomes 1 et 2 de Peano

Règle d'introduction déclarée à Coq dans `Datatypes.v`

```
Inductive nat : Set :=
| O : nat
| S : nat -> nat.
```

... ce qui revient à $\dfrac{}{0 \in N}\,\mathrm{O} + \dfrac{n \in N}{Sn \in N}\,\mathrm{S}$

On reconnaît ici les deux premiers axiomes de Peano :

1. $0 \in \mathbb{N}$;
2. $\forall n \in \mathbb{N}, S(n) \in \mathbb{N}$.

Élimination des entiers naturels : l'axiome 5 de Peano

Règle d'élimination automatiquement inférée

```
Print nat_ind.
```

forall P : nat -> Prop, P O -> (forall n : nat, P n -> P (S n)) -> forall n : nat, P n

... ce qui revient à $\dfrac{P(0) \qquad \forall n(P(n) \to P(Sn))}{\forall n P n}$,

autrement dit le principe de récurrence, l'axiome 5 de Peano ! Il n'a rien de gratuit : il est induit automatiquement par le système logique à partir du principe d'élimination des coupures.

La théorie Peano

- Qu'en est-il des axiomes 3 et 4 ? Cherchons dans la bibliothèque arithmétique de Coq.
- En vertu de la *correspondance preuves-programmes*, ce que les mathématiciens appellent « théorie » est exactement ce que les informaticiens appellent « module ».
- Charger en Coq le module Init/Peano.v  équivaut donc à se donner la théorie PA.
- C'est une base de données qui contient :
  1. des définitions (être le successeur, etc.) ;
  2. des lemmes et théorèmes (avec leur démonstration).

L'axiome 3 de Peano

Dans ce module, on trouve le théorème not_eq_S  :

```
Theorem not_eq_S : forall n m:nat, n <> m -> S n <> S m.
```

C'est l'axiome 3 de Peano — mais ici c'est un théorème !
Démonstration par injection : la définition de `nat` étant close (un présupposé de Coq), la seule façon de construire `S n` est de rajouter `S` devant un `n`, ce qui exclut tout `m` distinct de `n`.

C'est un théorème parce que le système suppose la clôture du type par les constructeurs : le type `nat` est le point fixe de la définition inductive, c'est-à-dire la partie minimale commune à tous les systèmes satisfaisant ces règles de construction.

L'axiome 4 de Peano

L'axiome 4 devient quant à lui le théorème O_S :

```
Theorem O_S : forall n:nat, 0 <> S n.
```

La démonstration de cet « axiome » procède par propriété discriminante : seul le constructeur `O` permet de construire 0, alors que le constructeur `S` ne permet de construire que des termes de la forme `S n`.

Justification des axiomes de l'arithmétique

- La théorie des types inductifs permet de montrer les rôles respectifs des axiomes de Peano :
  1. $0 \in \mathbb{N}$
  2. $\forall n \in \mathbb{N}, S(n) \in \mathbb{N}$
  3. $\forall nm \in \mathbb{N}, S(n) = S(m) \to n = m$
  4. $\neg \exists n, Sn = 0$
  5. $\forall P, P(0) \wedge (\forall n \in \mathbb{N}, P(n) \to P(S(n))) \to (\forall n \in \mathbb{N}, P(n))$
- On comprend désormais que :
  - les axiomes 1 et 2 sont les axiomes propres de la théorie ;
  - les axiomes 3 et 4 ne sont que des axiomes de clôture, condensés par la définition inductive par point fixe ;
  - l'axiome 5 est la règle d'élimination qui suit naturellement des axiomes 1 et 2.

Quel apport scientifique ?

- Techniquement, systématiser une liste d'axiomes n'apporte rien : on ne gagne aucun théorème que l'on ne pouvait pas démontrer avec les outils précédents.
- Mais le système gagne en intelligibilité une fois que nous comprenons quels sont les principes de construction sous-jacents à la liste d'axiomes, et le nouveau système peut faciliter la démonstration de théorèmes complexes.

Plan de la section

Plan de la section

Système des fonctions d'une machine universelle de Turing

- [Turing(1936), p. 235] : les « tables abrégées » comme langage fonctionnel de manipulation de machines de Turing.
- C'est dans ce langage de haut niveau que Turing décrit sa machine universelle.
- L'ensemble des fonctions de ce langage de haut niveau forme un système, visible dans l'onomastique.

