

*Jean Goubault-Larrecq*

---

# Systeme F, polymorphisme et arithmétique du second ordre

---

28 avril 2020

---

# Le système F

---

- ❖ Aujourd'hui: quantification  $\forall$  du **second ordre**



---

# Le système F

---

- ❖ Aujourd'hui: quantification  $\forall$  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  $PA_2$



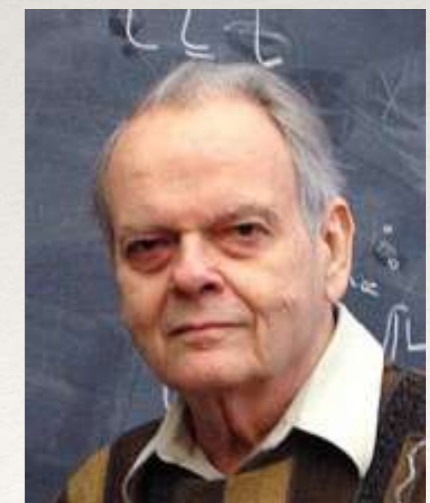
Jean-Yves Girard  
(les candidats de réductibilité,  
c'est lui)

# Le système F

- ❖ Aujourd'hui: quantification  $\forall$  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)



John C. Reynolds



# Systeme F

---

# Formules

---

- ❖ Ça ressemble à la logique du premier ordre, en plus simple (pas d'expressions)
- ❖ **Formules atomiques**  $\alpha =$  variables de formules  
(ou constantes, il suffit de ne pas quantifier dessus)
- ❖ Formules  $F, G, \dots ::= \alpha$  (formules atomiques)
  - |  $F \Rightarrow G$  (comme en logique prop.)
  - |  $\forall \alpha . F$  (nouveau)



# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

❖ Termes:

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

|  $uv$

|  $\lambda x.u$

❖ Réduction:

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E)$$

$$\frac{\Gamma, x:F \vdash x : F}{\Gamma, x:F \vdash u : G} (\lambda x)$$
$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} (\Rightarrow I)$$

# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

❖ Réduction:

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E)$$

$$\frac{\Gamma, x:F \vdash x : F}{\Gamma, x:F \vdash u : G} (\lambda x)$$
$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} (\Rightarrow I)$$



# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

❖ Réduction:

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

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{ (}\Rightarrow\text{E)}$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} \text{ (}\Rightarrow\text{I)}$$

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

# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

|  $uG$

❖ Réduction:

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

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{ (}\Rightarrow\text{E)}$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} \text{ (}\Rightarrow\text{I)}$$

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



# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

|  $uG$

❖ Réduction:

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

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{ (}\Rightarrow\text{E)}$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} \text{ (}\Rightarrow\text{I)}$$

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

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

|  $uG$

|  $\Lambda \alpha . u$

❖ Réduction:

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E)$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} (\Rightarrow I)$$

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

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )



# Systeme F<sub>2</sub>

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

|  $uG$

|  $\Lambda \alpha . u$

❖ Réduction:

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

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E)$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} (\Rightarrow I)$$

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

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

# Systeme ~~F<sub>2</sub>~~ F

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

~~|  $uG$~~

~~|  $\lambda \alpha . u$~~

❖ Réduction:

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

~~$(B^2) (\lambda \alpha . u)G \rightarrow u[\alpha:=G]$~~

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E)$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} (\Rightarrow I)$$

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

$$\frac{\Gamma \vdash u : F}{\Gamma \vdash \underbrace{\lambda \alpha . u}_u : \forall \alpha . F} (\forall^2 I)$$

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )



# Systeme F

❖ Types (formules):

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

|  $F \Rightarrow G$

|  $\forall \alpha . F$

❖ Termes:

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

|  $uv$

|  $\lambda x . u$

❖ Réduction:

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

$$\frac{}{\Gamma, x:F \vdash x : F} \text{(Ax)}$$
$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)}$$
$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x . u : F \Rightarrow G} \text{(}\Rightarrow\text{I)}$$
$$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]} \text{(}\forall^2\text{E)}$$
$$\frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F} \text{(}\forall^2\text{I)}$$

