Programmation Fonctionnelle Avancée

Séance 9 : Preuves et types, Curry-Howard

Alexandros Singh

Université Paris 8

3 décembre 2023

Les formules de la logique propositionnelle classique sont formées par la grammaire suivante :

$$\Phi ::= \bot \mid V \mid (\Phi \to \Phi) \mid (\Phi \vee \Phi) \mid (\Phi \wedge \Phi)$$

où V représente un ensemble infini de *variables propositionnelles* et le symbole $\bot$ représente la formule "toujours fausse".
Cela définit la syntaxe mais ne nous dit pas quelles formules sont vraies et sous quelles hypothèses !

La sémantique

Pour donner un sens à ces formules, on peut commencer par donner à chaque variable une valeur : vrai ou faux, et donne à $\perp$ la valeur false.
Les formules construites à l'aide des connecteurs $(\rightarrow, \wedge, \vee)$ reçoivent alors une valeur qui est fonction des valeurs de leurs composants :

A	B	$A \wedge B$
f	f	f
f	v	f
v	f	f
v	v	v

A	B	$A \vee B$
f	f	f
f	v	v
v	f	v
v	v	v

A	B	$A \rightarrow B$
f	f	v
f	v	v
v	f	f
v	v	v

D'autres opérations peuvent être définies sur la base de ce qui précède :

- $\neg A := A \rightarrow \perp$
- $A \leftrightarrow B := (A \rightarrow B) \wedge (B \rightarrow A)$
- etc ...

Tautologies classiques

Il existe des formules (appelées *tautologies*) qui sont vraies quelle que soit la valeur de leurs variables !

Une tautologie particulièrement simple mais très importante est le *tertium non datur* ou *principe du tiers exclu* :

A	$(A \vee \neg A)$
f	v
v	v

Voici d'autres tautologies :

- $(A \to B) \leftrightarrow (\neg B \to \neg A)$ (contraposition)
- $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$ (Loi de Morgan)
- $\ldots$

Pour prouver qu'une formule est une tautologie, il suffit de vérifier sa table de vérité !

Mais existe-t-il un moyen "purement syntaxique" de prouver les tautologies ?

La réponse est oui !
Il existe même plusieurs systèmes formels permettant de le faire. Nous nous concentrons ici sur un (sous-système) de la déduction naturelle de Gentzen :

- Les objets qu'il manipule sont les "séquents" $\Gamma \vdash A$: paires constituées d'une collection de formules Γ, le *contexte*, et d'une formule A, la *conclusion*.
- La manipulation est effectuée par des règles de la forme :
$$\frac{\Gamma_1 \vdash A_1 \ldots \Gamma_k \vdash A_k}{\Delta \vdash B}$$
que nous pouvons lire comme suit :

$$\frac{\text{Si nous pouvons déduire } A_1 \text{ de l'hypothèse } \Gamma_1,\ \ldots,\ A_k \text{ de l'hypothèse } \Gamma_k,}{\text{alors nous pouvons déduire } B \text{ de } \Delta.}$$

Déduction naturelle

$$\frac{}{\Gamma, \phi \vdash \phi}\ Ax$$

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\ \wedge I \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\ \wedge E_1 \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\ \wedge E_2$$

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi}\ \vee I_1 \qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash \psi \vee \phi}\ \vee I_2 \qquad \frac{\Gamma, \phi \vdash \rho \quad \Gamma, \psi \vdash \rho \quad \Gamma \vdash \phi \vee \psi}{\Gamma \vdash \rho}\ \vee E$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \psi \to \phi} \to I \qquad \frac{\Gamma \vdash \phi \to \psi \quad \Gamma \vdash \phi}{\Gamma \vdash \psi} \to E$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash \phi}\ \bot E \qquad \frac{}{\Gamma \vdash \phi \vee \neg\phi}\ PTE$$

Une preuve est alors une composition de ces règles :

$$\dfrac{\dfrac{}{\phi \wedge \psi \vdash \phi \wedge \psi}\,Ax}{\phi \wedge \psi \vdash \psi}\,\wedge E_2$$

