Jean Goubault-Larrecq

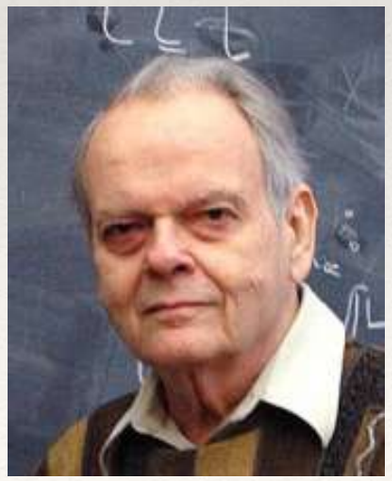
λ-calcul

10. Système F, polymorphisme et arithmétique du second ordre

Le système F

- Aujourd'hui: quantification ∀ du **second ordre**
- C'est le **système F (ou sa variante F_2)** inventé en logique (Girard, 1971) pour donner une preuve « finitiste » de la cohérence de **PA_2**
- et en informatique (Reynolds, 1974) pour donner un modèle du **polymorphisme** (préfigurant ML et Haskell)

Jean-Yves Girard
(les candidats de réductibilité, c'est lui)
http://ekouter.net/img/img/JeanYvesGirard.jpg



John C. Reynolds
https://www.cs.cmu.edu/sites/default/files/styles/news_item_image/public/john-reynolds.jpg?itok=FGDq9ABS

Système F

Formules

- Ça ressemble à la logique du premier ordre, en plus simple (pas d'expressions)

- **Formules atomiques** α = variables de formules
  (ou constantes, il suffit de ne pas quantifier dessus)

- Formules $F, G, \ldots ::= \alpha$ (formules atomiques)
  $\mid F \Rightarrow G$ (comme en logique prop.)
  $\mid \forall \alpha \,.\, F$ (nouveau)

Système F_2

- Types (formules):

  $F, G, \ldots ::= \alpha$

  $\mid F \Rightarrow G$

  $\mid \forall \alpha \,.\, F$

- Termes:

  $u, v, \ldots ::= x$

  $\mid uv$

  $\mid \lambda x.u$

  $\mid uG$

  $\mid \Lambda \alpha \,.\, u$

- Réduction:

  $(\beta) \quad (\lambda x.u)v \to u[x:=v]$

  $(B^2) \quad (\Lambda \alpha \,.\, u)G \to u[\alpha:=G]$

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax}) \qquad \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall \alpha \,.\, F}{\Gamma \vdash uG : F[\alpha:=G]}\,(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda \alpha \,.\, u : \forall \alpha \,.\, F}\,(\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

Système ~~F_2~~ F

- Types (formules):

  $F, G, \ldots ::= \alpha$

  $\mid F \Rightarrow G$

  $\mid \forall \alpha \,.\, F$

- Termes:

  $u, v, \ldots ::= x$

  $\mid uv$

  $\mid \lambda x.u$

  ~~$\mid uG$~~

  ~~$\mid \Lambda \alpha \,.\, u$~~

- Réduction:

  $(\beta) \quad (\lambda x.u)v \to u[x:=v]$

  ~~$(B^2) \quad (\Lambda \alpha \,.\, u)G \to u[\alpha:=G]$~~

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall \alpha \,.\, F}{\Gamma \vdash \underset{u}{\cancel{uG}} : F[\alpha:=G]}\,(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \underset{u}{\cancel{\Lambda \alpha \,.\, u}} : \forall \alpha \,.\, F}\,(\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

Système F

- Types (formules):
  $F, G, \ldots ::= \alpha \mid F \Rightarrow G \mid \forall \alpha \,.\, F$
- Termes:
  $u, v, \ldots ::= x \mid uv \mid \lambda x.u$
- Réduction:
  $(\beta)\quad (\lambda x.u)v \to u[x:=v]$

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall \alpha \,.\, F}{\Gamma \vdash u : F[\alpha:=G]}\,(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha \,.\, F}\,(\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

Système F = polymorphisme

- En système F, on a:
  $\lambda x.x : \forall\alpha\ .\ \alpha \Rightarrow \alpha$
  cons : $\forall\alpha\ .\ \alpha \Rightarrow$ **list**(α) =
- on peut écrire: **cons** (1,
  et aussi: **cons** ((1
- et aussi **id id** (où **id** = $\lambda x.x$) : $\forall\alpha\ .\ \alpha \Rightarrow \alpha$
- et aussi **cons** (**id**, **cons**(**id id**, **nil**)) : **list**($\forall\alpha\ .\ \alpha \Rightarrow \alpha$)

$$\frac{}{\Gamma,\ x{:}F \vdash x : F}\ (\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\ (\Rightarrow\mathrm{E}) \qquad \frac{\Gamma,\ x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\ (\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall\alpha\ .\ F}{\Gamma \vdash u : F[\alpha{:=}G]}\ (\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall\alpha\ .\ F}\ (\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

Comme en ML ou en Haskell

Impossible en ML ou en Haskell!

Système F_2 = polymorphisme, en plus lourd

$$\frac{}{\Gamma, x{:}F \vdash x : F}(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall\alpha\,.\,F}{\Gamma \vdash uG : F[\alpha{:=}G]}(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha\,.\,u : \forall\alpha\,.\,F}(\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

- En système F_2, on a:
  $\Lambda\alpha\,.\,\lambda x.x : \forall\alpha\,.\,\alpha \Rightarrow \alpha$
  cons : $\forall\alpha\,.\,\alpha \Rightarrow$ **list**$(\alpha) \Rightarrow$ **list**(α)

- on peut écrire: **cons** nat (1, **cons** nat (2, **nil** nat)) : **list**(**nat**)
  et aussi: **cons** (nat*nat) ((1,2), **cons** (nat*nat) ((3,4),
  nil (nat*nat))) : **list**(**nat***nat)

- aussi $\Lambda\alpha\,.$ **id** $(\alpha \Rightarrow \alpha)$ (**id** α) (où **id**$=\Lambda\alpha\,.\,\lambda x.x$) : $\forall\alpha\,.\,\alpha \Rightarrow \alpha$

- et aussi **cons** $(\forall\alpha\,.\,\alpha \Rightarrow \alpha)$ (**id**,
  cons $(\forall\alpha\,.\,\alpha \Rightarrow \alpha)$ ($\Lambda\alpha\,.$ **id** $(\alpha \Rightarrow \alpha)$ (**id** α),
  nil $(\forall\alpha\,.\,\alpha \Rightarrow \alpha)$)) : **list**$(\forall\alpha\,.\,\alpha \Rightarrow \alpha)$

F vs. F_2

- En programmation, on préférera souvent F
- En logique, F_2 est plus explicite…
- Je vais traiter de F_2, et je vous laisse vérifier que (presque) tout est pareil pour F

Propriétés fondamentales

- **Expressivité**…
  très forte
  (voir transparents suivants)

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall\alpha\,.\,F}{\Gamma \vdash u : F[\alpha{:=}G]}\,(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall\alpha\,.\,F}\,(\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

$(\beta)\quad (\lambda x.u)v \to u[x{:=}v]$
$(B)\quad (\Lambda\alpha\,.\,u)G \to u[\alpha{:=}G]$

- **Autoréduction:** si $\Gamma \vdash u : F$ est dérivable
  et $u \twoheadrightarrow v$ alors $\Gamma \vdash v : F$ est dérivable.

- **Normalisation forte:** si $\Gamma \vdash u : F$ est dérivable,
  alors u est fortement normalisable.

