

Introduction aux quantificateurs

INF110 (Logique et Fondements de l'Informatique)

David A. Madore
Télécom Paris
david.madore@enst.fr

2023–2024

<http://perso.enst.fr/madore/inf110/transp-inf110.pdf>

Git: 10c0912 Fri Jan 26 18:53:33 2024 +0100

Plan

Les
quantificateurs :
discussion
informelle

Logique du premier
ordre

Arithmétique du
premier ordre et
théorème de Gödel

Les quantificateurs : discussion informelle

Logique du premier ordre

Arithmétique du premier ordre et théorème de Gödel

Limitations du calcul propositionnel

- ▶ On a parlé pour l'instant de **calcul propositionnel**, qui ne connaît que les affirmations logiques et les connecteurs propositionnels $\Rightarrow, \wedge, \vee, \top, \perp$.
- ▶ Mais il y a deux notations logiques essentielles en mathématiques au-delà de ces connecteurs : les **quantificateurs** \forall, \exists , qui :
 - ▶ prennent une formule $P(v)$ dépendant d'une variable v libre (de type I),
 - ▶ lient cette variable, pour former une nouvelle formule : $\forall(v : I).P(v)$ ou $\exists(v : I).P(v)$ (parfois juste $\forall v.P(v)$ et $\exists v.P(v)$).
- ▶ Intuitivement, il faut penser à \forall et \exists comme des « \wedge et \vee en famille », c'est-à-dire que :
 - ▶ $\forall v.P(v)$, parfois noté $\bigwedge_v P(v)$ est à $P \wedge Q$ ce que $\prod_i p_i$ est à $p \times q$,
 - ▶ $\exists v.P(v)$, parfois noté $\bigvee_v P(v)$ est à $P \vee Q$ ce que $\sum_i p_i$ est à $p + q$.
- ▶ Il existe de **nombreux systèmes logiques** différant notamment en **ce qu'on a le droit de quantifier** (qui sont les v ici ? quel est leur domaine I ?).

Commençons par une discussion informelle de \forall et \exists .

On a déjà vu l'interprétation informelle des connecteurs, on introduit maintenant les quantificateurs :

- ▶ un témoignage de $P \wedge Q$, est un témoignage de P et un de Q ,
- ▶ un témoignage de $P \vee Q$, est un témoignage de P ou un de Q , et la donnée duquel des deux on a choisi,
- ▶ un témoignage de $P \Rightarrow Q$ est un moyen de transformer un témoignage de P en un témoignage de Q ,
- ▶ un témoignage de \top est trivial, ▶ un témoignage de \perp n'existe pas,
- ▶ un témoignage de $\forall v.P(v)$ est un moyen de transformer un x quelconque en un témoignage de $P(x)$,
- ▶ un témoignage de $\exists v.P(v)$ est la donnée d'un certain t_0 et d'un témoignage de $P(t_0)$.

Curry-Howard pour le \forall

- ▶ On a vu que Curry-Howard fait correspondre **conjonction logique** $P \wedge Q$ (« un témoignage de P et un de Q ») avec **type produit** $\sigma \times \tau$ (« une valeur de σ et une de τ »).
- ▶ De façon analogue, la **quantification universelle** $\forall v.P(v)$ (« une façon de transformer v en un témoignage de $P(v)$ »), qui est une sorte de *conjonction en famille* $\bigwedge_v P(v)$, correspondra au **type produit en famille** $\prod_v \sigma(v)$ (« fonction renvoyant pour chaque v une valeur de $\sigma(v)$ »).
- ▶ Ceci présuppose l'existence de **familles de types** $v \mapsto \sigma(v)$ (= types dépendant de quelque chose) dont on puisse prendre le produit.
- ▶ Une preuve de $\forall(v : I).P(v)$ correspondra à un terme de forme $\lambda(v : I).(\dots)$, où le type de (\dots) correspond à $P(v)$.
- ▶ Remarquer que $\forall(v : I).P$, si P ne dépend pas de v , « ressemble » à $I \Rightarrow P$ de la même manière que $\prod_{i \in I} S = S^I$ (ensemblissement ou numériquement). (Les détails dépendent de la nature de la quantification.)

Curry-Howard pour le \exists

► On a vu que Curry-Howard fait correspondre **disjonction logique** $P \vee Q$ (« un témoignage de P ou un de Q , avec la donnée duquel on a choisi ») avec **type somme** $\sigma + \tau$ (« une valeur de σ ou une de τ , avec un sélecteur entre les deux »).

► De façon analogue, la **quantification existentielle** $\exists v.P(v)$ (« la donnée d'un t_0 et d'un témoignage de $P(t_0)$ »), qui est une sorte de *disjonction en famille* $\bigvee_v P(v)$, correspondra au **type somme en famille** $\sum_v \sigma(v)$ (« donnée d'un t_0 et d'une valeur de type $\sigma(t_0)$ »).

► Une preuve de $\exists(v : I).P(v)$ correspondra à un terme de forme $\langle t_0, \dots \rangle$, où le type de (\dots) correspond à $P(t_0)$. (De nouveau, il faut des « familles de types ».)

► Remarquer que $\exists(v : I).P$, si P ne dépend pas de v , « ressemble » à $I \times P$ de la même manière que $\sum_{i \in I} S = I \times S$. (Les détails dépendent de la nature de la quantification.)

► Mais Curry-Howard atteint ses limites : il n'est pas dit que d'une preuve de $\exists v.P(v)$ on **puisse extraire** le t_0 correspondant dans autre chose qu'une preuve. (Les détails dépendent du système logique précis considéré et si Martin-Löf est dans la salle.)

Règles d'introduction et d'élimination de \forall

Les règles ci-dessous (et transp. suivants) sont **incomplètes** : il manque des explications sur le type I sur lequel on quantifie et comment on peut en former des « termes d'individus ».

► Introduction du \forall : pour montrer $\forall(v : I). Q$, on s'arrange (quitte à renommer la variable liée) pour que $v : I$ soit « frais », c'est-à-dire qu'il n'apparaisse (libre) dans **aucune hypothèse** en cours : si on montre Q sur ce v « arbitraire », on peut conclure $\forall(v : I). Q$.

(Rédaction : « soit v arbitraire (...) on a $Q(v)$; donc $\forall(v : I). Q(v)$ ».)

Ceci donnera un λ -terme noté $\lambda(v : I).(\dots)$ comme l'ouverture d'une hypothèse.

► Élimination du \forall : pour utiliser $\forall(v : I). Q$, on peut l'appliquer à un t quelconque (un **terme** de type I).

(Rédaction : « on a $\forall(v : I). Q(v)$ et t de type I ; en particulier, on a $Q(t)$ ».)

Ceci donnera un λ -terme noté ft comme l'application d'une implication.

Règles d'introduction et d'élimination de \exists

► Introduction du \exists : pour montrer $\exists(v : I).Q$, on peut le montrer sur un terme t quelconque de type I .