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

# Systeme F = polymorphisme

❖ En systeme F, on a:

$\lambda x.x : \forall \alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall \alpha . \alpha \Rightarrow \mathbf{list}(\alpha) \Rightarrow \mathbf{list}(\alpha)$  (d'efs manquantes: plus tard)

$$\begin{array}{c} \frac{}{\Gamma, x:F \vdash x : F} \text{(Ax)} \\ \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)} \quad \frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)} \\ \frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]} \text{(}\forall^2\text{E)} \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F} \text{(}\forall^2\text{I)} \\ \text{(si } \alpha \text{ pas libre dans [aucune des formules de] } \Gamma \text{)}$$



# Systeme F = polymorphisme

❖ En systeme F, on a:

$\lambda x.x : \forall \alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall \alpha . \alpha \Rightarrow \mathbf{list}(\alpha) \Rightarrow \mathbf{list}(\alpha)$  (défs manquantes: plus tard)

❖ on peut écrire: **cons** (1, **cons** (2, **nil**)) : **list**(nat)

et aussi: **cons** ((1,2), **cons** ((3,4), **nil**)) : **list**(nat\*nat)

	$\frac{}{\Gamma, x:F \vdash x : F}$ (Ax)
$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}$ ( $\Rightarrow$ E)	$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}$ ( $\Rightarrow$ I)
$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]}$ ( $\forall^2$ E)	$\frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F}$ ( $\forall^2$ I)
	<small>(si <math>\alpha</math> pas libre dans [aucune des formules de] <math>\Gamma</math>)</small>

# Systeme F = polymorphisme

❖ En systeme F, on a:

$\lambda x.x : \forall \alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall \alpha . \alpha \Rightarrow \mathbf{list}(\alpha) \Rightarrow \mathbf{list}(\alpha)$  (défs manquantes: plus tard)

❖ on peut écrire: **cons** (1, **cons** (2, **nil**)) : **list**(nat)

et aussi: **cons** ((1,2), **cons** ((3,4), **nil**)) : **list**(nat\*nat)

❖ et aussi **id id** (où **id** =  $\lambda x.x$ ) :  $\forall \alpha . \alpha \Rightarrow \alpha$

	$\frac{}{\Gamma, x:F \vdash x : F}$ (Ax)
$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}$ ( $\Rightarrow$ E)	$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}$ ( $\Rightarrow$ I)
$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]}$ ( $\forall^2$ E)	$\frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F}$ ( $\forall^2$ I)
	<small>(si <math>\alpha</math> pas libre dans [aucune des formules de] <math>\Gamma</math>)</small>



# Systeme F = polymorphisme

❖ En systeme F, on a:

$\lambda x.x : \forall \alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall \alpha . \alpha \Rightarrow \text{list}(\alpha)$

❖ on peut écrire: **cons** (1,

et aussi: **cons** ((1

❖ et aussi **id id** (où **id** =  $\lambda x.x$ ) :  $\forall \alpha . \alpha \Rightarrow \alpha$

	$\frac{}{\Gamma, x:F \vdash x : F}$ (Ax)
$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}$ ( $\Rightarrow$ E)	$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}$ ( $\Rightarrow$ I)
$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]}$ ( $\forall^2$ E)	$\frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F}$ ( $\forall^2$ I)
	<small>(si <math>\alpha</math> pas libre dans [aucune des formules de] <math>\Gamma</math>)</small>

Comme en ML ou en Haskell

plus tard)  
**list(nat\*nat)**

# Systeme F = polymorphisme

❖ En système F, on a:

$\lambda x.x : \forall \alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall \alpha . \alpha \Rightarrow \text{list}(\alpha)$

❖ 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))** :  $\text{list}(\forall \alpha . \alpha \Rightarrow \alpha)$

	$\frac{}{\Gamma, x:F \vdash x : F}$ (Ax)
$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G}$ ( $\Rightarrow$ E)	$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G}$ ( $\Rightarrow$ I)
$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]}$ ( $\forall^2$ E)	$\frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F}$ ( $\forall^2$ I)

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

Comme en ML ou en Haskell

plus tard)  
 $\text{list}(\text{nat} * \text{nat})$



# Systeme F = polymorphisme

❖ En système F, on a:

$\lambda x.x : \forall \alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall \alpha . \alpha \Rightarrow \text{list}(\alpha)$

❖ 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))** :  $\text{list}(\forall \alpha . \alpha \Rightarrow \alpha)$

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)}$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)}$$

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

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

Comme en ML ou en Haskell

plus tard)

**list(nat\*nat)**

Impossible en ML ou en Haskell!