Pouvoir expressif

Le « et » en système F_2

- On peut **définir** $F \wedge G = \forall \alpha \,.\, (F \Rightarrow G \Rightarrow \alpha) \Rightarrow \alpha$ (α fraîche)

- Intuitivement,

  $F \wedge G$ est le **plus petit** type impliqué par F et G

  (voyez $\forall$ comme une sorte d'intersection)

Le « et » en système F_2

- On peut **définir** $F \wedge G = \forall\alpha\ .\ (F \Rightarrow G \Rightarrow \alpha) \Rightarrow \alpha$ (α fraîche)
- $\langle u,v\rangle = \lambda x.xuv$ (eh oui, c'est la paire de Church!)

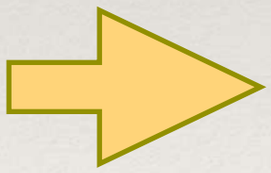
  $\pi_1 u = u(\lambda x,y.x)$

  $\pi_2 u = u(\lambda x,y.y)$

A vous de le découvrir en TD!

- $\pi_1 \langle u,v\rangle \to^+ u$

  $\pi_2 \langle u,v\rangle \to^+ v$



	Monde de la logique	Monde des programmes
F	formule	type
u	preuve	λ-terme (programme)
$u \to^* v$	élimination des détours	β-réduction (exécution)

Le « ou » en système F_2

- On peut **définir** $F \vee G = \forall\alpha\ .\ (F \Rightarrow \alpha) \Rightarrow (G \Rightarrow \alpha) \Rightarrow \alpha$
  (α fraîche)

- Intuitivement,

  $F \vee G$ est le **plus petit** type impliqué par F
  et impliqué par G

  (voyez $\forall$ comme une sorte d'intersection)

Le « ou » en système F_2

- On peut **définir** $\forall\alpha\,.\,(F \Rightarrow \alpha) \Rightarrow (G \Rightarrow \alpha) \Rightarrow \alpha$ (α fraîche)
- $\iota_1\, u = \Lambda\alpha\,.\,\lambda x,y.xu$

  $\iota_2\, u = \Lambda\alpha\,.\,\lambda x,y.yu$

  $\texttt{case}\ u\ \texttt{of}\ \iota_1\, x_1 \mapsto v_1 \mid \iota_2\, x_2 \mapsto v_2 \equiv u\,(\lambda x_1.v_1)(\lambda x_2.v_2)$

A vous de le découvrir en TD!

- $\texttt{case}\ \iota_1\, u\ \texttt{of}\ \iota_1\, x_1 \mapsto v_1 \mid \iota_2\, x_2 \mapsto v_2$

  $\to^+ v_1[x_1{:=}u]$

  $\texttt{case}\ \iota_2\, u\ \texttt{of}\ \iota_1\, x_1 \mapsto v_1 \mid \iota_2\, x_2 \mapsto v_2$

  $\to^+ v_2[x_2{:=}u]$

	Monde de la logique	Monde des programmes
F	formule	type
u	preuve	λ-terme (programme)
$u \to^* v$	élimination des détours	β-réduction (exécution)

Le « faux » en système F_2

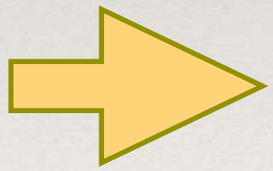
- On peut **définir** $\bot = \forall\alpha\ .\ \alpha$
- Intuitivement,

  $\bot$ est le **plus petit** type (tout court)

  (voyez $\forall$ comme une sorte d'intersection)

Le « faux » en système F_2

- On peut **définir** $\bot = \forall\alpha\,.\,\alpha$
- $\nabla u = uF$
- ~~$\nabla uv \to^+ \nabla u$~~
- … ah non, raté!



	Monde de la logique	Monde des programmes
F	formule	type
u	preuve	λ-terme (programme)
$u \to^* v$	élimination des détours	β-réduction (exécution)

Le quantificateur existentiel

- Initialement, J.-Y. Girard avait inclus une construction

$$\exists \alpha \,.\, F$$

avec les règles:

Jean-Yves Girard

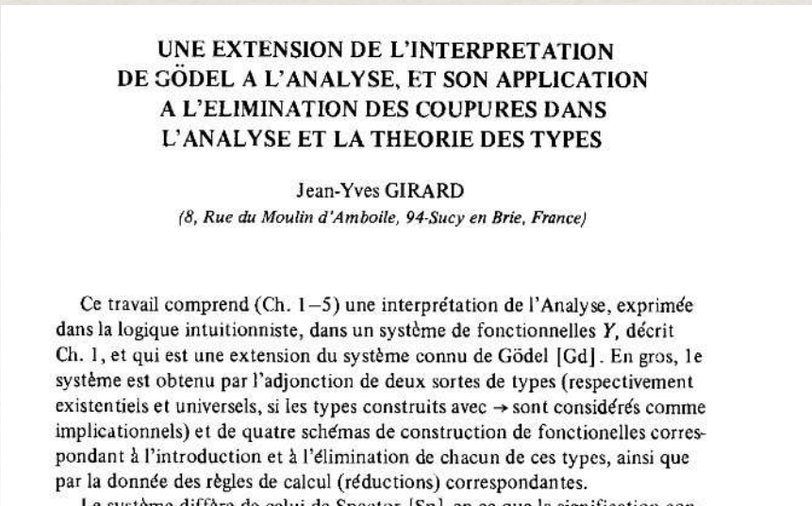
http://ekouter.net/img/img/JeanYvesGirard.jpg

$$\frac{\Gamma \vdash u : \exists \alpha \,.\, F \quad \Gamma, x{:}F \vdash v : H}{\Gamma \vdash \texttt{case}\ u\ \texttt{of}\ \iota(\alpha, x) \mapsto v : H}\ (\exists^2\mathrm{E})$$

(si α pas libre dans [aucune des formules de] Γ, H)

$$\frac{\Gamma \vdash u : F[\alpha{:=}G]}{\Gamma \vdash \iota(G, u) : \exists \alpha \,.\, F}\ (\exists^2\mathrm{I})$$

- Ce n'est pas la peine!



UNE EXTENSION DE L'INTERPRETATION
DE GÖDEL A L'ANALYSE, ET SON APPLICATION
A L'ELIMINATION DES COUPURES DANS
L'ANALYSE ET LA THEORIE DES TYPES

Jean-Yves GIRARD
(8, Rue du Moulin d'Amboile, 94-Sucy en Brie, France)

Ce travail comprend (Ch. 1–5) une interprétation de l'Analyse, exprimée dans la logique intuitionniste, dans un système de fonctionnelles *Y*, décrit Ch. 1, et qui est une extension du système connu de Gödel [Gd]. En gros, le système est obtenu par l'adjonction de deux sortes de types (respectivement existentiels et universels, si les types construits avec → sont considérés comme implicationnels) et de quatre schémas de construction de fonctionelles correspondant à l'introduction et à l'élimination de chacun de ces types, ainsi que par la donnée des règles de calcul (réductions) correspondantes.

Le « $\exists$ » en système F_2

- On peut **définir** $\exists\,\alpha\,.\,F = \forall\beta\,.\,(\forall\alpha\,.\,F \Rightarrow \beta) \Rightarrow \beta$ (β fraîche)

- Intuitivement,

  $\exists\,\alpha\,.\,F$ est le **plus petit** type H (ou β) tel que
  si (pour tout α) F implique H, alors H

$$\frac{\Gamma \vdash \exists\alpha\,.\,F \qquad \Gamma, F \vdash H}{\Gamma \vdash H}\ (\exists^2\mathrm{E})$$

(si α pas libre dans [aucune des formules de] Γ, H)

Le « ∃ » en système F_2

- On peut **définir** ∃

C'est un « couple » (G, u)…
et le type réel **d'implémentation**
G est caché de l'extérieur

A vous de le découvrir en TD!

