Leçon 4 : Proof-theoretic semantics, vérificationnisme et pragmatisme

Justification et encodage des règles logiques et calculatoires

Pablo Donato

Logique et Fondements de l'Informatique

Année 2026

La question centrale : d'où vient le sens d'un connecteur ?

La question centrale : d'où vient le sens d'un connecteur ?

En déduction naturelle, chaque connecteur possède deux types de règles :

Introductions — comment construire

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee I_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee I_2$$

Élimination — comment utiliser

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee E$$

Question philosophique

Lesquelles des deux sont *premières* ? Lesquelles **définissent** ce que *signifie* le connecteur ?

La question centrale : d'où vient le sens d'un connecteur ?

En déduction naturelle, chaque connecteur possède deux types de règles :

Introductions — comment construire

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee I_1 \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee I_2$$

Élimination — comment utiliser

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee E$$

Question philosophique

Lesquelles des deux sont *premières* ? Lesquelles **définissent** ce que *signifie* le connecteur ?

Deux grandes réponses en *proof-theoretic semantics* (théorie de la démonstration).

Le vérificationnisme (Gentzen, Prawitz, Dummett)

Principe

Le sens d'un connecteur = ses **conditions de vérification** = ses règles d'**introduction**.

- Pour $\vee$: prouver $A \vee B$, c'est apporter une preuve de A, ou une preuve de B.

Le vérificationnisme (Gentzen, Prawitz, Dummett)

Principe

Le sens d'un connecteur = ses **conditions de vérification** = ses règles d'**introduction**.

- Pour $\vee$: prouver $A \vee B$, c'est apporter une preuve de A, ou une preuve de B.
- Les règles d'élimination sont **dérivées canoniquement** (principe d'inversion de Prawitz) : une élimination est justifiée si elle *préserve la canonicité des preuves*.

Le vérificationnisme (Gentzen, Prawitz, Dummett)

Principe

Le sens d'un connecteur = ses **conditions de vérification** = ses règles d'**introduction**.

- Pour $\vee$: prouver $A \vee B$, c'est apporter une preuve de A, ou une preuve de B.
- Les règles d'élimination sont **dérivées canoniquement** (principe d'inversion de Prawitz) : une élimination est justifiée si elle *préserve la canonicité des preuves*.
- **Fundamental Assumption** (Dummett) : toute proposition prouvable possède une *preuve canonique*, se terminant *héréditairement* par une règle d'introduction.
  - C'est un *(meta-)théorème* en logique intuitionniste (normalisation de Prawitz).
  - Il permet de remplacer la sémantique tarskiste par une sémantique de preuves.
  - Philosophiquement : anti-réalisme, signification fondée sur l'usage (Wittgenstein) plutôt que sur la dénotation (Frege).

Slogan : « On comprend $\vee$ parce qu'on sait comment le construire. »

Le pragmatisme (Zeilberger §2.1, d'après Dummett)

Principe

Le sens d'un connecteur = ses **conditions d'utilisation** = ses règles d'**élimination**.

- (Dummett) : les éliminations sont tout aussi valides comme point de départ.

Le pragmatisme (Zeilberger §2.1, d'après Dummett)

Principe

Le sens d'un connecteur = ses **conditions d'utilisation** = ses règles d'**élimination**.

- (Dummett) : les éliminations sont tout aussi valides comme point de départ.
- Pour $\vee$: une preuve de $A \vee B$, c'est ce qui sait répondre à *tout usage canonique* — i.e., produire C dès qu'on sait dériver C de A et C de B.

Le pragmatisme (Zeilberger §2.1, d'après Dummett)

Principe

Le sens d'un connecteur = ses **conditions d'utilisation** = ses règles d'**élimination**.

- (Dummett) : les éliminations sont tout aussi valides comme point de départ.
- Pour $\vee$: une preuve de $A \vee B$, c'est ce qui sait répondre à *tout usage canonique* — i.e., produire C dès qu'on sait dériver C de A et C de B.
- **Procédure de justification descendante** (duale de Prawitz) : une règle est valide si toute *conséquence canoniquement obtenue* de sa conclusion peut être transformée en une conséquence canoniquement obtenue de ses prémisses.
  - $\uparrow$ Vérif. : preuve canonique = preuve *terminant par* des introductions
  - $\downarrow$ Pragma. : conséquence canonique = conséquence *commençant par* des éliminations

Slogan : « On comprend $\vee$ parce qu'on sait à quoi il sert. »

Les types inductifs : le choix vérificationniste de Rocq

Rocq choisit le vérificationnisme : on **déclare les constructeurs** directement.

Définitions réelles de la bibliothèque standard :

```
Inductive and (A B : Prop) : Prop :=
  | conj : A -> B -> A /\ B.
```

```
Inductive or (A B : Prop) : Prop :=
  | or_introl : A -> A \/ B
  | or_intror : B -> A \/ B.
```

Les types inductifs : le choix vérificationniste de Rocq

Rocq choisit le vérificationnisme : on **déclare les constructeurs** directement.