# Systeme $F_2$ = polymorphisme, en plus lourd

❖ En système  $F_2$ , on a:

$\Lambda\alpha . \lambda x.x : \forall\alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall\alpha . \alpha \Rightarrow \text{list}(\alpha) \Rightarrow \text{list}(\alpha)$

$$\begin{array}{c} \frac{}{\Gamma, x:F \vdash x : F} \text{(Ax)} \\ \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)} \quad \frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)} \\ \frac{\Gamma \vdash u : \forall\alpha . F}{\Gamma \vdash uG : F[\alpha:=G]} \text{(}\forall^2\text{E)} \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha . u : \forall\alpha . F} \text{(}\forall^2\text{I)} \\ \text{(si } \alpha \text{ pas libre dans [aucune des formules de] } \Gamma \text{)} \end{array}$$



# Systeme $F_2$ = polymorphisme, en plus lourd

❖ En systeme  $F_2$ , on a:

$\Lambda\alpha . \lambda x.x : \forall\alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall\alpha . \alpha \Rightarrow \text{list}(\alpha) \Rightarrow \text{list}(\alpha)$

❖ 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)**

$$\begin{array}{c} \frac{}{\Gamma, x:F \vdash x : F} \text{(Ax)} \\ \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)} \quad \frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)} \\ \frac{\Gamma \vdash u : \forall\alpha . F}{\Gamma \vdash uG : F[\alpha:=G]} \text{(}\forall^2\text{E)} \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha . u : \forall\alpha . F} \text{(}\forall^2\text{I)} \\ \text{(si } \alpha \text{ pas libre dans [aucune des formules de] } \Gamma \text{)} \end{array}$$

# Systeme $F_2$ = polymorphisme, en plus lourd

❖ En systeme  $F_2$ , on a:

$\Lambda\alpha . \lambda x.x : \forall\alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall\alpha . \alpha \Rightarrow \text{list}(\alpha) \Rightarrow \text{list}(\alpha)$

❖ 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  $\Lambda\alpha . \text{id } (\alpha \Rightarrow \alpha) (\text{id } \alpha)$  (où **id**= $\Lambda\alpha . \lambda x.x$ ) :  $\forall\alpha . \alpha \Rightarrow \alpha$

$$\begin{array}{c}
 \frac{}{\Gamma, x:F \vdash x : F} \text{(Ax)} \\
 \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)} \quad \frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)} \\
 \frac{\Gamma \vdash u : \forall\alpha . F}{\Gamma \vdash uG : F[\alpha:=G]} \text{(}\forall^2\text{E)} \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha . u : \forall\alpha . F} \text{(}\forall^2\text{I)} \\
 \text{(si } \alpha \text{ pas libre dans [aucune des formules de] } \Gamma \text{)}
 \end{array}$$



# Systeme $F_2$ = polymorphisme, en plus lourd

❖ En systeme  $F_2$ , on a:

$\Lambda\alpha . \lambda x.x : \forall\alpha . \alpha \Rightarrow \alpha$

**cons** :  $\forall\alpha . \alpha \Rightarrow \text{list}(\alpha) \Rightarrow \text{list}(\alpha)$

❖ 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  $\Lambda\alpha . \text{id} (\alpha \Rightarrow \alpha) (\text{id } \alpha)$  (où **id**= $\Lambda\alpha . \lambda x.x$ ) :  $\forall\alpha . \alpha \Rightarrow \alpha$

❖ et aussi **cons** ( $\forall\alpha . \alpha \Rightarrow \alpha$ ) (**id**,

**cons** ( $\forall\alpha . \alpha \Rightarrow \alpha$ ) ( $\Lambda\alpha . \text{id} (\alpha \Rightarrow \alpha) (\text{id } \alpha)$ ,