Tables complètes de Turing

- Les premières machines décrites par Turing le sont de manière explicite (p. 233) :

Configuration		*Behaviour*		
m-config.	*symbol*	*operations*	*final*	*m-config.*
b	None	P0, R		c
c	None	R		c
c	None	P1, R		e
e	None	R		b

Tables abrégées de Turing

- Turing définit ensuite certaines fonctions de « haut niveau », qui permettent d'abréger les tables (p. 236) :

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$	$\mathfrak{d}$	L	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$
	not $\mathfrak{d}$	L	$\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$
$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$	$\mathfrak{a}$		$\mathfrak{C}$
	not $\mathfrak{a}$	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$
	None	R	$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$
$\mathfrak{f}_2(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$	$\mathfrak{a}$		$\mathfrak{C}$
	not $\mathfrak{a}$	R	$\mathfrak{f}_1(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$
	None	R	$\mathfrak{B}$

From the *m*-configuration $\mathfrak{f}(\mathfrak{C}, \mathfrak{B}, \mathfrak{a})$ the machine finds the symbol of form $\mathfrak{a}$ which is farthest to the left (the "first $\mathfrak{a}$") and the *m*-configuration then becomes $\mathfrak{C}$. If there is no $\mathfrak{a}$ then the *m*-configuration becomes $\mathfrak{B}$.

Onomastique de Turing

- Turing nomme ses fonctions de manière mnémonique (pour des anglophones) :

f	find / first
l	left
r	right
q	queue
e	erase
pe	print at the end
c	copy
re	replace
	etc.

Onomastique de Turing

- En outre, les noms de fonctions sont construits de façon très systématique :
  - q, l, e, pe… sont des fonctions globales ;
  - q_1, q_2, e_1, pe_1… sont des fonctions locales, avec numérotation des lignes.
- Le nom même des fonctions donne donc des informations sur elles.
- Les fonctions obéissent à plusieurs étages successifs d'abstraction.

Fonctions de bas niveau

Les fonctions de bas niveau manipulent directement la bande :

m-config.	symbol	operations	final *m*-config.
f(E, B, α)	ə	L	f_1(E, B, α)
	not ə	L	f(E, B, α)
f_1(E, B, α)	α	R	E
	not α	R	f_1(E, B, α)
	None	R	f_2(E, B, α)
f_2(E, B, α)	α		E
	not α	R	f_1(E, B, α)
	None	R	B
q(E)	Any	R	q(E)
	None	R	q_1(E)
q_1(E)	Any	R	q(E)
	None		E

Fonctions de niveau intermédiaire

- Les fonctions de niveau intermédiaire ne manipulent la bande que dans leur partie privée (q_1, e_1, ...) :

m-config.	symbol	operations	final *m*-config.
q(E,α)			q(q$_1$(E,α))
q$_1$(E,α)	α		E
	Not α	L	q$_1$(E,α)
e(E,B,α)			f(e$_1$(E,B,α),B,α)
e$_1$(E,B,α)		E	E
pe(E,β)			f(pe$_1$(E,β),E, ə)
pe$_1$(E,β)	Any	R,R	pe$_1$(E,β)
	None	Pβ	E
c(E,B,α)			f'(c$_1$(E),B,α)
c$_1$(E)	β		pe(E,β)

Fonctions de haut niveau

Les fonctions de haut niveau ne manipulent pas directement la bande. Elles généralisent les opérations d'écriture, déplacement, etc. en les exprimant de façon purement fonctionnelle, sans effet de bord [Mélès(2020)].

m-config.	symbol	operations	final *m*-config.
f'(E,B,α)			f(l(E),B,α)
f''(E,B,α)			f(r(E),B,α)
ce(E,B,α)			c(e(E,B,α),B,α)
ce(B,α)			ce(ce(B,α),B,α)
re(B,α,β)			re(re(B,α,β),B,α,β)
cr(E,B,α)			c(re(E,B,α,a),B,α)
cr(B,α)			cr(cr(B,α), re(B,a,α),α)
cpe(E$_{1}$,U,E,α{},β)			cp(e(e(E$_{1}$,E,β{}),E,α),U,E,α,β

Fonctions homonymes

- Certaines fonctions sont homonymes (à signature près) et obéissent à une construction systématique :

Opération	Application unique	Application itérée
effacement de symbole	$e(E,B,\alpha)$	$e(B,\alpha)$
déplacement de symbole	$ce(E,B,\alpha)$	$ce(B,\alpha)$
remplacement de symbole	$re(E,B,\alpha,\beta)$	$re(B,\alpha,\beta)$
copie de symbole	$cr(E,B,\alpha)$	$cr(B,\alpha)$
comparaison de symboles	$cpe(E_1,U,E,\alpha,\beta)$	$cpe(U,E,\alpha,\beta)$

L'onomastique recèle donc une conception systématique de la liste de fonctions correspondantes.