(Rédaction : « pour ce t de type I on a $Q(t)$; en particulier, on a $\exists(v : I).Q(v)$ ».)

Ceci donnera un λ -terme noté $\langle t, \dots \rangle$ comme pour une conjonction.

► Élimination du \exists : pour utiliser $\exists(v : I).P(v)$ pour montrer une conclusion Q , on s'arrange (quitte à renommer la variable liée) pour que v soit « frais », c'est-à-dire qu'il n'apparaisse (libre) dans **aucune hypothèse** en cours **ni dans la conclusion** Q : si on montre Q à partir de P sur ce v « arbitraire », on peut conclure Q à partir de $\exists v.P$.

(Rédaction : « on a $\exists(v : I).P(v)$: soit v arbitraire tel que $P(v) : (\dots)$ on a Q ; donc Q ».)

Ceci donnera un λ -terme noté $(\text{match } \dots \text{ with } \langle v, h \rangle \mapsto \dots)$.

Aperçu d'ensemble des règles de la déduction naturelle

	Intro	Élim
\Rightarrow	$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}$	$\frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}$
\wedge	$\frac{\Gamma \vdash Q_1 \quad \Gamma \vdash Q_2}{\Gamma \vdash Q_1 \wedge Q_2}$	$\frac{\Gamma \vdash Q_1 \wedge Q_2 \quad \Gamma \vdash Q_1 \wedge Q_2}{\Gamma \vdash Q_1 \quad \Gamma \vdash Q_2}$
\vee	$\frac{\Gamma \vdash Q_1}{\Gamma \vdash Q_1 \vee Q_2} \quad \frac{\Gamma \vdash Q_2}{\Gamma \vdash Q_1 \vee Q_2}$	$\frac{\Gamma \vdash P_1 \vee P_2 \quad \Gamma, P_1 \vdash Q \quad \Gamma, P_2 \vdash Q}{\Gamma \vdash Q}$
\top	$\overline{\Gamma \vdash \top}$	(néant)
\perp	(néant)	$\frac{\Gamma \vdash \perp}{\Gamma \vdash Q}$ (ou pour la logique classique : $\frac{\Gamma, \neg Q \vdash \perp}{\Gamma \vdash Q}$)
\forall	$\frac{\Gamma, v : I \vdash Q}{\Gamma \vdash \forall(v : I). Q}$ (<i>v frais</i>)	$\frac{\Gamma \vdash \forall(v : I). Q \quad \Gamma \vdash t : I}{\Gamma \vdash Q[v \setminus t]}$
\exists	$\frac{\Gamma \vdash t : I \quad \Gamma \vdash Q[v \setminus t]}{\Gamma \vdash \exists(v : I). Q}$	$\frac{\Gamma \vdash \exists(v : I). P \quad \Gamma, v : I, P \vdash Q}{\Gamma \vdash Q}$ (<i>v frais</i>)

► « *v* frais » = « *v* n'apparaît nulle part ailleurs » (cf. transp. précédents).

► Le contexte Γ peut contenir des formules (hypothèses) et des variables d'« individus » (avec leur type, p.ex. $v : I$).

Notations des λ -termes pour \forall

► On a vu en calcul propositionnel qu'on peut noter les démonstrations par des « λ -termes » qui peuvent ensuite être réinterprétés comme des programmes (c'est Curry-Howard).

Complétons ces notations pour \forall, \exists :

► Introduction du \forall : si s désigne une preuve de Q faisant intervenir v variable libre de type I , on notera $\lambda(v : I).s$ la preuve de $\forall(v : I).Q$ obtenue par introduction du \forall . (**N.B.** v peut apparaître dans Q mais pas dans Γ .)

$$\frac{\Gamma, v : I \vdash s : Q}{\Gamma \vdash \lambda(v : I).s : \forall(v : I).Q}$$

► Élimination du \forall : si f désigne une preuve de $\forall(v : I).Q$ et t un terme de type I , on notera ft la preuve de $Q[v \setminus t]$ (c'est-à-dire Q avec v remplacé par t) obtenue par élimination du \forall sur ce terme.

(**N.B.** on n'explique pas comment le « terme d'individu » t est formé.)

$$\frac{\Gamma \vdash f : \forall(v : I).Q \quad \Gamma \vdash t : I}{\Gamma \vdash ft : Q[v \setminus t]}$$

► Ceci est conforme à l'idée de BHK : une preuve de $\forall(v : I).Q(v)$ prend un x de type I et renvoie une preuve de $Q(x)$.

Notations des λ -termes pour \exists

► Introduction du \exists : si t désigne un terme de type I et z une preuve de $Q[v \setminus t]$ (pour ce t -là, donc), on notera $\langle t, z \rangle$ la preuve de $\exists(v : I). Q$ obtenue par introduction du \exists .

$$\frac{\Gamma \vdash t : I \quad \Gamma \vdash z : Q[v \setminus t]}{\Gamma \vdash \langle t, z \rangle : \exists(v : I). Q}$$

► Élimination du \exists : si z désigne une preuve de $\exists(v : I). P$ et s une preuve, faisant intervenir v variable libre de type I , de Q qui ne fait pas intervenir v , et h hypothèse supposant P (pour ce v -là, donc), on notera $(\text{match } z \text{ with } \langle v, h \rangle \mapsto s)$ la preuve de Q obtenue par élimination du \exists .
(**N.B.** v peut apparaître dans s mais pas dans Γ ni Q .)

$$\frac{\Gamma \vdash z : \exists(v : I). P \quad \Gamma, v : I, h : P \vdash s : Q}{\Gamma \vdash (\text{match } z \text{ with } \langle v, h \rangle \mapsto s) : Q}$$

► Ceci est conforme à l'idée de BHK : une preuve de $\exists(v : I). P(v)$ est la donnée d'un t de type I d'une preuve de $Q(t)$.

Récapitulatif des notations

(À comparer au transp. 9.)