**nil** ( $\forall\alpha . \alpha \Rightarrow \alpha$ )) : **list**( $\forall\alpha . \alpha \Rightarrow \alpha$ )

$$\begin{array}{c}
 \frac{}{\Gamma, x:F \vdash x : F} \text{(Ax)} \\
 \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)} \quad \frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)} \\
 \frac{\Gamma \vdash u : \forall\alpha . F}{\Gamma \vdash uG : F[\alpha:=G]} \text{(}\forall^2\text{E)} \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda\alpha . u : \forall\alpha . F} \text{(}\forall^2\text{I)} \\
 \text{(si } \alpha \text{ pas libre dans [aucune des formules de] } \Gamma\text{)}
 \end{array}$$

---

# 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)

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E) \quad \frac{\overline{\Gamma, x:F \vdash x : F} (\text{Ax}) \quad \Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} (\Rightarrow I)$$
$$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha := G]} (\forall^2 E) \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F} (\forall^2 I)$$

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

$$\begin{aligned} (\beta) \quad & (\lambda x.u)v \rightarrow u[x:=v] \\ (B) \quad & (\lambda \alpha . u)G \rightarrow u[\alpha:=G] \end{aligned}$$

# Propriétés fondamentales

- ❖ **Expressivité...**  
très forte  
(voir transparents suivants)

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E) \quad \frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} (\Rightarrow I)$$
$$\frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]} (\forall^2 E) \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F} (\forall^2 I)$$

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

$$(\beta) \quad (\lambda x.u)v \rightarrow u[x:=v]$$
$$(B) \quad (\lambda \alpha . u)G \rightarrow u[\alpha:=G]$$



# Propriétés fondamentales

- ❖ **Expressivité...**  
très forte  
(voir transparents suivants)

$$\begin{array}{c}
 \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(\(\Rightarrow\text{E}\)} \quad \frac{\Gamma, x:F \vdash x : F}{\Gamma, x:F \vdash u : G} \text{(\(\Rightarrow\text{I}\)} \\
 \frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash u : F[\alpha:=G]} \text{(\(\forall^2\text{E}\)} \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash u : \forall \alpha . F} \text{(\(\forall^2\text{I}\)} \\
 \text{(si } \alpha \text{ pas libre dans [aucune des formules de] } \Gamma \text{)}
 \end{array}$$

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

$$\begin{array}{l}
 (\beta) \quad (\lambda x . u)v \rightarrow u[x:=v] \\
 (B) \quad (\lambda \alpha . u)G \rightarrow u[\alpha:=G]
 \end{array}$$

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

Pouvoir expressif



---

# Le « et » en système $F_2$

---

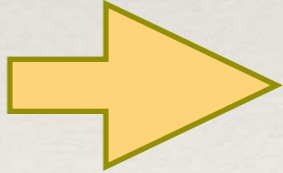
- ❖ On peut **définir**  $F \wedge G = \forall \alpha . (F \Rightarrow G \Rightarrow \alpha) \Rightarrow \alpha$  ( $\alpha$  fraîche)
- ❖ Intuitivement,  
 $F \wedge G$  est le **plus petit** type impliqué par  $F$  et  $G$   
(voyez  $\forall$  comme une sorte d'intersection)

---

# Le « et » en système $F_2$

---

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



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

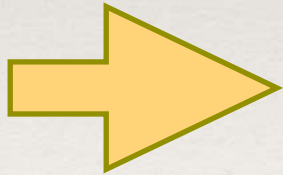


# Le « et » en système $F_2$

❖ On peut définir  $F \wedge G = \forall \alpha . (F \Rightarrow G \Rightarrow \alpha) \Rightarrow \alpha$  ( $\alpha$  fraîche)

❖  $\langle u, v \rangle = \lambda \alpha . \lambda x . x u v$  (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!