Définitions réelles de la bibliothèque standard :

```
Inductive and (A B : Prop) : Prop :=
  | conj : A -> B -> A /\ B.
```

```
Inductive or (A B : Prop) : Prop :=
  | or_introl : A -> A \/ B
  | or_intror : B -> A \/ B.
```

- Les éliminateurs (`and_ind`, `or_ind`) sont **dérivés automatiquement** — avec leur contenu calculatoire complet.

Les types inductifs : le choix vérificationniste de Rocq

Rocq choisit le vérificationnisme : on **déclare les constructeurs** directement.

Définitions réelles de la bibliothèque standard :

```
Inductive and (A B : Prop) : Prop :=
  | conj : A -> B -> A /\ B.
```

```
Inductive or (A B : Prop) : Prop :=
  | or_introl : A -> A \/ B
  | or_intror : B -> A \/ B.
```

- Les éliminateurs (`and_ind`, `or_ind`) sont **dérivés automatiquement** — avec leur contenu calculatoire complet.
- C'est le vérificationnisme incarné : les introductions définissent tout, les éliminations *s'en suivent*.

Interlude : encodages de Church des connecteurs

Connecteur	Encodage de Church (pragmatiste)
$A \vee B$	$\forall P,\ (A \to P) \to (B \to P) \to P$
$A \wedge B$	$\forall P,\ (A \to B \to P) \to P$
$\bot$	$\forall P,\ P$
$\exists x,\ P\,x$	$\forall Q,\ (\forall x,\ P\,x \to Q) \to Q$

- Valides dans le **Calcul des Constructions pur** (sans `Inductive`).
- En logique du 2$^{\text{nd}}$ ordre — ce qui est le cas de Rocq, où l'on peut quantifier sur `Prop`.
- Logiquement correctes, mais *coûteuses en pratique*.

Pourquoi Rocq choisit `Inductive` plutôt que Church

Les deux approches sont **logiquement équivalentes** en logique du 2nd ordre (Rocq peut quantifier sur `Prop`).

Mais l'encodage de Church a des coûts calculatoires sérieux :

- Une preuve de $A \vee B$ est une *fonction* (λ-terme), pas une donnée : son usage nécessite des β-réductions coûteuses.
- Les structures de données encodées à la Church (listes, arbres...) sont inutilisables pour de vrais programmes car trop lentes (complexité en temps).

Pourquoi Rocq choisit `Inductive` plutôt que Church

Les deux approches sont **logiquement équivalentes** en logique du 2nd ordre (Rocq peut quantifier sur `Prop`).

Mais l'encodage de Church a des coûts calculatoires sérieux :

- Une preuve de $A \vee B$ est une *fonction* (λ-terme), pas une donnée : son usage nécessite des β-réductions coûteuses.
- Les structures de données encodées à la Church (listes, arbres...) sont inutilisables pour de vrais programmes car trop lentes (complexité en temps).

Avec `Inductive` :

- Représentation **concrète en mémoire** (e.g. les paires et injections vues en Leçon 2).
- **Pattern-matching** compilable directement en code machine efficace.
- Éliminateurs dérivés avec leur contenu calculatoire complet.

Pourquoi Rocq choisit `Inductive` plutôt que Church

Les deux approches sont **logiquement équivalentes** en logique du 2nd ordre (Rocq peut quantifier sur `Prop`).

Mais l'encodage de Church a des coûts calculatoires sérieux :

- Une preuve de $A \vee B$ est une *fonction* (λ-terme), pas une donnée : son usage nécessite des β-réductions coûteuses.
- Les structures de données encodées à la Church (listes, arbres...) sont inutilisables pour de vrais programmes car trop lentes (complexité en temps).

Avec `Inductive` :

- Représentation **concrète en mémoire** (e.g. les paires et injections vues en Leçon 2).
- **Pattern-matching** compilable directement en code machine efficace.
- Éliminateurs dérivés avec leur contenu calculatoire complet.

L'ironie

Rocq choisit le vérificationnisme pour des raisons... *pragmatiques* !

L'égalité revisitée : eq est inductive

La définition réelle de = dans Rocq :

```
Inductive eq (A : Type) (x : A) : A -> Prop :=
  | eq_refl : eq A x x.
```

- Un **seul constructeur** : on ne peut prouver `x = y` qu'en exhibant que `x` et `y` sont *le même objet*.

L'égalité revisitée : eq est inductive

La définition réelle de = dans Rocq :

```
Inductive eq (A : Type) (x : A) : A -> Prop :=
  | eq_refl : eq A x x.
```

- Un **seul constructeur** : on ne peut prouver `x = y` qu'en exhibant que `x` et `y` sont *le même objet*.
- Définition vérificationniste *maximalement restrictive* : la seule preuve canonique de `x = y` est la réflexivité.