	Intro	Élim
\Rightarrow	$\frac{\Gamma, v : P \vdash s : Q}{\Gamma \vdash \lambda(v : P). s : P \Rightarrow Q}$	$\frac{\Gamma \vdash f : P \Rightarrow Q \quad \Gamma \vdash z : P}{\Gamma \vdash fz : Q}$
\wedge	$\frac{\Gamma \vdash z_1 : Q_1 \quad \Gamma \vdash z_2 : Q_2}{\Gamma \vdash \langle z_1, z_2 \rangle : Q_1 \wedge Q_2}$	$\frac{\Gamma \vdash z : Q_1 \wedge Q_2}{\Gamma \vdash \pi_1 z : Q_1} \quad \frac{\Gamma \vdash z : Q_1 \wedge Q_2}{\Gamma \vdash \pi_2 z : Q_2}$
\vee	$\frac{\Gamma \vdash z : Q_i}{\Gamma \vdash \iota_i^{(Q_1, Q_2)} z : Q_1 \vee Q_2} \quad (i \in \{1, 2\})$	$\frac{\Gamma \vdash r : P_1 \vee P_2 \quad \Gamma, h_1 : P_1 \vdash s_1 : Q \quad \Gamma, h_2 : P_2 \vdash s_2 : Q}{\Gamma \vdash (\text{match } r \text{ with } \iota_1 h_1 \mapsto s_1, \iota_2 h_2 \mapsto s_2) : Q}$
\top	$\overline{\Gamma \vdash \bullet : \top}$	(néant)
\perp	(néant)	$\frac{\Gamma \vdash r : \perp}{\Gamma \vdash \text{exfalse}^{(Q)} r : Q}$
\forall	$\frac{\Gamma, v : I \vdash s : Q}{\Gamma \vdash \lambda(v : I). s : \forall(v : I). Q}$	$\frac{\Gamma \vdash f : \forall(v : I). Q \quad \Gamma \vdash t : I}{\Gamma \vdash ft : Q[v \setminus t]}$
\exists	$\frac{\Gamma \vdash t : I \quad \Gamma \vdash z : Q[v \setminus t]}{\Gamma \vdash \langle t, z \rangle : \exists(v : I). Q}$	$\frac{\Gamma \vdash z : \exists(v : I). P \quad \Gamma, v : I, h : P \vdash s : Q}{\Gamma \vdash (\text{match } z \text{ with } \langle v, h \rangle \mapsto s) : Q}$

► Les séquents en gris sont des formations de termes d'individus (cf. transp. suivant).

Monde des termes et monde logique

- ▶ Les règles pour les démonstrations écrites notamment transp. 9 à 12 **ne sont pas complètes**, il manque les explications sur les séquents en gris (formation des termes).
- ▶ Selon le système logique, on peut distinguer deux « mondes » **plus ou moins séparés ou confondus** :
 - ▶ le monde des termes (individus) et types (d'individus),
 - ▶ le monde logique, avec preuves et propositions.
- ▶ Les règles données aux transp. précédents sont les règles de construction des **démonstrations** (\rightarrow monde logique), où se placent tous les séquents sauf ceux marqués en gris.
- ▶ Les règles du monde des termes peuvent être calquées sur le monde des démonstrations, plus simples, ou différentes.
- ▶ En Coq, les deux mondes sont **séparés mais parallèles** : *Prop* pour le type des propositions et *Type* pour le type des types d'individus.

Que peut-on quantifier au juste ?

- ▶ Outre la question des termes d'individus et leur séparation du monde logique, il manque les explications sur ce qu'on a le droit de quantifier et d'abstraire ; notamment :
 - ▶ peut-on former des propositions comme $\forall(A : *). (A \Rightarrow A)$ (où « $*$ » est le type des propositions) avec preuve $\lambda(A : *). \lambda(h : A). h$, i.e., quantifier sur les propositions (et abstraire dessus dans les termes) ?
 - ▶ peut-on former des objets comme $\lambda(A : *). A$ de type $* \rightarrow *$, i.e., abstraire sur les propositions ?
 - ▶ peut-on former des propositions comme $\forall(x : I). A(x)$ où A a pour type $I \rightarrow *$, i.e., quantifier et abstraire sur des individus ?
- ▶ Différents systèmes logiques diffèrent dans la réponse à ces questions, notamment, les 8 systèmes du « λ -cube » de Barendregt (jusqu'à mélanger complètement preuves et individus).

Le problème du \exists et des types sommes

Doit-on croire à ceci (pour U et V deux types) ?

$$(\forall(x : U). \exists(y : V). P(x, y)) \Rightarrow (\exists(f : U \Rightarrow V). \forall(x : U). P(x, f(x)))$$

« preuve(?) » : $\lambda(h : \dots). \langle \lambda(x : U). (\text{match } hx \text{ with } \langle v, z \rangle \mapsto v),$
 $\lambda(x : U). (\text{match } hx \text{ with } \langle v, z \rangle \mapsto z) \rangle$

(Cet énoncé porte le nom d'**axiome du choix** : c'est un analogue pour la théorie des types de l'axiome du choix (de Zermelo) en théorie des ensembles.)

- ▶ Si on voit \forall et \exists comme des types produit et **somme** en famille respectivement, **oui** : $\forall(x : U). \exists(y : V). P(x, y)$ représente une fonction qui prend un x de type U et renvoie un y de type V ainsi qu'un $P(x, y)$ correspondant : on peut collecter tous ces y en une fonction $f : U \Rightarrow V$.
- ▶ Si on voit \exists comme un quantificateur **logique**, alors **non** : le y renvoyé par \exists ne peut servir qu'à l'intérieur d'une preuve, pas être collecté en une fonction.
- ▶ C'est ici la différence principale entre des systèmes comme Coq (où l'énoncé ci-dessus ne sera pas prouvable pour $P : U \times V \rightarrow Prop$) et les systèmes à la Martin-Löf comme Agda (où Curry-Howard est suivi « jusqu'au bout » : il n'y a pas de \exists uniquement logique, et cet énoncé est prouvable comme indiqué).

Imprédicativité

► On appelle **imprédicativité** la possibilité de définir une proposition ou un type en quantifiant sur toutes les propositions ou types **y compris celui qu'on définit** : c'est une forme de circularité.

► P.ex., $\forall(Z : *). (Z \Rightarrow A)$ représente le type des fonctions capables de renvoyer un type A à partir d'un type Z quelconque, y compris celui qu'on définit.

Cette imprédicativité est utile pour définir des constructions sur les types.

Exemples (informellement, et en notant « $*$ » le « type des types » imprédicatif) :

- $A \cong \forall(Z : *). (Z \Rightarrow A)$: donné une valeur x de type A on peut en fabriquer une de type $Z \Rightarrow A$ comme $\lambda(z : Z). x$ pour tout type Z , mais réciproquement, donné une valeur de type $\forall(Z : *). (Z \Rightarrow A)$ on peut l'appliquer à $Z = \top$ pour obtenir une valeur de type A .
- $A \cong \forall(Z : *). ((A \Rightarrow Z) \Rightarrow Z)$: dans un sens on fabrique $\lambda(k : A \Rightarrow Z). kx$ comme pour le CPS, dans l'autre sens, appliquer à $Z = A$ et l'identité.
- $\perp \cong \forall(Z : *). Z$ ► $A \wedge B \cong \forall(Z : *). ((A \Rightarrow B \Rightarrow Z) \Rightarrow Z)$
- $A \vee B \cong \forall(Z : *). ((A \Rightarrow Z) \Rightarrow (B \Rightarrow Z) \Rightarrow Z)$

► Cela **peut** donner des incohérences logiques (paradoxe de Girard).

Logique du premier ordre : principe

► La **logique du premier ordre** ou **calcul des prédicats** (du 1^{er} ordre) est la plus simple qui ajoute les quantificateurs. Les « choses » sur lesquelles on a le droit de quantifier s'appellent des **individus**.