Plan de la section

Onomastique dans le code source de Linux

- La liste des fonctions de conversion des formats temporels dans le fichier `time.c` du noyau Linux obéit à une construction systématique (la combinatoire des conversions), que manifeste la construction des noms de fonctions : 
  - `jiffies_to_msecs`
  - `jiffies_to_usecs`
  - `timespec64_to_jiffies`
  - `jiffies_to_timespec64`
  - `jiffies_to_clock_t`
  - `clock_t_to_jiffies`
  - `...`

Plan de la section

Les définitions redondantes en programmation orientée objet

- Le processus de systématisation explique aussi la construction de définitions redondantes dans les bibliothèques standard de langages de programmation.
- Exemple : la bibliothèque arithmétique standard du langage Java Integer.java  est une litanie de tautologies.
- L'objet Integer  est défini comme... un `int` ! (dans le langage de base de Java)
- De nombreuses autres fonctions  relatives aux entiers sont redéfinies de façon parfaitement redondante avec le langage de base.
- Techniquement, ce fichier n'a donc aucun intérêt : il n'introduit que des notations alternatives pour des notations préexistantes.

Quelle utilité ?

- Le seul gain est informel, en expressivité linguistique : Integer.java, Float.java et Long.java introduisent des noms de fonctions manifestant un système combinatoire sous-jacent :

```
Integer.toString()
Integer.toHexString()
Long.toString()
Long.toHexString()
Float.toString()
Float.toHexString()
```

- Cette bibliothèque définit ainsi simplement un langage systématique par-dessus le langage de base, qui permette de penser dans un nouveau cadre.

Plan de la section

Quelques autres listes

En programmation, on trouve souvent des listes de fonctions :

- les appels système sous Unix  ;
- les fonctions d'une bibliothèque standard (ex. : la bibliothèque d'entrées sorties du langage C, stdio.h  ;
- etc.

Ces enrichissements du langage constituent souvent un cadre de haut niveau, défini dans le langage de base et destiné à se substituer à lui. Il est parfois strictement moins puissant que le cadre de base (ex. : les appels système).
C'est un terrain privilégié pour essayer de déterminer le système sous-jacent.

Plan de la section

Plan de la section

Absence de gain technique

- Une partie de l'activité dans les sciences formelles consiste non pas à construire de nouveaux objets, mais à se contenter de mettre un ordre dans ceux que l'on connaissait déjà :
  1. la déduction naturelle ne permet de rien démontrer de plus que les axiomatiques à la Hilbert ;
  2. la bibliothèque arithmétique de Coq ne peut rien démontrer de plus que la présentation habituelle des axiomes de Peano ;
  3. le langage de haut niveau de Turing ne permet de rien calculer de plus que le langage de base ;
  4. les bibliothèques standard de langages de programmation n'étendent en rien le domaine calculatoire (souvent déjà complet) ;
  5. les appels système d'un système d'exploitation permettent d'effectuer *strictement moins* de choses que l'utilisation directe de la machine (puisque le système d'exploitation nous interdit de faire certaines bêtises).

Un gain informel

- S'il y a un gain dans ces langages systématiques de niveau supérieur, c'est donc sur un autre axe que celui de la technique ; par exemple un gain en intelligibilité ou en praticité d'utilisation.

Bibliographie I

- «Software Heritage», https ://www.softwareheritage.org/ ?lang=fr.
- Aristote. 2014, *Œuvres complètes*, Flammarion, Paris, France, ISBN 978-2-08-127316-0.
- David, R., K. Nour et C. Raffalli. 2004, *Introduction à la logique : théorie de la démonstration*, Dunod, Paris, France, ISBN 978-2-10-006796-1.
- Di Cosmo, R. et S. Zacchiroli. 2017, «Software heritage : Why and how to preserve software source code», dans *iPRES 2017-14th International Conference on Digital Preservation*, p. 1–10.

Bibliographie II

- Gentzen, G. 1955, *Recherches sur la déduction logique*, Presses universitaires de France, Paris, France.
- Hilbert, D. et W. Ackermann. 1928, *Grundzüge der theoretischen Logik*, Verlag von Julius Springer, Berlin, Allemagne.
- Kant, I. 1944, *Critique de la raison pure*, PUF, Paris, France, ISBN 978-2-13-060871-4.
- Mélès, B. 2020, «Les langages de Turing», *Intellectica*, vol. 72, n° 1, p. 81–110.
- Peano, G. 1889, *Arithmetices principia : nova methodo exposita*, Fratres Bocca, Augustae Taurinonum, Italie.

Bibliographie III

 Turing, A. M. 1936, «On computable numbers, with an application to the Entscheidungsproblem», *Proceedings of the London Mathematical Society*, vol. 42, n° 2, p. 230–265.

 Whitehead, A. N. et B. Russell. 1910, *Principia Mathematica*, Cambridge University Press, Cambridge (UK).