Une preuve est alors une composition de ces règles :

$$\dfrac{\dfrac{\dfrac{}{\phi \land \psi \vdash \phi \land \psi}\,Ax}{\phi \land \psi \vdash \psi}\,\land E_2 \qquad \dfrac{\dfrac{}{\phi \land \psi \vdash \phi \land \psi}\,Ax}{\phi \land \psi \vdash \phi}\,\land E_1}{\phi \land \psi \vdash \psi \land \phi}\,\land I$$

Une preuve est alors une composition de ces règles :

$$\dfrac{\dfrac{\dfrac{}{\phi \land \psi \vdash \phi \land \psi}\,Ax}{\phi \land \psi \vdash \psi}\,\land E_2 \qquad \dfrac{\dfrac{}{\phi \land \psi \vdash \phi \land \psi}\,Ax}{\phi \land \psi \vdash \phi}\,\land E_1}{\dfrac{\phi \land \psi \vdash \psi \land \phi}{\vdash \phi \land \psi \to \psi \land \phi}\to I}\,\land I$$

Une preuve est alors une composition de ces règles :

$$
\dfrac{\dfrac{\dfrac{}{\phi \land \psi \vdash \phi \land \psi}\,Ax}{\phi \land \psi \vdash \psi}\,\land E_2 \qquad \dfrac{\dfrac{}{\phi \land \psi \vdash \phi \land \psi}\,Ax}{\phi \land \psi \vdash \phi}\,\land E_1}{\dfrac{\phi \land \psi \vdash \psi \land \phi}{\vdash \phi \land \psi \to \psi \land \phi}\to I}\,\land I
$$

Nous avons prouvé que $\phi \land \psi \to \psi \land \phi$ est toujours vrai, quelles que soient les valeurs de ϕ et ψ ! C'est une tautologie !

Une preuve est alors une composition de ces règles :

$$\dfrac{\dfrac{\dfrac{}{\phi \wedge \psi \vdash \phi \wedge \psi}\,Ax}{\phi \wedge \psi \vdash \psi}\,\wedge E_2 \qquad \dfrac{\dfrac{}{\phi \wedge \psi \vdash \phi \wedge \psi}\,Ax}{\phi \wedge \psi \vdash \phi}\,\wedge E_1}{\dfrac{\phi \wedge \psi \vdash \psi \wedge \phi}{\vdash \phi \wedge \psi \to \psi \wedge \phi}\,\to I}\,\wedge I$$

Nous avons prouvé que $\phi \wedge \psi \to \psi \wedge \phi$ est toujours vrai, quelles que soient les valeurs de ϕ et ψ ! C'est une tautologie !

Théorème (Emil Post et autres)

Une formule est une tautologie dans la logique propositionnelle classique si et seulement si elle a une dérivation commençant et terminant avec le contexte vide.

Théorème

Il existe deux nombres irrationnels x et y pour lesquels x^y est rationnel.

Preuve :

- Rappelons que $\sqrt{2}$ est irrationnel.
- Considérons le nombre $\sqrt{2}^{\sqrt{2}}$: il est **soit rationnel, soit irrationnel** (principe du tiers exclu).
- S'il est rationnel, prenons $x = y = \sqrt{2}$.
- Sinon, on prend $x = \sqrt{2}^{\sqrt{2}}$ et $y = \sqrt{2}$. Alors $x^y = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^2 = 2$, ce qui est rationnel.

“Ce ne sont pas des mathématiques , c'est de la théologie !”
Paul Gordan, soi-disant en réponse au travail de Hilbert

La preuve n'est pas *constructive* :

- Nous savons que l'une des deux affectations possibles de x et y vérifie le théorème.
- Cependant, la preuve ne nous dit pas laquelle !
- En tant qu'informaticiens, nous devrions être inquiets : nous aimons avoir des valeurs concrètes que nous pouvons calculer !
- Nous devons nous détourner de la logique classique pour nous tourner vers autre chose...