- $\iota(G, u) = \Lambda\beta\ .\ \lambda x.xGu$
  `case` u `of` $\iota(\alpha, x) \mapsto v = u$

- `case` $\iota(G, u)$ `of` $\iota(\alpha, x) \mapsto v$
  $\to^+ v[\alpha:=G, x:=u]$

	Monde de la logique	Monde des programmes
F	formule	type
u	preuve	λ-terme (programme)
$u \to^* v$	élimination des détours	β-réduction (exécution)

Types abstraits

```
datatype formule = …
abstype thm = T of formule
with
  fun and_intro (T F,T G:thm)
      = T (F /\ G)
  fun and_elim1 (T (F /\ G):thm)
      = T F
  …
end
```

thm = $\exists\alpha$.

$(\alpha \Rightarrow \alpha \Rightarrow \alpha) \wedge$ (* and_intro *)

$(\alpha \Rightarrow \alpha) \wedge$ (* and_elim1 *)

…

ι(`formule`, <$\lambda x,y.x \wedge y$, …>) : `thm`

Comme en LCF, notion de **type abstrait:**

Théorème. Toute valeur de type `thm` est de la forme `T F` où `F` est **démontrable**.

Types de données

- Encore mieux: on peut **définir** les types de données habituels (**nat**, **list**(*F*), etc.) en système F
- … mais pas tous ceux de ML ou de Haskell quand même (on verra pourquoi après)

Le type **nat** des entiers

« type de s » : $\alpha \Rightarrow \alpha$

- On définit **nat** = $\forall\alpha\,.\,(\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$

« type de 0 » : α

- L'entier de Church $\lceil n \rceil = \Lambda\alpha\,.\,\lambda f,x.f^n(x)$: **nat**

- Le **récurseur**: si u:F, v:**nat** $\Rightarrow F \Rightarrow F$, et w:**nat**,

$$\mathtt{R}uvw = \pi_1\, F\ (w\ (F{*}\mathbf{nat})$$
$$(\lambda z.\ \langle v(\pi_2\,\mathbf{nat}\ z)(\pi_1\, F\ z), \mathtt{s}(\pi_2\,\mathbf{nat}\ z)\rangle)$$
$$\langle u, \lceil 0 \rceil \rangle) : F$$

(~formule du prédécesseur)

- $\mathtt{R}uv\lceil 0 \rceil \rightarrow^+ u$ $\qquad$ $\mathtt{R}uv\lceil n{+}1 \rceil =_\beta v\lceil n \rceil(\mathtt{R}uv\lceil n \rceil)$

(voir exercice 39 du poly `types.pdf`)

Le type **list**(F) des listes de F

« type de **cons** » : $F \Rightarrow \alpha \Rightarrow \alpha$

- On définit $\mathbf{list}(F) = \forall\alpha\ .\ (F \Rightarrow \alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$

« type de **nil** » : α

- $\mathbf{nil} = \Lambda\alpha\ .\ \lambda c,n.n : \mathbf{list}(F)$

  $\mathbf{cons} = \lambda\ a, l\ .\ \Lambda\alpha\ .\ \lambda c,n.\ c\ a\ (l\ \alpha\ c\ n) : F \Rightarrow \mathbf{list}(F) \Rightarrow \mathbf{list}(F)$

- Le **récurseur** sur les listes… exercice!

(analogue à l'exercice 39 du poly `types.pdf`)

Un type **pas** de données

- `type void = V of (void -> void)`
- `let i (f:void->void) : void = V f`
- ```
  let r (v:void) : (void -> void)
    = match v with
          V f -> f
  ```
- Note: « `r` o `i` est l'identité » (sémantiquement)
  Ca vous dit quelque chose?
  ```

Un type **pas** de données

- `type void = V of (void -> void)`
- `let i (f:void->void) : void = V f`
- ```
  let r (v:void) : (void -> void)
    = match v with
          V f -> f
  ```
- Permet un codage de tout le λ-calcul (non typé) en Caml… sans `let rec`!
  ```

Un type **pas** de données

- ```
  type void = V of (void -> void)
  ```
- ```
  let i (f:void->void) : void = V f
  ```
- ```
  let r (v:void) : (void -> void)
    = match v with
          V f -> f
  ```
- ```
  let delta = i (fun x:void -> r x x)
  let omega = r delta delta
  ```
  … ne termine pas (or tout terme du système F termine)

Normalisation forte

Effacement, comme au premier ordre?

- Rapp
- Fonc
  enlèv
- $E(P(e_1, \ldots, e_n)) = P \ldots E(\forall \ldots \Rightarrow G) = E(F) \Rightarrow E(G)$

  $E(\Lambda i \,.\, u) = E(u)$

  $E(uv) = E(u)E(v) \quad E(\lambda x.u) = \lambda x.E(u) \quad E(x) = x$

- **Observation**

  Si $u \to v$ par (β), alors $E(u) = E(v)$ par (β)

  Si $u \to v$ par (B), alors $E(u) = E(v)$

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall \alpha \,.\, F}{\Gamma \vdash uG : F[\alpha{:=}G]}\,(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha \,.\, u : \forall\alpha \,.\, F}\,(\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall i \,.\, G}{\Gamma \vdash ue : G[i{:=}e]}\,(\forall\mathrm{E}) \qquad \frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i \,.\, u : \forall i \,.\, G}\,(\forall\mathrm{I})$$

(si i pas libre dans [aucune des formules de] Γ)

$$(\beta) \quad (\lambda x.u)v \to u[x{:=}v]$$
$$(B) \quad (\Lambda\alpha \,.\, u)G \to u[\alpha{:=}G]$$

$$(\beta) \quad (\lambda x.u)v \to u[x{:=}v]$$
$$(B) \quad (\Lambda i \,.\, u)e \to u[i{:=}e]$$

Mais que voudriez-vous effacer ici?
Si vous effacez G, vous effacez tout…

Candidats de réductibilité

- … candidats de réductibilité, comme en $\mathbf{HA}_1$
- $\mathrm{RED}_\alpha = \mathrm{SN}$, $\mathrm{RED}_{F\Rightarrow G} = \mathrm{RED}_F \Rightarrow \mathrm{RED}_G$
  $\mathrm{RED}_{\forall\alpha\,.\,F} = \{u \mid \text{pour tout } G,\ uG \in \mathrm{RED}_{F[\alpha:=G]}\}$
- Juste un petit souci: pourquoi ceci est-il une définition valide? (par récurrence sur quoi?)
- … ben non, ça n'est pas valide, et c'est **irréparable**
- Il va falloir une nouvelle idée!

Candidats de réductibilité

- … candidats de réductibilité, comme en $\mathbf{HA}_1$
- $\mathrm{RED}_\alpha = \mathrm{SN}$, $\mathrm{RED}_{F \Rightarrow G} = \mathrm{RED}_F \Rightarrow \mathrm{RED}_G$
  $\mathrm{RED}_{\forall \alpha \,.\, F} = \{u \mid \text{pour tout } G,\ uG \in \mathrm{RED}_{F[\alpha := G]}\}$
- Prenons le type « faux » $\forall \alpha \,.\, \alpha$:
  la définition de $\mathrm{RED}_{\forall \alpha \,.\, \alpha}$ dépend de RED_G
  pour **tous** les types G (y compris « faux » lui-même)
- Aucun ordre bien fondé n'existe qui permette de faire de ça une définition valide!