► Côté typage, elle n'est pas très heureuse : les « individus » apparaissent comme un type I unique, *ad hoc*, qu'on ne peut presque pas manipuler (la logique ne permet pas de faire des couples, fonctions, etc., des individus).

► Néanmoins, elle a une **grande importance mathématique** car le dogme « orthodoxe » est que :

Les mathématiques se font dans la « théorie des ensembles de Zermelo-Fraenkel en logique du premier ordre » (ZFC).

Le manque d'expressivité de la logique (pas de couples, fonctions, etc.) est **compensé par la théorie elle-même** (constructions ensemblistes des couples, fonctions, etc.).

► La **sémantique** (Tarskienne) de la logique du premier ordre a aussi des propriétés agréables (théorème de complétude de Gödel).

Logique du premier ordre : sortes de variables et syntaxe

- ▶ En (pure) logique du premier ordre, on a diverses sortes de variables :
 - ▶ les **variables d'individus** ($x, y, z\dots$) en nombre illimité,
 - ▶ les **variables de prédicats** n -aires, ou de **relations** n -aires [entre individus] ($A^{(n)}, B^{(n)}, C^{(n)}\dots$), pour chaque entier naturel n .
- ▶ L'indication d'arité des variables de prédicats est généralement omise (elle peut se lire sur la formule).
- ▶ Une **formule** (logique) est (inductivement) :
 - ▶ l'application $A^{(n)}(x_1, \dots, x_n)$ d'une variable propositionnelle à n variables d'individus,
 - ▶ l'application d'un connecteur : $(P \Rightarrow Q)$, $(P \wedge Q)$, $(P \vee Q)$ où P, Q sont deux formules, ou encore \top , \perp ,
 - ▶ une quantification : $\forall x.P$ ou $\exists x.P$ (pour $\forall(x : I).P$ ou $\exists(x : I).P$), qui **lie** la variable d'individu x dans P .
- ▶ On ne peut quantifier que sur les individus (« premier ordre »).

Exemples de formules du premier ordre

► Les **formules propositionnelles** sont encore des formules du premier ordre, en interprétant chaque variable propositionnelle comme une variable de prédicat 0-aire (« nulaire ») : $A \wedge B \Rightarrow B \wedge A$ par exemple.

Autres exemples (qui seront par ailleurs tous démontrables) :

- $(\forall x.A(x)) \wedge (\exists x.\top) \Rightarrow (\exists x.A(x))$ (ici, A est un prédicat unaire)
- $(\forall x.\neg A(x)) \Leftrightarrow (\neg \exists x.A(x))$ (idem)
- $(\exists x.\neg A(x)) \Rightarrow (\neg \forall x.A(x))$ (idem)
- $(\exists x.A) \Leftrightarrow (\exists x.\top) \wedge A$ (ici, A est un prédicat **nulaire**)
- $(\forall x.A) \Leftrightarrow ((\exists x.\top) \Rightarrow A)$ (idem)
- $(\exists x.\forall y.B(x, y)) \Rightarrow (\forall y.\exists x.B(x, y))$ (ici, B est un prédicat binaire)

N.B. On a suivi la convention que \forall, \exists ont une priorité plus faible que les connecteurs $\Rightarrow, \vee, \wedge, \neg$. Tout le monde n'est pas d'accord avec cette convention !

N.B.2 : Il serait peut-être préférable de noter Bxy que $B(x, y)$.

Remarques sur la logique du premier ordre

- ▶ Le type I (non écrit) des « individus », le seul sur lequel on peut quantifier, est **complètement spécial** en logique du premier ordre : on ne l'écrit même pas, on ne peut pas former $I \times I$ ni $I \rightarrow I$ ni rien d'autre.
- ▶ Ce type I ne se « mélange » pas aux relations A, B, C, \dots : les individus vivent dans un monde hermétiquement séparé des preuves.
- ▶ Dans la variante la plus simple, les seuls termes d'individus sont les variables d'individus (i.e., la seule façon d'obtenir $t : I$ est d'avoir $t : I$ dans le contexte !).
- ▶ On ne suppose pas I habité, i.e. $\exists x.\top$ n'est pas démontrable (pas plus que $\forall x.A(x) \Rightarrow \exists x.A(x)$). Cf. transp. 26.
- ▶ Pour avoir le droit d'écrire $A(x)$, la variable x doit avoir été introduite : la règle d'axiome devrait s'écrire correctement :

$$\frac{}{\Gamma, x : I \vdash x : I} \quad \text{et} \quad \frac{\Gamma \vdash x_1 : I \quad \dots \quad \Gamma \vdash x_n : I}{\Gamma, A(x_1, \dots, x_n) \vdash A(x_1, \dots, x_n)}$$

Logique du premier ordre : reprise des règles logiques

	Intro	Élim
\Rightarrow	$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q}$	$\frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}$
\wedge	$\frac{\Gamma \vdash Q_1 \quad \Gamma \vdash Q_2}{\Gamma \vdash Q_1 \wedge Q_2}$	$\frac{\Gamma \vdash Q_1 \wedge Q_2}{\Gamma \vdash Q_1} \quad \frac{\Gamma \vdash Q_1 \wedge Q_2}{\Gamma \vdash Q_2}$
\vee	$\frac{\Gamma \vdash Q_1}{\Gamma \vdash Q_1 \vee Q_2} \quad \frac{\Gamma \vdash Q_2}{\Gamma \vdash Q_1 \vee Q_2}$	$\frac{\Gamma \vdash P_1 \vee P_2 \quad \Gamma, P_1 \vdash Q \quad \Gamma, P_2 \vdash Q}{\Gamma \vdash Q}$
\top	$\overline{\Gamma \vdash \top}$	(néant)
\perp	(néant)	$\frac{\Gamma \vdash \perp}{\Gamma \vdash Q}$ (ou pour la logique classique : $\frac{\Gamma, \neg Q \vdash \perp}{\Gamma \vdash Q}$)
\forall	$\frac{\Gamma, x : I \vdash Q}{\Gamma \vdash \forall x. Q}$ (x frais)	$\frac{\Gamma \vdash \forall x. Q \quad \Gamma \vdash t : I}{\Gamma \vdash Q[x \setminus t]}$
\exists	$\frac{\Gamma \vdash t : I \quad \Gamma \vdash Q[x \setminus t]}{\Gamma \vdash \exists x. Q}$	$\frac{\Gamma \vdash \exists x. P \quad \Gamma, x : I, P \vdash Q}{\Gamma \vdash Q}$ (x frais)

► Les hypothèses en gris $\Gamma \vdash t : I$ (formation des termes) ne sont parfois pas écrites dans les arbres de démonstration, mais sont essentielles (cf. transp. 26).