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

# Le « et » en système $F_2$

❖ On peut définir  $F \wedge G = \forall \alpha . (F \Rightarrow G \Rightarrow \alpha) \Rightarrow \alpha$  ( $\alpha$  fraîche)

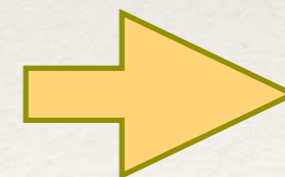
❖  $\langle u, v \rangle = \lambda \alpha . \lambda x . x u v$  (eh oui, c'est la paire de Church!)

$\pi_1 u = u$  A vous de le découvrir en TD!

$\pi_2 u = u$

❖  $\pi_1 \langle u, v \rangle \rightarrow^+ u$

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



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



---

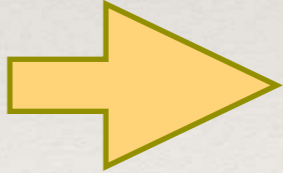
# Le « ou » en système $F_2$

---

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



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



# Le « ou » en système $F_2$

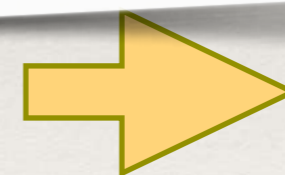
❖ On peut définir  $\forall \alpha . (F \Rightarrow \alpha) \Rightarrow (G \Rightarrow \alpha) \Rightarrow \alpha$  ( $\alpha$  fraîche)

❖  $l_1 u = \Lambda \alpha . \lambda x, y. x u$

$l_2 u = \Lambda \alpha . \lambda x, y. y u$

case  $u$  of  $(\lambda x_1. v_1) \mid (\lambda x_2. v_2)$

A vous de le découvrir en TD!



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

# Le « ou » en système $F_2$

❖ On peut définir  $\forall \alpha . (F \Rightarrow \alpha) \Rightarrow (G \Rightarrow \alpha) \Rightarrow \alpha$  ( $\alpha$  fraîche)

❖  $l_1 u = \Lambda \alpha . \lambda x, y. x u$

$l_2 u = \Lambda \alpha . \lambda x, y. y u$

case  $u$  of  $l_1 x_1 \mapsto v_1 \mid l_2 x_2 \mapsto v_2$   $(\lambda x_2. v_2)$

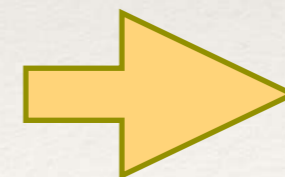
A vous de le découvrir en TD!

❖ case  $l_1 u$  of  $l_1 x_1 \mapsto v_1 \mid l_2 x_2 \mapsto v_2$

$\rightarrow^+ v_1[x_1 := u]$

case  $l_2 u$  of  $l_1 x_1 \mapsto v_1 \mid l_2 x_2 \mapsto v_2$

$\rightarrow^+ v_2[x_2 := u]$



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



---

# Le « faux » en système $F_2$

---

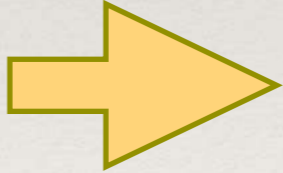
- ❖ On peut **définir**  $\perp = \forall \alpha . \alpha$
- ❖ Intuitivement,  
 $\perp$  est le **plus petit type** (tout court)  
(voyez  $\forall$  comme une sorte d'intersection)

---

# Le « faux » en système $F_2$

---

❖ On peut définir  $\perp = \forall \alpha . \alpha$

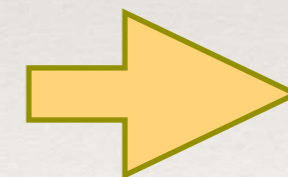


	Monde de la logique	Monde des programmes
$F$	formule	type
$u$	preuve	$\lambda$ -terme (programme)
$u \rightarrow^* v$	élimination des détours	$\beta$ -réduction (exécution)



# Le « faux » en système $F_2$

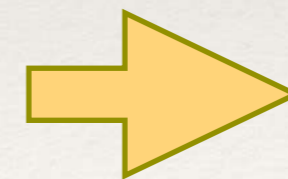
- ❖ On peut définir  $\perp = \forall \alpha . \alpha$
- ❖  $\nabla u = uF$