Sémantique …

- Vous vous souvenez de mon explication des candidats de réductibilité comme valeurs de vérité particulières?
- J'avais écrit ⟦F⟧ au lieu de RED$_F$
- Ce q… de donner une sém…
- $[\![\alpha]\!] = \mathrm{SN}$

  $[\![F \Rightarrow G]\!] = [\![F]\!] \Rightarrow [\![G]\!]$

  $[\![\forall \alpha \,.\, F]\!] = \{u \mid \text{pour tout } G,\ uG \in [\![F[\alpha := G]]\!]\}$

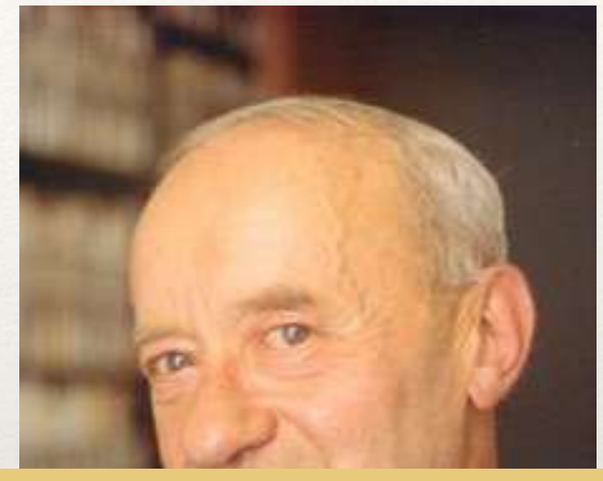
Mais ça, c'est bizarre, non?

Ça n'a aucun sens…

Sémantique à la Tarski…

- Dans une sémantique à la Tarski,
  on se donn**ement**
  ϱ tel que ϱ et pas SN (« vrai »)! ur de vérité
  pour toute α

- $[\![\alpha]\!]\varrho = \varrho(\alpha)$
  $[\![F \Rightarrow G]\!]\varrho = [\![F]\!]\varrho \Rightarrow [\![G]\!]\varrho$
  $[\![\forall \alpha \,.\, F]\!]\varrho =$ infimum (« et ») des $[\![F]\!](\varrho[\alpha \mapsto S])$,
  lorsque S parcourt les valeurs de vérité



Eh, mais, ça, c'est une définition valide par récurrence sur la formule!

Sémantique à la Tarski-Girard

- $[\![\alpha]\!]\varrho = \varrho(\alpha)$

$[\![F \Rightarrow G]\!]\varrho = [\![F]\!]\varrho \Rightarrow [\![G]\!]\varrho$

$[\![\forall \alpha \,.\, F]\!]\varrho = \{u \mid$ pour tout G, pour tout candidat S,

$u G \in [\![F]\!](\varrho[\alpha \mapsto S])\}$

En notation « RED »

- Les **environnements** ϱ s'appellent
  des **contextes de candidats** $\mathcal{C}$ (de réductibilité)

- $\mathrm{RED}_{\alpha}{}^{\mathcal{C}} = \mathcal{C}(\alpha)$

  $\mathrm{RED}_{F \Rightarrow G}{}^{\mathcal{C}} = \mathrm{RED}_{F}{}^{\mathcal{C}} \Rightarrow \mathrm{RED}_{G}{}^{\mathcal{C}}$

  $\mathrm{RED}_{\forall \alpha \,.\, F}{}^{\mathcal{C}} = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S,\ uG \in \mathrm{RED}_{F}{}^{\mathcal{C}[\alpha \mapsto S]}\}$

- (Comparez:

  $[\![\alpha]\!]\varrho = \varrho(\alpha)$

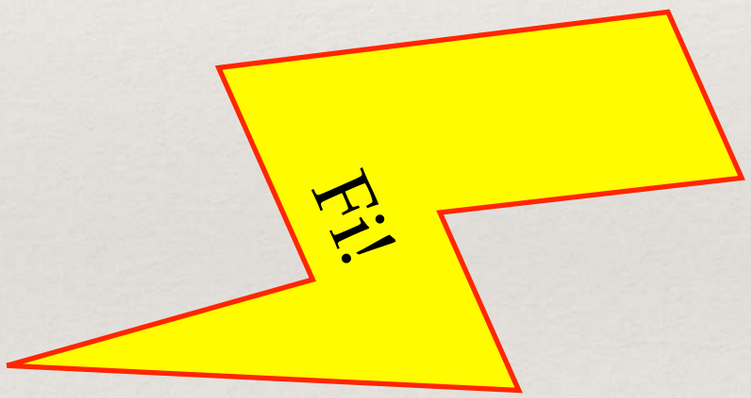
  $[\![F \Rightarrow G]\!]\varrho = [\![F]\!]\varrho \Rightarrow [\![G]\!]\varrho$

  $[\![\forall \alpha \,.\, F]\!]\varrho = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S,\ uG \in [\![F]\!](\varrho[\alpha \mapsto S])\})$

L’ironie du sort

- Bien entendu, c’est Girard, pas Tarski, qui a inventé ça.

By George Bergman - The Oberwolfach photo collection, https://opc.mfo.de/detail?photo_id=6091, GFDL 1.2, https://commons.wikimedia.org/w/index.php?curid=7981479



http://ekouter.net/img/img/JeanYvesGirard.jpg

L'opérateur Π

- $\mathrm{RED}_{\forall\alpha\,.\,F^{\mathcal{C}}} = \{u \mid$ pour tout G, pour tout candidat S,
$uG \in \mathrm{RED}_{F^{\mathcal{C}[\alpha\mapsto S]}}\}$

- Autrement dit,

$$\mathrm{RED}_{\forall\alpha\,.\,F^{\mathcal{C}}} = \Pi(S \mapsto \mathrm{RED}_{F^{\mathcal{C}[\alpha\mapsto S]}}),$$

- où pour toute fonction f : candidats $\to$ candidats,
$\Pi(f) = \{u \mid$ pour tout G, pour tout candidat S,
$uG \in f(S)\}$

(CR1)

- $\Pi(f) = \{u \mid$ pour tout G, pour tout candidat S, $uG \in f(S)\}$
- (CR1): pour toute f : candidats → candidats,
  $\Pi(f) \subseteq$ SN
- En effet, pour tout $u \in \Pi(f)$,
  choisissons un type G quelconque (il en existe)
  posons S = SN — car SN est un candidat
  alors $uG \in f(S)$
- Comme $f(S)$ candidat, $uG \in$ SN par (CR1) sur $f(S)$.
  Donc $u \in$ SN. □

(CR2)

- $\Pi(f) = \{u \mid$ pour tout G, pour tout candidat S, $uG \in f(S)\}$
- (CR2): Si $u \in \Pi(f)$ et $u \to u'$ alors $u' \in \Pi(f)$.

- Pour tout type G, pour tout candidat S, $uG \in f(S)$
- Or $uG \to u'G$
  et $f(S)$ candidat, donc $u'G \in f(S)$ par (CR2) sur $f(S)$
- Donc $u' \in \Pi(f)$. □

(CR3)

- $\Pi(f) = \{u \mid$ pour tout G, pour tout candidat S, $uG \in f(S)\}$
- (CR3): Si u **neutre** et tous ses réduits en une étape u' sont dans $\Pi(f)$, alors $u \in \Pi(f)$.

  (neutre=ne commençant ni par λ ni par Λ)
- Pour tout type G, pour tout candidat S,
  les réduits en une étape de uG sont les $u'G$
- … qui sont tous dans $f(S)$ par hypothèse
- Or uG est neutre, et $f(S)$ candidat, donc $uG \in f(S)$ par (CR3) sur $f(S)$
- Donc $u \in \Pi(f)$. □

Candidats de réductibilité

- $\mathrm{RED}_{\alpha}{}^{\mathcal{C}} = \mathcal{C}(\alpha)$

  $\mathrm{RED}_{F \Rightarrow G}{}^{\mathcal{C}} = \mathrm{RED}_{F}{}^{\mathcal{C}} \Rightarrow \mathrm{RED}_{G}{}^{\mathcal{C}}$

  $\mathrm{RED}_{\forall \alpha \,.\, F}{}^{\mathcal{C}} = \Pi(S \mapsto \mathrm{RED}_{F}{}^{\mathcal{C}[\alpha \mapsto S]})$

  $(\Pi(f) = \{u \mid$ pour tout G, pour tout candidat S, $uG \in f(S)\})$

- Donc, par récurrence sur le type F:

Lemme B. Pour tout type F,
pour tout contexte de candidats $\mathcal{C}$,
$\mathrm{RED}_{F}{}^{\mathcal{C}}$ est un candidat.