Exemple de preuve : présentation drapeau

(1)	$\exists x. \forall y. B(x, y)$	
(2)	y'	
(3)	$x, \forall y. B(x, y)$	
(4)	$B(x, y')$	$\forall\text{Élim sur (3) et } y'$
(5)	$\exists x'. B(x', y')$	$\exists\text{Int sur } x \text{ et (4)}$
(6)	$\exists x'. B(x', y')$	$\exists\text{Elim sur (1) de (3) dans (5)}$
(7)	$\forall y'. \exists x'. B(x', y')$	$\forall\text{Int de (2) dans (6)}$
(8)	$(\exists x. \forall y. B(x, y)) \Rightarrow (\forall y'. \exists x'. B(x', y'))$	$\Rightarrow\text{Int de (1) dans (7)}$

« Supposons $\exists x. \forall y. B(x, y)$. Considérons un y' arbitraire. Considérons un x tel que $\forall y. B(x, y)$. En particulier, on a $B(x, y')$. En particulier, $\exists x'. B(x', y')$. Or on pouvait trouver un tel x car $\exists x. \forall y. B(x, y)$, donc on a bien la conclusion $\exists x'. B(x', y')$. Le choix de y' étant arbitraire, $\forall y'. \exists x'. B(x', y')$. Finalement, on a prouvé $(\exists x. \forall y. B(x, y)) \Rightarrow (\forall y'. \exists x'. B(x', y'))$. »

On reprend la preuve donnée aux transp. précédents :

$$\begin{array}{c}
 \text{Ax} \frac{}{x : I, \forall y.B(x, y), y' : I \vdash \forall y.B(x, y)} \quad \text{Ax} \frac{}{x : I, \forall y.B(x, y), y' : I \vdash B(x, y')} \quad \forall\text{ÉLIM} \\
 \exists\text{ÉLIM} \frac{\text{Ax} \frac{}{\exists x.\forall y.B(x, y) \vdash \exists x.\forall y.B(x, y)}}{\exists x.\forall y.B(x, y), y' : I \vdash \exists x'.B(x', y')} \quad \exists\text{INT} \frac{}{x : I, \forall y.B(x, y), y' : I \vdash \exists x'.B(x', y')} \\
 \forall\text{INT} \frac{}{\exists x.\forall y.B(x, y), y' : I \vdash \exists x'.B(x', y')} \\
 \Rightarrow\text{INT} \frac{}{\vdash (\exists x.\forall y.B(x, y)) \Rightarrow (\forall y'.\exists x'.B(x', y'))}
 \end{array}$$

Avec les notations du transp. 12 elle se note :

$$\lambda(u : \exists x.\forall y.B(x, y)). \lambda(y' : I). (\text{match } u \text{ with } \langle x, v \rangle \mapsto \langle x, vy' \rangle)$$

Exemples de preuves écrites comme λ -termes

► $(\forall x.(A(x) \Rightarrow C)) \Rightarrow ((\exists x.A(x)) \Rightarrow C)$

Preuve : $\lambda(f : \forall x.(A(x) \Rightarrow C)). \lambda(p : \exists x.A(x)). (\text{match } p \text{ with } \langle x, w \rangle \mapsto f x w)$

► $((\exists x.A(x)) \Rightarrow C) \Rightarrow (\forall x.(A(x) \Rightarrow C))$

Preuve : $\lambda(g : (\exists x.A(x)) \Rightarrow C). \lambda(x : I). \lambda(u : A(x)). g \langle x, u \rangle$

► Notamment pour C valant \perp on a prouvé $(\forall x.\neg A(x)) \Leftrightarrow \neg(\exists x.A(x))$ ci-dessus.

► $(\exists x.(A(x) \Rightarrow C)) \Rightarrow ((\forall x.A(x)) \Rightarrow C)$

Preuve : $\lambda(p : \exists x.(A(x) \Rightarrow C)). \lambda(f : \forall x.A(x)). (\text{match } p \text{ with } \langle x, w \rangle \mapsto w(f x))$

► Notamment pour C valant \perp on a prouvé $(\exists x.\neg A(x)) \Rightarrow \neg(\forall x.A(x))$ ci-dessus.

► $(\forall x.A(x)) \Rightarrow (\exists x.\top) \Rightarrow (\exists x.A(x))$

Preuve : $\lambda(f : \forall x.A(x)). \lambda(p : \exists x.\top). (\text{match } p \text{ with } \langle x, u \rangle \mapsto \langle x, f x \rangle)$

► $\forall z.\exists x.\top$ (cf. transp. suivant)

Preuve : $\lambda(z : I). \langle z, \bullet \rangle$

Pourquoi des variables d'individus avec les hypothèses ?

► L'introduction d'une variable d'individu libre porte en elle l'hypothèse que **l'univers des individus est habité** ($\exists x.T$). Ce fait **n'est pas prouvable** sans cette hypothèse. On a $z : I \vdash \exists x.T$ (ici z variable qcque) mais **on n'a pas** $\vdash \exists x.T$.

► Exiger que de pouvoir former $t : I$ à partir de Γ permet d'écartier la démonstration **incorrecte** suivante :

$$\begin{array}{c} \text{TINT} \frac{}{\vdash T} \\ \exists\text{INT} \frac{}{\vdash \exists x.T} \end{array}$$

► En revanche, celle-ci **est correcte** (en utilisant le terme z pour t dans $\exists\text{Int}$) :

$$\begin{array}{c} \text{AX} \frac{}{z : I \vdash z : I} \quad \frac{}{z : I \vdash T} \text{TINT} \\ \exists\text{INT} \frac{}{z : I \vdash \exists x.T} \\ \forall\text{INT} \frac{}{\vdash \forall z.\exists x.T} \end{array}$$

N.B. Ces problèmes n'ont rien à voir avec la logique intuitionniste, ils sont identiques en logique classique. On **n'a pas** non plus $\forall x.A(x) \Rightarrow \exists x.A(x)$.

Monde des individus et monde logique

- ▶ En logique du premier ordre, on a **deux « mondes » complètement séparés** :
 - ▶ le monde des individus, avec un seul type (I) et des variables sur lesquelles on peut quantifier,
 - ▶ le monde logique, avec propositions et preuves.
- ▶ Les propositions ont « moralement » un type (qu'on pourrait appeler « $*$ » ou « *Prop* »), mais on ne l'écrit pas. Les relations n -aires ont « moralement » le type $I^n \rightarrow Prop$, pas non plus écrit.
- ▶ Dans le transparent 21, tous les séquents concernent le monde logique (= construction des démonstrations), sauf ceux marqués en gris.
- ▶ Dans la version la plus simple de la logique du premier ordre (pas de *fonctions* d'individus, seulement des *relations*), la seule règle du monde des individus est :

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

(on ne peut former un terme d'individu qu'en invoquant une variable du contexte).