Diverses controverses dans les mathématiques du XIXe siècle ont conduit à l'essor de la logique intuitionniste, dont les principes sont généralement attribués à Brouwer (XXe siècle).
Dans la logique intuitionniste, les assertions ne sont plus des questions de “vérité” mais de “construction” (interprétation de Brouwer-Heyting-Kolmogorov) :

- Une construction de $A \wedge B$ est une construction de A et une construction de B.
- Une construction de $A \vee B$ est une construction de A ou de B.
- Une construction de $P \to Q$ est une **fonction** qui transforme une construction de P en une construction de Q
- Il n'y a pas de construction pour $\perp$.

La négation devient alors beaucoup plus intéressante :

- $\neg A := A \to \perp$: Une construction de $\neg A$ est une fonction qui transforme toute construction de A en un objet inexistant !

Le principe du tiers exclu n'est plus une tautologie selon cette interprétation !

Mais qu’est ce qu’une fonction ?

Church, dans les années 1930 et dans le cadre de ses recherches sur les fondements des mathématiques, a introduit un système formel dont les termes sont :

- *Variables* $x, y, \dots$

Church, dans les années 1930 et dans le cadre de ses recherches sur les fondements des mathématiques, a introduit un système formel dont les termes sont :

- *Variables* $x, y, \ldots$
- *Abstractions* $\lambda x.t$, où x est une variable et t un terme.

Church, dans les années 1930 et dans le cadre de ses recherches sur les fondements des mathématiques, a introduit un système formel dont les termes sont :

- *Variables* $x, y, \ldots$
- *Abstractions* $\lambda x.t$, où x est une variable et t un terme.
- *Applications* $(a\ b)$ où a et b sont des termes.

Church, dans les années 1930 et dans le cadre de ses recherches sur les fondements des mathématiques, a introduit un système formel dont les termes sont :

- *Variables* $x, y, \ldots$
- *Abstractions* $\lambda x.t$, où x est une variable et t un terme.
- *Applications* $(a\ b)$ où a et b sont des termes.

Intuitivement :

- $\lambda x.t$ représente une **fonction** qui prend en entrée x et retourne t.
- Une application $(a\ b)$ consiste alors à fournir un argument b (une autre fonction ou une variable) à une fonction a.

Il n'y a donc que des fonctions et des variables... Qu'est-ce qu'on peut faire avec ça ?

Church, dans les années 1930 et dans le cadre de ses recherches sur les fondements des mathématiques, a introduit un système formel dont les termes sont :

- *Variables* $x, y, \ldots$
- *Abstractions* $\lambda x.t$, où x est une variable et t un terme.
- *Applications* $(a\ b)$ où a et b sont des termes.

Intuitivement :

- $\lambda x.t$ représente une **fonction** qui prend en entrée x et retourne t.
- Une application $(a\ b)$ consiste alors à fournir un argument b (une autre fonction ou une variable) à une fonction a.

Il n'y a donc que des fonctions et des variables... Qu'est-ce qu'on peut faire avec ça ?

Tout ce qui est calculable !

(Gödel, Church, Turing, Kleene, , . . .)

- Que des fonctions ? Cela vous rappelle quelque chose ?

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !
- Comment calculons-nous des choses ? En utilisant la règle de la β-réduction :

$$(\lambda x.t \ y) \xrightarrow{\beta} t[x := y]$$

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !
- Comment calculons-nous des choses ? En utilisant la règle de la β-réduction :

$$(\lambda x.t\ y) \xrightarrow{\beta} t[x := y]$$

- Intuitivement : pour appliquer une fonction $f(x)$ à une variable y, il faut remplacer x par y dans la définition de f.