Lemmes utiles (1)

- $\mathrm{RED}_{\alpha}{}^{\mathcal{C}} = \mathcal{C}(\alpha)$

  $\mathrm{RED}_{F \Rightarrow G}{}^{\mathcal{C}} = \mathrm{RED}_{F}{}^{\mathcal{C}} \Rightarrow \mathrm{RED}_{G}{}^{\mathcal{C}}$

  $\mathrm{RED}_{\forall \alpha \,.\, F}{}^{\mathcal{C}} = \Pi(S \mapsto \mathrm{RED}_{F}{}^{\mathcal{C}[\alpha \mapsto S]})$

  $(\Pi(f) = \{u \mid$ pour tout G, pour tout candidat S, $uG \in f(S)\})$

- Comme avant, mais pour tout contexte de candidats $\mathcal{C}$:

- **Lemme C.** Si (pour tout $v \in \mathrm{RED}_{F}{}^{\mathcal{C}}$, $s[x{:=}v] \in \mathrm{RED}_{G}{}^{\mathcal{C}}$)
  alors $\lambda x.s \in \mathrm{RED}_{F \Rightarrow G}{}^{\mathcal{C}}$

Lemmes utiles (2)

- $\mathrm{RED}_{\forall\alpha\,.\,F^{\mathcal{C}}} = \Pi(S \mapsto \mathrm{RED}_{F^{\mathcal{C}[\alpha\mapsto S]}})$

  $(\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S,\ uG \in f(S)\})$

- **Lemme C′.** Si (pour tout G et pour tout candidat S, $u[\alpha{:=}G] \in \mathrm{RED}_{F^{\mathcal{C}[\alpha \mapsto S]}}$)
  alors $\Lambda\alpha\,.\,u \in \mathrm{RED}_{\forall\alpha\,.\,F^{\mathcal{C}}}$

- Pour tous G et S, on regarde les réduits en une étape de $(\Lambda\alpha\,.\,u)G$… (en système F_2 — en F, c'est une tautologie)

- et on applique (CR3) + récurrence sur u le long de $\to$. □

Quelques lemmes triviaux

By George Bergman - The Oberwolfach photo collection, https://opc.mfo.de/detail?photo_id=6091, GFDL 1.2, https://commons.wikimedia.org/w/index.php?curid=7981479

- Dans une sémantique à la Tarski, on a toujours:
- **(Substitution.)** $[\![F[\alpha:=G]]\!]\varrho = [\![F]\!](\varrho[\alpha \mapsto [\![G]\!]\varrho])$
- **(Liberté.)** Si α pas libre dans F, $[\![F]\!](\varrho[\alpha \mapsto S]) = [\![F]\!]\varrho$

(Récurrences triviales sur F.)

Quelques lemmes utiles

- De même, ici nous avons:

- **Lemme S.** $\mathrm{RED}_{F[\alpha:=G]}^{\mathcal{C}} = \mathrm{RED}_{F}^{\mathcal{C}[\alpha \mapsto S]}$ où $S{=}\mathrm{RED}_{G}^{\mathcal{C}}$

- **Lemme L.** Si α pas libre dans F, $\mathrm{RED}_{F}^{\mathcal{C}[\alpha \mapsto S]} = \mathrm{RED}_{F}^{\mathcal{C}}$

(Récurrences triviales sur F.)

Normalisation forte

- **Def:** $\theta \in \underline{\mathrm{RED}}_\Gamma{}^{\mathcal{C}}$ ssi pour tout $x{:}F$ dans Γ, $\theta(x) \in \mathrm{RED}_F{}^{\mathcal{C}}$

  + $\theta(\alpha)$ est un type pour tout α (en F_2, pas F)

- **Thm.** si $\Gamma \vdash u : G$ dérivable alors pour tout contexte $\mathcal{C}$,
  pour toute $\theta \in \underline{\mathrm{RED}}_\Gamma{}^{\mathcal{C}}$, $u\theta \in \mathrm{RED}_G{}^{\mathcal{C}}$.

- Preuve: par récurrence sur la dérivation de $\Gamma \vdash u : G$,
  comme les dernières fois;
  on utilise la déf. de $\mathrm{RED}_{F\Rightarrow G}{}^{\mathcal{C}}$ dans le cas de ($\Rightarrow$E),
  le Lemme C pour ($\Rightarrow$I), etc.
  Deux cas nouveaux… (prochains transparents)

Normalisation forte

- **Thm.** si $\Gamma \vdash u : G$ dérivable alors pour tout contexte $\mathcal{C}$,
  pour toute $\theta \in \underline{\mathrm{RED}}_\Gamma{}^{\mathcal{C}}$, $u\theta \in \mathrm{RED}_G{}^{\mathcal{C}}$.

- **Cas ($\forall^2$E).** Par h.r., $u\theta \in \mathrm{RED}_{\forall\alpha\,.\,F}{}^{\mathcal{C}}$.

$$\frac{\Gamma \vdash u : \forall\alpha\,.\,F}{\Gamma \vdash uG : F[\alpha:=G]}\ (\forall^2\mathrm{E})$$

- Par déf., $u\theta G \in \mathrm{RED}_F{}^{\mathcal{C}[\alpha \mapsto S]}$
  où $S = \mathrm{RED}_G{}^{\mathcal{C}}$

$$\mathrm{RED}_{\forall\alpha\,.\,F}{}^{\mathcal{C}} = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S,\ uG \in \mathrm{RED}_F{}^{\mathcal{C}[\alpha\mapsto S]}\}$$

- Donc $u\theta G = (uG)\theta \in \mathrm{RED}_{F[\alpha:=G]}{}^{\mathcal{C}}$

Lemme S. $\mathrm{RED}_{F[\alpha:=G]}{}^{\mathcal{C}} = \mathrm{RED}_F{}^{\mathcal{C}[\alpha \mapsto S]}$ où $S = \mathrm{RED}_G{}^{\mathcal{C}}$

Normalisation forte

- **Thm.** si Γ r tout contexte $\mathcal{C}$, pour $\mathrm{RED}_G{}^{\mathcal{C}}$.