Curry-Howard pour la logique du premier ordre

- ▶ Il faut penser Curry-Howard dans le sens preuve \mapsto programme.
(Faute de description précise de règles de typage on ne peut pas espérer mieux ici.)
- ▶ Curry-Howard va mélanger le monde logique avec le monde des individus.
- ▶ On convertit les propositions en types :
 - ▶ $\Rightarrow, \wedge, \vee, \top, \perp$ deviennent $\rightarrow, \times, +, 1, 0$ comme en calcul propositionnel,
 - ▶ \forall, \exists deviennent produits et sommes \prod, \sum paramétrés par $v : I$.
- ▶ On convertit preuves en programmes selon l'interprétation fonctionnelle des notations données au transp. 12.
- ▶ P.ex., la preuve de $(\exists x. \forall y. B(x, y)) \Rightarrow (\forall y'. \exists x'. B(x', y'))$ donnée transp. 24 :
 $\lambda(u : \exists x. \forall y. B(x, y)). \lambda(y' : I). (\text{match } u \text{ with } \langle x, v \rangle \mapsto \langle x, v y' \rangle)$

devient un programme de type

$$\left(\sum_{x:I} \prod_{y:I} B(x, y) \right) \rightarrow \left(\prod_{y':I} \sum_{x':I} B(x', y') \right)$$

L'égalité au premier ordre

- ▶ En général on veut travailler en logique du premier ordre **avec égalité**.

C'est-à-dire qu'on introduit une relation binaire « = » (notée de façon infix) sujette aux **axiomes** suivants :

- ▶ réflexivité : $\text{refl} : \forall x.(x = x)$
- ▶ substitution : $\text{subst}^{(\lambda s.P(s))} : \forall x.\forall y.((x = y) \Rightarrow P(x) \Rightarrow P(y))$ pour toute formule $P(s)$ ayant une variable d'individu libre s .
- ▶ La logique du premier ordre montre ici ses limites : on n'a pas le droit de quantifier sur $P(s)$ ni même d'introduire $\lambda(s : I).P(s)$. Il faut donc comprendre qu'on a un « schéma d'axiomes » de substitution, dont chaque $\text{subst}^{(\lambda s.P(s))}$ est une instance (et $\lambda s.P(s)$ une notation *ad hoc*).

- ▶ Exemple de preuve : la symétrie $\forall x.\forall y.((x = y) \Rightarrow (y = x))$ est prouvée en appliquant la substitution à $P(s)$ valant « $s = x$ », donc par le λ -terme

$$\lambda(x : I). \lambda(y : I). \lambda(u : (x = y)). \text{subst}^{(\lambda s.s=x)} x y u (\text{refl } x)$$

- ▶ Autre exemple : la transitivité $\forall x.\forall y.\forall z.((x = y) \Rightarrow (y = z) \Rightarrow (x = z))$ par

$$\lambda(x : I). \lambda(y : I). \lambda(z : I). \lambda(u : (x = y)). \lambda(v : (y = z)). \text{subst}^{(\lambda s.x=s)} y z v u$$

L'arithmétique de Heyting et de Peano

- ▶ L'**arithmétique du premier ordre** est une (tentative d')axiomatisation des entiers naturels en logique du premier ordre. Elle est basée sur les **axiomes de Peano** (transp. suivant).
- ▶ On parle d'**arithmétique de Heyting** (HA) en logique intuitionniste, et de **Peano** (PA) en logique classique (**mêmes axiomes**, seule la logique change).
- ▶ Le cadre de base est la logique du premier ordre **avec égalité** (cf. transp. 29) et avec des opérations de formation de termes d'individus 0 (nullaire), S (unaire) et $+$, \times , Δ (binaires) :
 - ▶ 0 est un terme, ▶ si m est un terme, (Sm) en est un,
 - ▶ si m, n sont deux termes, $(m + n)$, $(m \times n)$, $(m \Delta n)$ en sont.

Ils sont censés représenter le successeur (Sn désigne $n + 1$), la somme, le produit et l'exponentiation. On omet les parenthèses comme d'habitude. On peut abrégier $1 = S0$ et $2 = S(S0)$, etc.

Les axiomes de Peano

On garde les axiomes de l'égalité :

- ▶ $\forall n.(n = n)$
- ▶ $\forall m.\forall n.((m = n) \Rightarrow P(m) \Rightarrow P(n))$ (schéma de substitution)

Les **axiomes de Peano** du premier ordre s'y ajoutent :

- ▶ $\forall n.\neg(Sn = 0)$
- ▶ $\forall m.\forall n.((Sm = Sn) \Rightarrow (m = n))$
- ▶ $P(0) \Rightarrow (\forall n.(P(n) \Rightarrow P(Sn))) \Rightarrow (\forall n.P(n))$ (schéma de **récurrence**)
- ▶ $\forall m.(m + 0 = m)$ ▶ $\forall m.\forall n.(m + (Sn) = S(m + n))$
- ▶ $\forall m.(m \times 0 = 0)$ ▶ $\forall m.\forall n.(m \times (Sn) = m \times n + m)$
- ▶ $\forall m.(m \Delta 0 = S0)$ ▶ $\forall m.\forall n.(m \Delta (Sn) = m \Delta n \times m)$

Ci-dessus, $P(s)$ désigne une formule ayant une variable d'individu libre s : la substitution de l'égalité et la récurrence sont des **schémas d'axiomes** (un axiome pour chaque P possible) car on ne peut pas quantifier sur P au premier ordre.

Exemple de preuve en arithmétique de Heyting

Montrons que $\forall n.(n = 0 \vee \neg n = 0)$ (classiquement c'est une évidence logique, mais c'est aussi démontrable intuitionistement) :

► On procède par récurrence. Notons par $P(k)$ la formule $k = 0 \vee \neg k = 0$:

► $P(0)$ vaut car $0 = 0$ vaut (réflexivité de l'égalité).

► $P(Sn)$ vaut car $\neg(Sn = 0)$ (premier axiome de Peano). En particulier, $P(n) \Rightarrow P(Sn)$.

► Donc, par récurrence, $\forall n.P(n)$, ce qu'on voulait prouver.

► Le λ -terme de cette preuve ressemble à quelque chose comme ceci :

$$\text{recurr}^{(\lambda k.(k=0 \vee \neg k=0))} (\iota_1^{(0=0, \neg 0=0)} (\text{refl } 0)) (\lambda(n : \text{nat}). \lambda(h : (n = 0) \vee \neg(n = 0)). \\ \iota_2^{(Sn=0, \neg Sn=0)} (\text{succnotz } n)).$$

Ici, « nat » a été mis pour le type des individus (entiers naturels), « succnotz » pour l'axiome de Peano qui affirme $\forall n.\neg(Sn = 0)$, et « $\text{recurr}^{(\lambda k.P(k))}$ » pour celui qui affirme $P(0) \Rightarrow (\forall n.(P(n) \Rightarrow P(Sn))) \Rightarrow (\forall n.P(n))$.