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !
- Comment calculons-nous des choses ? En utilisant la règle de la β-réduction :

$$(\lambda x.t\ y) \xrightarrow{\beta} t[x := y]$$

- Intuitivement : pour appliquer une fonction $f(x)$ à une variable y, il faut remplacer x par y dans la définition de f.
- Par exemple :

$$(\lambda x.x\ y) \xrightarrow{\beta} y$$

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !
- Comment calculons-nous des choses ? En utilisant la règle de la β-réduction :

$$(\lambda x.t\ y) \xrightarrow{\beta} t[x := y]$$

- Intuitivement : pour appliquer une fonction $f(x)$ à une variable y, il faut remplacer x par y dans la définition de f.
- Par exemple :

$$(\lambda x.x\ y) \xrightarrow{\beta} y$$

$$(\lambda x.(x\ x)\ y) \xrightarrow{\beta} (y\ y)$$

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !
- Comment calculons-nous des choses ? En utilisant la règle de la β-réduction :

$$(\lambda x.t\ y) \xrightarrow{\beta} t[x := y]$$

- Intuitivement : pour appliquer une fonction $f(x)$ à une variable y, il faut remplacer x par y dans la définition de f.
- Par exemple :

$$(\lambda x.x\ y) \xrightarrow{\beta} y$$

$$(\lambda x.(x\ x)\ y) \xrightarrow{\beta} (y\ y)$$

$$(\lambda x.(x\ x)\ \lambda x.(x\ x)) \xrightarrow{\beta} ?$$

- Que des fonctions ? Cela vous rappelle quelque chose ?
- C'est la base de la programmation fonctionnelle !
- Comment calculons-nous des choses ? En utilisant la règle de la β-réduction :

$$(\lambda x.t\ y) \xrightarrow{\beta} t[x := y]$$

- Intuitivement : pour appliquer une fonction $f(x)$ à une variable y, il faut remplacer x par y dans la définition de f.
- Par exemple :

$$(\lambda x.x\ y) \xrightarrow{\beta} y$$

$$(\lambda x.(x\ x)\ y) \xrightarrow{\beta} (y\ y)$$

$$(\lambda x.(x\ x)\ \lambda x.(x\ x)) \xrightarrow{\beta} (\lambda x.(x\ x)\ \lambda x.(x\ x)) \xrightarrow{\beta} \ldots$$

Dans le cadre de son approche des fondements des mathématiques, Church a également introduit une variante typée de ce système (pour éviter certains paradoxes) :

- Les termes sont maintenant annotés avec des types.

Dans le cadre de son approche des fondements des mathématiques, Church a également introduit une variante typée de ce système (pour éviter certains paradoxes) :

- Les termes sont maintenant annotés avec des types.
- Les types sont définis par la grammaire :

$$t, t' := T \mid t \to t'$$

  où est une collection de types de base et $t \to t$ représente intuitivement les fonctions qui prennent un argument de type t et renvoient un argument de type t.

Dans le cadre de son approche des fondements des mathématiques, Church a également introduit une variante typée de ce système (pour éviter certains paradoxes) :

- Les termes sont maintenant annotés avec des types.
- Les types sont définis par la grammaire :

$$t, t' := T \mid t \to t'$$

  où est une collection de types de base et $t \to t$ représente intuitivement les fonctions qui prennent un argument de type t et renvoient un argument de type t.
- Les termes de ce nouveau calcul ne sont valides que si on peut leur attribuer un type. Par exemple, si $x : A$ alors $\lambda x.x : A \to A$. Si, de plus, $y : A$, alors $(\lambda x.x\ y) : A$.

Dans le cadre de son approche des fondements des mathématiques, Church a également introduit une variante typée de ce système (pour éviter certains paradoxes) :

- Les termes sont maintenant annotés avec des types.
- Les types sont définis par la grammaire :

$$t, t' := T \mid t \to t'$$

  où est une collection de types de base et $t \to t$ représente intuitivement les fonctions qui prennent un argument de type t et renvoient un argument de type t.
- Les termes de ce nouveau calcul ne sont valides que si on peut leur attribuer un type. Par exemple, si $x : A$ alors $\lambda x.x : A \to A$. Si, de plus, $y : A$, alors $(\lambda x.x\ y) : A$.
- Essayez de trouver un type pour $(\lambda x.(x\ x)\ \lambda x.(x\ x))$...