Je vous laisse vérifier les conditions sur les variables libres et liées…

- **Cas ($\forall^2$I).** ar h.r., (pour tous $\mathcal{C}$, G, S), pour tout contexte de la forme $\mathcal{C}[\alpha \mapsto S]$, $u(\theta+[\alpha:=G]) \in \mathrm{RED}_F{}^{\mathcal{C}[\alpha \mapsto S]}$.

$$\frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha\,.\,u : \forall\alpha\,.\,F}\ (\forall^2\mathrm{I})$$

(si α pas libre dans [aucune des formules de] Γ)

- I.e., pour tout $\mathcal{C}$, pour tous G et S, $u\theta[\alpha:=G] \in \mathrm{RED}_F{}^{\mathcal{C}[\alpha \mapsto S]}$
- Donc $\Lambda\alpha\,.\,u\theta \in \mathrm{RED}_{\forall\alpha\,.\,F}{}^{\mathcal{C}}$. □

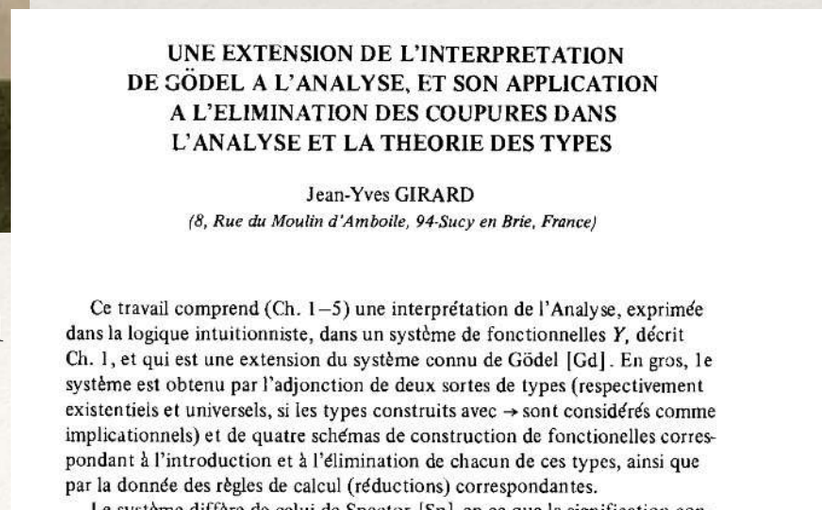
Lemme C′. Si (pour tout G et pour tout candidat S, $u[\alpha:=G] \in \mathrm{RED}_F{}^{\mathcal{C}[\alpha \mapsto S]}$) alors $\Lambda\alpha\,.\,u \in \mathrm{RED}_{\forall\alpha\,.\,F}{}^{\mathcal{C}}$

Normalisation forte

- **Thm.** Tout terme typable en système F_2 est fortement normalisable.
- Voilà!
- Ceci implique la cohérence de $\mathbf{HA}_2$…

Jean-Yves Girard

http://ekouter.net/img/img/JeanYvesGirard.jpg



UNE EXTENSION DE L'INTERPRETATION
DE GÖDEL A L'ANALYSE, ET SON APPLICATION
A L'ELIMINATION DES COUPURES DANS
L'ANALYSE ET LA THEORIE DES TYPES

Jean-Yves GIRARD
(8, Rue du Moulin d'Amboile, 94-Sucy en Brie, France)

Ce travail comprend (Ch. 1–5) une interprétation de l'Analyse, exprimée dans la logique intuitionniste, dans un système de fonctionnelles *Y*, décrit Ch. 1, et qui est une extension du système connu de Gödel [Gd]. En gros, le système est obtenu par l'adjonction de deux sortes de types (respectivement existentiels et universels, si les types construits avec → sont considérés comme implicationnels) et de quatre schémas de construction de fonctionelles correspondant à l'introduction et à l'élimination de chacun de ces types, ainsi que par la donnée des règles de calcul (réductions) correspondantes.

Logique du 2nd ordre

= Logique du 1er ordre + …

- **Formules atomiques** $A ::= P(e_1, \ldots, e_n)$

  mais ici P/n peut être quantifiée!

- Formules $F, G, \ldots ::= A$ (formules atomiques)
  $\mid F \Rightarrow G$ (comme en logique prop.)
  $\mid \forall i \,.\, F$ (premier ordre)
  $\mid \forall P_{/n} \,.\, F$ (second ordre)

Substitution

Ex: $(P(n) \Rightarrow P(\mathrm{s}(n)))\ [P{:=}n{\mapsto}(\text{« } n \text{ se décompose en facteurs premiers » })]$
$= \text{« } n \text{ se décompose en facteurs premiers »}$
$\Rightarrow \text{« } \mathrm{s}(n) \text{ se décompose en facteurs premiers »}$

- **Substitution du 1er ordre:** $F[i{:=}e]$ (comme d'habitude)
- **Substitution du 2nd ordre:** $F[P{:=}(i_1, \ldots, i_n){\mapsto}G]$
  définie par récurrence sur F…
- Cas important: si $F = P(e_1, \ldots, e_n)$
  $F[P{:=}(i_1, \ldots, i_n){\mapsto}G] = G[i_1{:=}e_1, \ldots, i_n{:=}e_n]$

Substitution du 2nd ordre, qu'on est en train de définir

Substitution du 1er ordre, déjà définie

Logique (minimale) du 2nd ordre

Logique du premier ordre intuitionniste)

$$\frac{}{\Gamma, x{:}F \vdash x : F}\ (\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\ (\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\ (\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall i\ .\ G}{\Gamma \vdash ue : G[i{:=}e]}\ (\forall\mathrm{E}) \qquad \frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i\ .\ u : \forall i\ .\ G}\ (\forall\mathrm{I})$$

(si i pas libre dans [aucune des formules de] Γ)

$$\frac{\Gamma \vdash u : \forall P_{/n}\ .\ F}{\Gamma \vdash uG : F[P{:=}(i_1, \ldots, i_n){\mapsto}G]}\ (\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda P_{/n}\ .\ u : \forall P_{/n}\ .\ F}\ (\forall^2\mathrm{I})$$

(si P pas libre dans [aucune des formules de] Γ)

Réductions

(β) $(\lambda x.u)v \to u[x:=v]$

(B) $(\Lambda i\ .\ u)e \to u[i:=e]$

(B^2) $(\Lambda P\ .\ u)G \to u[P:=(i_1, \ldots, i_n)\mapsto G]$

- **Thm (normalisation forte).** Les λ-termes de preuve de la logique du 2nd ordre **normalisent fortement**.
- Preuve: par effacement du premier ordre…

Effacement du 1er ordre

$$\frac{}{\Gamma, x{:}F \vdash x : F}\ (\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\ (\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x\, u : F \Rightarrow G}\ (\Rightarrow\mathrm{I})$$

Système F!

$$\frac{\Gamma \vdash u : \forall P\ .\ F}{\Gamma \vdash uG : F[P{:=}\ G]}\ (\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda P\ .\ u : \forall P\ .\ F}\ (\forall^2\mathrm{I})$$

(si P pas libre dans [aucune des formules de] Γ)

Normalisation forte

- **Observation clé:** Si $\Gamma \vdash u : F$ au 2nd ordre,
  alors $E(\Gamma) \vdash E(u) : E(F)$ en système F.

- Si $u \to v$ par (β) ou (B^2), alors $E(u) \to E(v)$ par (β) ou (B^2)
  Si $u \to v$ par (B), alors $E(u) = E(v)$

- Supposons $\Gamma \vdash u : F$ au 2nd ordre,
  toute réduction infinie partant de u n'utilise qu'un nb. fini de règles (β)+(B^2)… car le système F termine

- A partir d'un certain rang, seulement (B)… qui termine. □

(β) $(\lambda x.u)v \to u[x:=v]$
(B) $(\Lambda i\ .\ u)e \to u[i:=e]$
(B^2) $(\Lambda P\ .\ u)G \to u[P:=(i_1, \ldots, i_n)\mapsto G]$

Cohérence

- **Thm.** Les λ-termes de preuve de la logique du 2nd ordre **normalisent fortement**.

- **Corollaire.** La logique du 2nd ordre est **cohérente**.

- (argument habituel:
  pas de preuve en forme normale de $\vdash P()$
  et donc pas non plus de $\vdash \bot$, où $\bot = \forall P_{/0} \,.\, P()$.)