Quelques théorèmes de l'arithmétique du premier ordre

On peut prouver en arithmétique de Heyting (et notamment, de Peano) que :

- ▶ l'addition est commutative, associative, a 0 pour élément neutre...
- ▶ la multiplication est commutative, associative, a $1 = S0$ pour élément neutre...
- ▶ les identités habituelles sur l'addition, la multiplication, l'exponentiation,
- ▶ les propriétés basiques de $m \leq n$ défini par $\exists k.(n = m + k)$,
- ▶ les propriétés basiques du codage de Gödel $\langle m, n \rangle$ défini par $m + \frac{1}{2}(m + n)(m + n + 1)$,
- ▶ les propriétés basiques des suites finies codées par $\langle\langle a_0, \dots, a_{k-1} \rangle\rangle := \langle a_0, \langle a_1, \langle \dots, \langle a_{k-1}, 0 \rangle + 1 \dots \rangle + 1 \rangle + 1$,
- ▶ l'existence et l'unicité de la division euclidienne,
- ▶ des propriétés arithmétique de base : existence d'une infinité de nombres premiers, existence et unicité de la DFP, irrationalité de $\sqrt{2}$ ($\forall p.\forall q.((p \times p = 2 \times q \times q) \Rightarrow q = 0)$), etc.

Curry-Howard pour l'arithmétique de Heyting

- ▶ La formule $m = n$ est une relation binaire sur les entiers naturels : elle doit devenir un **type** (paramétré par m, n , et habité seulement lorsqu'ils sont égaux) sous l'effet de Curry-Howard.
- ▶ Il faut y penser comme le type des **témoignages d'égalité** de m et n . En pratique, ce sera un type ayant seul habitant $\{m\}$ lorsque $m = n$ et aucun sinon.
- ▶ Chaque axiome de Peano doit devenir un programme (à penser comme l'API d'une bibliothèque « entiers naturels »). Le seul non trivial est le schéma de récurrence $P(0) \Rightarrow (\forall n.(P(n) \Rightarrow P(Sn))) \Rightarrow (\forall n.P(n))$: il faut y penser comme la **primitive récursion** d'une fonction, qui à $c \in A_0$ et $f(n, —) : A_n \rightarrow A_{n+1}$ associe la suite $u_n \in \prod_{n \in \mathbb{N}} A_n$ définie par $u_0 = c$ et $u_{n+1} = f(n, u_n)$, ou en OCaml


```
let recurr = fun c -> fun f -> let rec u = fun n -> if n==0 then c else f
(n-1) (u (n-1)) in u ;;
val recurr : 'a -> (int -> 'a -> 'a) -> int -> 'a = <fun>
```

...mais avec un type qui permet à chaque u_n d'être dans un A_n différent :

$$A_0 \rightarrow (\prod_n (A_n \rightarrow A_{n+1})) \rightarrow (\prod_n A_n).$$

Curry-Howard pour l'arithmétique de Heyting (2)

À quoi ressemble le programme associé à une preuve dans l'arithmétique de Heyting ?

On peut souvent s'en faire une idée d'après son type, p.ex. :

- ▶ La commutativité de la multiplication $\forall m. \forall n. (m \times n = n \times m)$ prend m et n et renvoie un témoignage d'égalité de $m \times n$ et $n \times m$ (c'est-à-dire en fait $m \times n$ calculé de deux manières différentes).
- ▶ La preuve de $\forall n. (n = 0 \vee \neg n = 0)$ donnée transp. 32 prend en entrée n et renvoie un type somme avec soit un témoignage d'égalité de n à 0 soit un programme qui donné un tel témoignage renvoie qqch d'impossible. Donc en pratique, ce programme prend n et teste si $n = 0$.
- ▶ Une preuve de $\forall m. \exists n. Q(m, n)$ va correspondre à un programme qui prend m et renvoie n ainsi qu'un témoignage de $Q(m, n)$.

Notamment, si $\forall m. \exists n. Q(m, n)$ est prouvable dans l'arithmétique de **Heyting**, alors on peut en déduire φ_e générale récursive totale telle que $\forall m. Q(m, \varphi_e(m))$ (**extraction** de programme à partir de la preuve).

► L'arithmétique de Heyting a la **propriété de la disjonction** : si elle prouve $Q_1 \vee Q_2$, alors elle prouve Q_1 ou Q_2 .

► Et la **propriété de l'existence** : si elle prouve $\exists n.Q(n)$, alors elle prouve $Q(n)$ pour un n explicite.

Petits caractères : ces faits, comme l'extraction de programme, dépendent d'un résultat de normalisation sur l'arithmétique de Heyting (donc de $\text{Consis}(\text{HA})$).

*

► Soit P^{CPS} la formule obtenue en ajoutant « $\neg\neg$ » devant la formule tout entière, devant la conclusion de chaque \Rightarrow , et après chaque $\forall k$. Alors Heyting prouve P^{CPS} ssi Peano prouve P :

$$\text{PA} \vdash P \text{ ssi } \text{HA} \vdash P^{\text{CPS}}$$

Une différence entre Heyting et Peano

► On peut formaliser les fonctions générales récursives (ou machines de Turing) en arithmétique de Heyting. Par exemple, $\varphi_e(i)\downarrow$ signifie $\exists n.T(n, e, i)$ où T est le prédicat (p.r.) de la forme normale de Kleene, « n code un arbre de calcul valable de φ_e sur l'entrée i ».

► Si PA prouve $\forall m.\varphi_e(m)\downarrow$, alors HA le prouve (la réciproque est évidente).

► Si HA prouve $\forall m.\exists n.Q(m, n)$, alors il existe e telle que HA prouve $\forall m.\varphi_e(m)\downarrow$ et $\forall m.Q(m, \varphi_e(m))$.

► La formule

$$\forall e.\forall i.(\varphi_e(i)\downarrow \vee \neg\varphi_e(i)\downarrow)$$

(c'est-à-dire $\forall e.\forall i.((\exists n.T(n, e, i)) \vee \neg(\exists n.T(n, e, i)))$) est évidemment démontrable dans l'arithmétique de Peano (= en logique classique). Elle **n'est pas démontrable** en arithmétique de Heyting (= en logique intuitioniste), car par Curry-Howard on pourrait extraire de la preuve un algorithme résolvant le problème de l'arrêt.

Petits caractères : ces faits, comme l'extraction de programme, dépendent d'un résultat de normalisation sur l'arithmétique de Heyting (donc de Consis(HA)).

Formalisation de la démontrabilité

Idée-clé : tester **si une preuve est valable** est algorithmiquement **décidable** (même primitif récursif).

En revanche, tester si un **énoncé est un théorème** est seulement **semi-décidable** (en parcourant toutes les preuves possibles).

Plus précisément :

► On peut construire un codage de Gödel pour les formules arithmétiques et les preuves dans l'arithmétique de Heyting (ou de Peano), et notamment écrire un prédicat **primitif récursif**

$$\text{Pf}_{\text{HA}}(n, k) \quad \text{resp.} \quad \text{Pf}_{\text{PA}}(n, k)$$

qui signifie « n est le code de Gödel d'une preuve dans l'arithmétique de Heyting (resp. Peano) de la formule codée par k ».