Dans le cadre de son approche des fondements des mathématiques, Church a également introduit une variante typée de ce système (pour éviter certains paradoxes) :

- Les termes sont maintenant annotés avec des types.
- Les types sont définis par la grammaire :

$$t, t' := T \mid t \to t'$$

  où est une collection de types de base et $t \to t$ représente intuitivement les fonctions qui prennent un argument de type t et renvoient un argument de type t.
- Les termes de ce nouveau calcul ne sont valides que si on peut leur attribuer un type. Par exemple, si $x : A$ alors $\lambda x.x : A \to A$. Si, de plus, $y : A$, alors $(\lambda x.x\ y) : A$.
- Essayez de trouver un type pour $(\lambda x.(x\ x)\ \lambda x.(x\ x))$... Pro-tip : il n'y en a pas !

Existe-t-il un moyen de vérifier si un terme est bien formé ?

Existe-t-il un moyen de vérifier si un terme est bien formé ? Oui ! Il est bien formé s'il peut être dérivé à l'aide des règles de typage suivantes :

Existe-t-il un moyen de vérifier si un terme est bien formé ? Oui ! Il est bien formé s'il peut être dérivé à l'aide des règles de typage suivantes :

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash y : B}{\Gamma \vdash \lambda x.y : A \to B}$$

$$\frac{\Gamma, f : A \to B \qquad \Gamma \vdash x : B}{\Gamma \vdash (f\ x) : B}$$

Existe-t-il un moyen de vérifier si un terme est bien formé ? Oui ! Il est bien formé s'il peut être dérivé à l'aide des règles de typage suivantes :

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash y : B}{\Gamma \vdash \lambda x.y : A \to B}$$

$$\frac{\Gamma, f : A \to B \qquad \Gamma \vdash x : B}{\Gamma \vdash (f\ x) : B}$$

Mais attendez un peu, si nous effaçons les termes et ne gardons que les types, nous obtenons...

Lambda-calcul simplement typé

Existe-t-il un moyen de vérifier si un terme est bien formé ? Oui ! Il est bien formé s'il peut être dérivé à l'aide des règles de typage suivantes :

$$\frac{}{\Gamma, A \vdash A} \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}$$

$$\frac{\Gamma, A \to B \qquad \Gamma \vdash B}{\Gamma \vdash B}$$

Mais attendez un peu, si nous effaçons les termes et ne gardons que les types, nous obtenons... un sous-ensemble de la déduction naturelle de Gentzen !

La fin

- Cette relation est valable pour la déduction naturelle intuitionniste complète (le système sans la règle PTE), si nous incluons des primitives pour les paires dans notre calcul.

La fin

- Cette relation est valable pour la déduction naturelle intuitionniste complète (le système sans la règle PTE), si nous incluons des primitives pour les paires dans notre calcul.
- Nous avons eu un petit aperçu de la célèbre correspondance Curry-Howard :

  Les preuves sont des types, les types sont des preuves !

- Cette relation est valable pour la déduction naturelle intuitionniste complète (le système sans la règle PTE), si nous incluons des primitives pour les paires dans notre calcul.
- Nous avons eu un petit aperçu de la célèbre correspondance Curry-Howard :

Les preuves sont des types, les types sont des preuves !

- Cette correspondance s'étend bien au-delà de ce que nous avons vu : logique des prédicats, types dépendants, théorie des types homotopiques, ...

La fin

- Cette relation est valable pour la déduction naturelle intuitionniste complète (le système sans la règle PTE), si nous incluons des primitives pour les paires dans notre calcul.
- Nous avons eu un petit aperçu de la célèbre correspondance Curry-Howard :

  Les preuves sont des types, les types sont des preuves !

- Cette correspondance s'étend bien au-delà de ce que nous avons vu : logique des prédicats, types dépendants, théorie des types homotopiques, ...

$$\frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma, x : A \vdash y : B}{\Gamma \vdash \lambda x . y : A \to B}$$

$$\frac{\Gamma, f : A \to B \qquad \Gamma \vdash x : B}{\Gamma \vdash (f\ x) : B}$$