Arithmétique d'ordre 2 ($\mathbf{PA}_2$, $\mathbf{HA}_2$)

(sans les détails)

PA_2 = logique du 2nd ordre + …

« **Axiomes de Peano** »

pour chaque formule F

- (Refl) $\forall i \,.\, i \approx i$
- (Subst) $\forall i, j \,.\, i \approx j \Rightarrow F(i) \Rightarrow F(j)$
  (plus formellement, $\forall i, j \,.\, i \approx j \Rightarrow F \Rightarrow F[i:=j]$)
- $\forall i \,.\, \neg 0 \approx \mathrm{s}(i)$
- $\forall i, j \,.\, \mathrm{s}(i) \approx \mathrm{s}(j) \Rightarrow i \approx j$
- $\forall i \,.\, i+0 \approx i$
- $\forall i, j \,.\, i+\mathrm{s}(j) \approx \mathrm{s}(i+j)$
- $\forall i \,.\, i*0 \approx 0$
- $\forall i, j \,.\, i*\mathrm{s}(j) \approx i+i*j$

pour chaque formule F

- **Principe de récurrence:** $F(0) \Rightarrow (\forall j \,.\, F(j) \Rightarrow F(\mathrm{s}(j))) \Rightarrow \forall j \,.\, F(j)$

Note

- **PA**$_2$ = arithmétique de Peano du 2nd ordre
  = log. classique du 2nd ordre + Peano + récurrence

- **HA**$_2$ = arithmétique de **Heyting** du 2nd ordre
  = log. **intuitionniste** du 2nd ordre + Peano + réc.

- C'est cette dernière que nous allons étudier
  … les stakhanovistes ajouteront
  l'opérateur **C** de Felleisen pour obtenir **PA**$_2$!

$\mathbf{HA}_2$:

$$\frac{}{\Gamma, x{:}F \vdash x : F}\,(\mathrm{Ax})$$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}\,(\Rightarrow\mathrm{E}) \qquad \frac{\Gamma, x{:}F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}\,(\Rightarrow\mathrm{I})$$

$$\frac{\Gamma \vdash u : \forall i\,.\,G}{\Gamma \vdash ue : G[i{:=}e]}\,(\forall\mathrm{E}) \qquad \frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i\,.\,u : \forall i\,.\,G}\,(\forall\mathrm{I})$$

(si i pas libre dans [aucune des formules de] Γ)

$$\frac{\Gamma \vdash u : \forall P_{/n}\,.\,F}{\Gamma \vdash uG : F[P{:=}(i_1, \ldots, i_n){\mapsto}G]}\,(\forall^2\mathrm{E}) \qquad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda P_{/n}\,.\,u : \forall P_{/n}\,.\,G}\,(\forall^2\mathrm{I})$$

(si P pas libre dans [aucune des formules de] Γ)

+

$$\frac{}{\Gamma \vdash r_0 : 0 \approx 0}\,(Refl_0) \qquad \frac{\Gamma \vdash u : F_1 \quad F_1 \leftrightarrow^*_{\mathbb{N}} F_2}{\Gamma \vdash u : F_2}\,(\leftrightarrow^*_{\mathbb{N}})$$

$$\frac{\Gamma \vdash u : F[i := 0] \quad \Gamma \vdash v : \forall j \cdot F[i := j] \Rightarrow F[i := \mathtt{S}(j)]}{\Gamma \vdash Ruvt : F[i := t]}\,(Rec)$$

La 1ère astuce de Girard

- La cohérence de $\mathbf{HA}_2$ (ou de $\mathbf{PA}_2$) se ramène (de façon finitiste) à celle de la logique du 2nd ordre
- **Astuce (relativisation):** on remplace chaque formule quantifiée du 1er ordre

  $\forall i \,.\, G$

  par sa relativisée:

  $\forall i \,.\, \mathbf{na}$

Un entier, c'est un objet e
qui vérifie un
principe de récurrence local

- où $\mathbf{nat}(e)$ abrège: $\forall P_{/1} \,.\, (\forall j \,.\, P(j) \Rightarrow P(\mathrm{s}(j))) \Rightarrow P(0) \Rightarrow P(e)$

La 1ère astuce de Girard

la relativisée de F

- Girard observe que si $\vdash F$ prouvable en $\mathbf{HA}_2$, alors
  $\vdash \mathrm{rel}(F)$ est prouvable

Donc si la logique du 2nd ordre + ax. Peano est cohérente, alors $\mathbf{HA}_2$ aussi

- **Note.** $\mathrm{rel}(\text{« faux »}) = \text{« faux »}$

$\bot = \forall P_{/0} \,.\, P()$

$\bot = \forall \alpha \,.\, \alpha$

Effacement du 1er ordre

(la 2ème astuce de Girard)

- Mais, par effacement du premier ordre,
  les λ-termes de preuve de la logique du 2nd ordre + ax. de Peano (qui disparaissent tous!) **terminent**
  (Ça marche aussi avec (Refl_0) + preuve modulo $(\hookrightarrow^*_{\mathbb{N}})$ à la place des axiomes de Peano.)
- Par un argument usuel d'examen de formes normales, cette dernière est cohérente.
- Donc $\mathbf{HA}_2$ aussi!

Cohérence de l'arithmétique

- **Thm.** Il n'y a pas de preuve de $\vdash \bot$ en $\mathbf{HA}_2$.
- Par le second théorème d'incomplétude de Gödel, ce théorème n'est pas démontrable en $\mathbf{HA}_2$ (ou en $\mathbf{PA}_2$).

- En particulier, notre preuve n'est pas **finitiste** (codable en $\mathbf{PA}_1$)

Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I[1]**.**

Von **Kurt Gödel** in Wien.

1.

Die Entwicklung der Mathematik in der Richtung zu
Exaktheit hat bekanntlich dazu geführt, daß weite Gebiete
formalisiert wurden, in der Art, daß das Beweisen nach
wenigen mechanischen Regeln vollzogen werden kann. Die
sondsten derzeit aufgestellten formalen Systeme sind das Sy
Principia Mathematica (PM)[2]) einerseits, das Zermelo-Fr
sche (von J. v. Neumann weiter ausgebildete) Axiomensys
Mengenlehre[3]) andererseits. Diese beiden Systeme sind so v
alle heute in der Mathematik angewendeten Beweismethoden
formalisiert, d. h. auf einige wenige Axiome und Schlußregeln
geführt sind. Es liegt daher die Vermutung nahe, daß diese
und Schlußregeln dazu ausreichen, alle mathematischen Fra
sich in den betreffenden Systemen überhaupt formal aus
lassen, auch zu entscheiden. Im folgenden wird gezeigt,
nicht der Fall ist, sondern daß es in den beiden ang
Systemen sogar relativ einfache Probleme aus der Theorie
wöhnlichen ganzen Zahlen gibt[4]), die sich aus den Axiome

Par Auteur inconnu — Familienalbum der Familie Gödel,
Scan from Gianbruno Guerrerio,
Kurt Gödel - Logische Paradoxien und mathematische Wahrheit, S.24,
Domaine public,
https://commons.wikimedia.org/w/index.php?curid=10595692

Quoi de finitiste dans notre preuve?

- C'est quelque chose dans notre preuve de terminaison qui ne l'est pas… mais quoi?
- Evidemment, la notion d'**ensemble** $\mathrm{RED}_{F^{\mathcal{C}}}$…

  mais ce n'est pas si simple.
- Au lieu d'écrire 'soit $u \in \mathrm{RED}_{F \Rightarrow G^{\mathcal{C}}}$'

  on peut écrire

  'soit u tel que (pour tout $v \in \mathrm{RED}_{F^{\mathcal{C}}}$, $uv \in \mathrm{RED}_{G^{\mathcal{C}}}$)'

  et …

Quoi de finitiste dans notre preuve?

Ça, c'est du 2nd ordre, pas du 1er ordre

- Et au lieu d'écrire 'soit $u \in \mathrm{RED}_{\forall \alpha . F}$'

  on peut écrire

  'soit u tel que (pour tout G, pour tout candidat S, $uG \in \mathrm{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$)'

et par récurrence sur le type, éliminer toute mention d'ensembles $\mathrm{RED}_F^{\mathcal{C}}$

Quoi de finitiste dans notre preuve?