► Notamment, $\exists n. \text{Pf}(n, k)$ peut se lire comme « k code un théorème », et l'ensemble de ces k est (au moins) semi-décidable.

Démonstrabilité de l'arrêt

Idée-clé : si une machine de Turing s'arrête, on peut démontrer (dans l'arithmétique de Heyting) qu'elle s'arrête en donnant une trace d'exécution pas à pas.

Plus précisément :

► Si $T(n, e, i)$, i.e., si n est un arbre de calcul de φ_e sur l'entrée i , alors on peut de façon algorithmique (même p.r.) tirer de n une preuve de $T(n, e, i)$ dans l'arithmétique de Heyting (i.e., un n' tel que $\text{Pf}_{\text{HA}}(n', k)$ où k est le code de Gödel de l'énoncé $\varphi_e(i) \downarrow$, i.e. $\exists n.T(n, e, i)$).

Si une machine de Turing s'arrête, alors le fait qu'elle s'arrête est prouvable
(dans l'arithmétique de Heyting, *a fortiori* de Peano).

► On notera aussi que si programme fait une boucle infinie *explicite évidente*, alors le fait qu'il ne termine pas est également prouvable.

Gödel, Rosser et Turing jouent ensemble

(Preuve du théorème de Gödel revue et corrigée par Turing et par Rosser.)

Soit g le programme suivant :

- ▶ g cherche en parallèle une preuve (dans l'arithmétique de Peano, disons) de l'énoncé « le programme g termine » (i.e. $\varphi_g(0)\downarrow$) et de l'énoncé « le programme g ne termine pas »,
- ▶ c'est-à-dire qu'il énumère les entiers et, pour chacun, teste s'il est le code de Gödel d'une preuve de $\varphi_g(0)\downarrow$ ou de $\neg\varphi_g(0)\downarrow$,
- ▶ s'il trouve (en premier) une preuve que g termine, alors il fait une boucle infinie explicite,
- ▶ s'il trouve (en premier) une preuve que g ne termine pas, alors il termine immédiatement.

Ce programme g a bien un sens, comme d'habitude, par l'« astuce de Quine » (théorème de récursion de Kleene) + le fait que la vérification des preuves est algorithmique (transp. 38).

Le théorème de Gödel

Admettons provisoirement ce qu'on notera « Consis(PA) » :

l'arithmétique de Peano ne prouve pas \perp

- ▶ Si le programme g trouve une preuve qu'il ne termine pas, alors il termine. Mais ce point donne une preuve qu'il termine (transp. 39). Donc on a une preuve de \perp dans l'arithmétique de Peano, contredisant le point ci-dessus.
- ▶ Si g trouve une preuve qu'il termine, il fait une boucle infinie. Mais ce point donne une preuve que g ne termine pas (boucle infinie explicite). Donc on a une preuve de \perp dans l'arithmétique de Peano, contredisant le point ci-dessus.

Conclusion : g ne trouve ni de preuve qu'il termine ni de preuve qu'il ne termine pas. Donc :

- ▶ g ne termine pas,
- ▶ ce fait-là n'est pas prouvable dans l'arithmétique de Peano,
- ▶ mais on l'a prouvé à l'aide de Consis(PA), donc Consis(PA) lui-même n'est pas prouvable dans Peano (toujours en supposant Consis(PA)).

Cohérence de Peano

► L'énoncé $\text{Consis}(\text{PA})$ peut se lire ainsi :

« Le programme g' qui parcourt les entiers et, pour chacun, teste s'il est le code de Gödel d'une preuve de \perp dans l'arithmétique de Peano et dans ce cas termine, ne termine pas. »

Cet énoncé **a un sens** dans l'arithmétique du premier ordre, mais (on vient de le voir) **n'est pas démontrable** s'il est vrai.

► L'énoncé $\text{Consis}(\text{HA})$ analogue pour l'arithmétique de Heyting **est équivalent** à $\text{Consis}(\text{PA})$ par la traduction CPS (et cette équivalence est prouvable dans HA).

Notamment, Peano ne prouve pas non plus $\text{Consis}(\text{HA})$. Par propriété de la disjonction, HA ne prouve même pas $\text{Consis}(\text{HA}) \vee \neg \text{Consis}(\text{HA})$.

► En revanche, ZFC (le cadre usuel pour faire des mathématiques) démontre $\text{Consis}(\text{PA})$: « Les axiomes de Peano sont vrais dans \mathbb{N} donc leurs conséquences le sont aussi, et notamment \perp ne peut pas en faire partie. (Et au passage, si PA démontre $\varphi_e(i) \downarrow$, alors $\varphi_e(i) \downarrow$ est vrai.) »

Donc ZFC est strictement plus fort que PA (même pour l'arithmétique).

Le théorème de Gödel généralisé

Pour n'importe quelle sorte de « théorie logique » (classique ou intuitioniste, pas limitée au premier ordre) T telle que :

- ▶ les énoncés et démonstrations sont codables par des entiers naturels,
- ▶ on peut algorithmiquement tester si un entier naturel code une démonstration valable dans T , et quelle est sa conclusion,
- ▶ T permet de formaliser « $\varphi_e(i) \downarrow$ » (calculablement en e et i),
- ▶ si $\varphi_e(i) \downarrow$ alors on peut tirer d'une trace d'exécution une preuve de ce fait dans T , et idem pour une boucle infinie explicite,

on peut construire le programme g_T qui cherche en parallèle dans T une preuve de que g_T termine ou ne termine pas, et fait une boucle infinie explicite dans le premier cas, termine immédiatement dans le second.

- ▶ Si T ne prouve pas \perp (hypothèse notée « $\text{Consis}(T)$ »), alors g_T ne termine pas, mais T ne peut pas le prouver. Notamment, T ne prouve pas $\text{Consis}(T)$ (toujours si $\text{Consis}(T)$). Ceci s'applique notamment à Coq, à ZFC, etc.

L'ensemble des théorèmes est semi-décidable non décidable

Avec T comme au transparent précédent (et sous l'hypothèse $\text{Consis}(T)$), par exemple PA supposons par l'absurde qu'on puisse décider algorithmiquement si une formule P est un théorème de T .

Soit g'' le programme qui :

- ▶ teste si $\varphi_{g''}(0)\downarrow = 0$ est un théorème de T (grâce à l'hypothèse effectuée),
 - ▶ si oui, termine et renvoie 1,
 - ▶ si non, termine et renvoie 0.
- ▶ Par construction, g'' termine forcément et renvoie soit 0 soit 1. Si g'' termine et renvoie 0, alors T le prouve, donc g'' termine et renvoie 1, contradiction ; si g'' termine et renvoie 1, alors T le prouve, donc (par $\text{Consis}(T)$) il ne prouve pas que g'' termine et renvoie 0, donc g'' termine et renvoie 0, contradiction.