L'égalité revisitée : eq est inductive

La définition réelle de = dans Rocq :

```
Inductive eq (A : Type) (x : A) : A -> Prop :=
  | eq_refl : eq A x x.
```

- Un **seul constructeur** : on ne peut prouver `x = y` qu'en exhibant que `x` et `y` sont *le même objet*.
- Définition vérificationniste *maximalement restrictive* : la seule preuve canonique de `x = y` est la réflexivité.
- L'éliminateur dérivé automatiquement est... le **principe de Leibniz** :

$$\boxed{x = y \textit{ ssi } \text{pour toute propriété } P,\ P(x) \Rightarrow P(y)}$$

Leibniz justifié par Prawitz

Leibniz (*Discours de métaphysique*, 1686) :

> *Deux choses sont identiques si et seulement si elles partagent toutes leurs propriétés (indiscernabilité des identiques).*

Leibniz justifié par Prawitz

Leibniz (*Discours de métaphysique*, 1686) :

> *Deux choses sont identiques si et seulement si elles partagent toutes leurs propriétés (indiscernabilité des identiques).*

Dans Rocq, `rewrite` = application de l'éliminateur de eq.

Leibniz justifié par Prawitz

Leibniz (*Discours de métaphysique*, 1686) :

> *Deux choses sont identiques si et seulement si elles partagent toutes leurs propriétés (indiscernabilité des identiques).*

Dans Rocq, `rewrite` = application de l'éliminateur de eq.

La chaîne de justification :

1. eq est défini avec un seul introducteur : `eq_refl`
2. Son éliminateur (dérivé vérificationnistiquement) = substitution universelle
3. Cette substitution *est* le principe de Leibniz

Ouverture : HoTT et au-delà de Leibniz

Que *signifie* une preuve de `x = y` au-delà de la réflexivité ?

Ouverture : HoTT et au-delà de Leibniz

Que *signifie* une preuve de `x = y` au-delà de la réflexivité ?

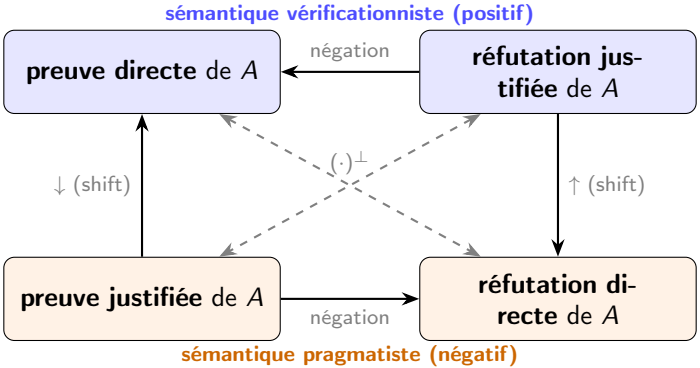
Homotopy Type Theory — HoTT (Voevodsky, ∼2006–2013)

- Interpréter les *types* comme des **espaces topologiques** (au sens de la théorie de l'homotopie).
- Les *termes* = points dans l'espace.
- Les *preuves d'égalité* = **chemins** entre deux points.
- Une preuve de `x = y` peut avoir une *structure propre* : plusieurs chemins distincts entre les mêmes points.
- **Axiome d'univalence** : deux types équivalents sont *égaux*.

La question « qu'est-ce que l'égalité ? » reste ouverte et philosophiquement/mathématiquement féconde.

Ouverture : le Calculus of Unity (Zeilberger, 2008)

Dummett a montré que les deux approches sont cohérentes. Zeilberger propose de les **unifier** en distinguant quatre jugements, sur le modèle du *carré logique d'Aristote* :



Via Curry-Howard : **stratégies d'évaluation** (call-by-value $(+)$ vs. call-by-name $(-)$)

Synthèse

	Vérificationnisme	**Pragmatisme**
Slogan	Sens = comment construire	Sens = comment utiliser
Philosophes	Gentzen, Prawitz, Dummett	(Dummett / Zeilberger)
En Rocq	`Inductive` (constructeurs)	Encodage Church (λ-termes purs)
Éliminateurs	Dérivés auto., calculatoire efficace	Lisibles dans le type, mais inefficaces
`rewrite`	Éliminateur de eq = Leibniz	—
Avantage	Efficacité (choix de Rocq)	Uniformité, pas de primitives
Unification	Calculus of Unity (Zeilberger 2008) : polarités $+/-$	

Références I

[Dum91] Michael Dummett. *The Logical Basis of Metaphysics*. The William James Lectures, 1976. Cambridge, Massachusetts : Harvard University Press, 1991.

[Pra65] Dag Prawitz. *Natural Deduction : A Proof-Theoretical Study*. Stockholm : Almqvist & Wiksell, 1965.

[Zei08] Noam Zeilberger. "On the Unity of Duality". In : *Annals of Pure and Applied Logic* 153.1–3 (avr. 2008), p. 66-96. issn : 01680072. doi : 10.1016/j.apal.2008.01.001. url : https://linkinghub.elsevier.com/retrieve/pii/S0168007208000080 (visité le 07/03/2026).