- En fait, si je vous donne une dérivation d'un jugement **fixé** $\Gamma \vdash u : F$ en $\mathbf{HA}_2$,
- je peux vous refaire toute la preuve que u est fortement normalisant…
- en inlinant toutes les mentions d'ensembles $\mathrm{RED}_G{}^{\mathcal{C}}$ et SN
- et cette preuve est formalisable en $\mathbf{HA}_2$

Quoi de finitiste dans notre preuve?

- La seule chose non formalisable en $\mathbf{HA}_2$ dans notre preuve, c'est la quantification universelle:
  pour tout jugement $\Gamma \vdash u : F$ en $\mathbf{HA}_2$, u termine.

- Phénomène bien connu (depuis Gödel): $\mathbf{HA}_2$
  n'est pas **ω-complète:**
  il existe une propriété P telle qu'on peut prouver $P(0)$, $P(1)$, …, $P(n)$, … (pour tout entier n), mais pas $\forall i \,.\, P(i)$

Curry-Howard

- Ce que nous dit notre formalisation de $\mathbf{HA}_2$, c'est que: les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ prouvablement totales en $\mathbf{HA}_2$ sont celles codables en λ-calcul typé en système F

~~+ **récurrence primitive à tous les types**~~

Pas besoin!
(1ère astuce de Girard)

Note: « ∃ » en logique du 2nd ordre

- … est définissable, comme en système F:

$$\exists\, P_{/n} \,.\, F \quad = \quad \forall \beta \,.\, (\forall P_{/n} \,.\, F \Rightarrow \beta) \Rightarrow \beta$$

- et pareil pour le « et », le « ou », le « non »,
  le ∃ du premier ordre, etc.

Fonctions provablement totales

- Une

telle

- Alor

$\vdash u : \forall i\ \mathbf{nat}(i) \Rightarrow \exists j\ .\ \mathbf{nat}(j) \wedge \mathrm{rel}(R(i,j))$

en logique du 2nd ordre + axiomes de Peano.

- Par effacement du premier ordre, on obtient un terme
$\vdash E(u) : \mathbf{nat} \Rightarrow \mathbf{nat} \wedge E(\mathrm{rel}(R(i,j)))$ typé en système F

Eh oui,

$\mathbf{nat} = \forall\alpha\ .\ (\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$ (système F)

est **l'effacement** du premier ordre

du **principe de récurrence** local

$\mathbf{nat}(e) = \forall P_{/1}\ .\ (\forall j\ .\ P(j) \Rightarrow P(\mathrm{s}(j))) \Rightarrow P(0) \Rightarrow P(e)$

Fonctions provablement totales

- Par effacement du premier ordre, on obtient un terme
  $\vdash E(u) : \mathbf{nat} \Rightarrow \mathbf{nat} \wedge E(\mathrm{rel}(R(i,j)))$ typé en système F
- et donc un terme $\vdash \lambda x\,.\,\pi_1(E(u)(x)) : \mathbf{nat} \Rightarrow \mathbf{nat}$
- On peut montrer que pour tout entier i,
  $$(\lambda x\,.\,\pi_1(E(u)(x)))\ulcorner i \urcorner \to^* \ulcorner j \urcorner$$
  où $\vdash R(i,j)$ est prouvable en $\mathbf{HA}_2$.
- (La preuve est donnée par une forme normale d'une preuve obtenue en appliquant $u : \forall i\,.\,\mathbf{nat}(i) \Rightarrow \exists j\,.\,\mathbf{nat}(j) \wedge \mathrm{rel}(R(i,j))$ à i et à une preuve de $\vdash \ulcorner i \urcorner : \mathbf{nat}(i)$ en logique du 2nd ordre.)

Fonctions provablement totales

- Réciproquement, si $\vdash u : \mathbf{nat} \Rightarrow \mathbf{nat}$ clos et typé en système F, alors:

- pour tout entier i, il existe un entier j t.q. $u\ulcorner i\urcorner \to^* \ulcorner j\urcorner$, **prouvablement en $\mathbf{HA}_2$**:
  — la réduction $u\ulcorner i\urcorner \to^* \ldots$ termine,
  prouvablement en $\mathbf{HA}_2$ (voir argument d'inlining des RED_{F^c})
  — une forme normale close de type **nat**
  est un entier de Church (à un détail près, voir TD)

- u définit donc une **fonction prouvablement totale en $\mathbf{HA}_2$.**
  (Je rappelle: j'omets quelques détails…)

Vers l’infini et au-delà

Arithmétique d'ordre ω

- Où l'on quantifie aussi sur des relations entre relations (topologies, tribus, …), sur des relations entre relations entre relations, etc.

- Voir section 4.5 du poly `types.pdf`

- La normalisation forte n'est pas si compliquée: on construit juste un **modèle** intuitif…
  à part que les valeurs de vérité sont les **candidats**

Attention!

- Tout ça finit par donner l'impression que pour toute logique, un argument par candidats va en montrer la cohérence: **non**!
- Par exemple, la **théorie naïve des ensembles** (intuitionniste)

$$\frac{\Gamma \vdash u : t\,\varepsilon\,\{i \mid F\}}{\Gamma \vdash \varepsilon_1 u : F[i := t]}\,(\varepsilon E) \qquad \frac{\Gamma \vdash u : F[i := t]}{\Gamma \vdash \varepsilon_2 u : t\,\varepsilon\,\{i \mid F\}}\,(\varepsilon I)$$

Le paradoxe de Russell (version intuitionniste)

$[A \triangleq \{i \mid F(i)\} \quad F(e) \triangleq \neg(e\ \varepsilon\ e)]$

- … est **incohérente**

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \neg(A\,\varepsilon\,A) \vdash x : F(A)}\,(Ax)}{x : \neg(A\,\varepsilon\,A) \vdash \varepsilon_2 x : A\,\varepsilon\,A}\,(\varepsilon I)}{x : \neg(A\,\varepsilon\,A) \vdash x(\varepsilon_2 x) : \bot}\,(\Rightarrow E)}{\vdash \lambda x \cdot x(\varepsilon_2 x) : \neg\neg(A\,\varepsilon\,A)}\,(\Rightarrow I) \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x : A\,\varepsilon\,A \vdash x : A\,\varepsilon\,\{i \mid \neg(i\,\varepsilon\,i)\}}\,(Ax)}{x : A\,\varepsilon\,A \vdash \varepsilon_1 x : \neg(A\,\varepsilon\,A)}\,(\varepsilon E)}{x : A\,\varepsilon\,A \vdash \varepsilon_1 x x : \bot}\,(\Rightarrow E)}{\vdash \lambda x \cdot \varepsilon_1 x x : \neg(A\,\varepsilon\,A)}\,(\Rightarrow I)$$

$$\vdash (\lambda x \cdot x(\varepsilon_2 x))(\lambda x \cdot \varepsilon_1 x x) : \bot\ (\Rightarrow E)$$

Attention!

- Tout ça finit par donner l'impression que pour toute logique, un argument par candidats va en montrer la cohérence: **non**!
- Un contre-exemple plus vicieux:
  le système U^- (logique d'ordre ω **polymorphe**)
  est **incohérent** (Coquand, 1994)

https://www.ihes.fr/wp-content/uploads/2019/12/t_coquand_web-1.jpg

La prochaine fois

- Nous parlerons de quelques stratégies d'implémentation du lambda-calcul
- (machines de Krivine, calculs à substitutions explicites)
  - notamment comment gérer **l'alpha-renommage**
  - … mais on ne fera pas beaucoup de pratique, et on se reposera très vite quelques questions théoriques. 🙂