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

# Le « faux » en système $F_2$

- ❖ On peut définir  $\perp = \forall \alpha . \alpha$
- ❖  $\nabla u = uF$
- ❖  $\nabla uv \rightarrow^+ \nabla u$



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



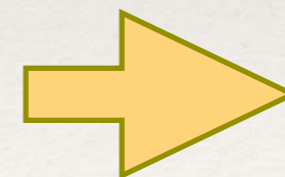
# Le « faux » en système $F_2$

❖ On peut définir  $\perp = \forall \alpha . \alpha$

❖  $\nabla u = uF$

❖  ~~$\nabla uv \rightarrow^+ \nabla u$~~

❖ ... ah non, raté!



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

# Le quantificateur existentiel

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

$$\exists \alpha . F$$

avec les règles:

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma, H$ )

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



Jean-Yves Girard

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  $\rightarrow$  sont considérés comme implicationnels) et de quatre schémas de construction de fonctionnelles 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 quantificateur existentiel

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

$$\exists \alpha . F$$

avec les règles:

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma, H$ )

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

- ❖ Ce n'est pas la peine!



Jean-Yves Girard

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  $\rightarrow$  sont considérés comme implicationnels) et de quatre schémas de construction de fonctionnelles 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  $\exists \alpha . F = \forall \beta . (\forall \alpha . F \Rightarrow \beta) \Rightarrow \beta$  ( $\beta$  fraîche)

❖ Intuitivement,

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

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

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma, H$ )

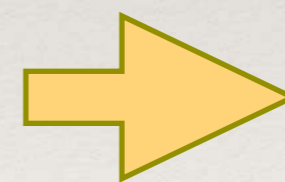


# Le « $\exists$ » en système $F_2$

❖ On peut définir  $\exists \alpha . F =$

$\forall \beta . (\forall \alpha . F \Rightarrow \beta) \Rightarrow \beta$  ( $\beta$  fraîche)

A vous de le découvrir en TD!



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

# Le « $\exists$ » en système $F_2$

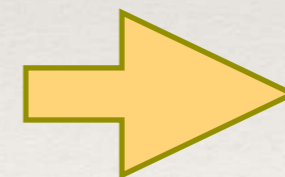
❖ On peut définir  $\exists \alpha . F =$

❖  $\iota(G, u) = \Lambda \beta . \lambda x . xGu$

case  $u$  of  $\iota(\alpha, x) \mapsto v = uH$

$\forall \beta . (\forall \alpha . F \Rightarrow \beta) \Rightarrow \beta$  ( $\beta$  fraîche)

A vous de le découvrir en TD!



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



# Le « $\exists$ » en système $F_2$

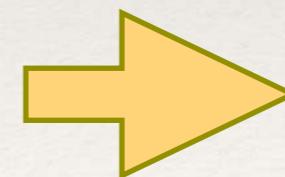
❖ On peut définir  $\exists \alpha . F =$

❖  $\iota(G, u) = \Lambda \beta . \lambda x . x G u$

case  $u$  of  $\iota(\alpha, x) \mapsto v = u$

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

$\forall \beta . (\forall \alpha . F \Rightarrow \beta) \Rightarrow \beta$  ( $\beta$  fraîche)  
A vous de le découvrir en TD!  
 $(\beta, x$  fraîches)  
 $(\lambda x . v)$



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

# Le « $\exists$ » en système $F_2$

C'est un « couple »  $(G, u)$ ...  
et le type réel d'implémentation

- ❖ On peut définir  $\exists$
- ❖  $\iota(G, u) = \Lambda\beta . \lambda x. xGu$   
case  $u$  of  $\iota(\alpha, x) \mapsto v = uE$
- ❖ case  $\iota(G, u)$  of  $\iota(\alpha, x) \mapsto v$   
 $\rightarrow^+ v[\alpha:=G, x:=u]$

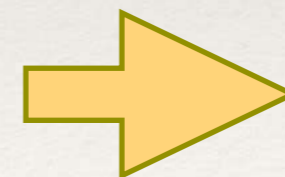
$G$  est caché de l'extérieur

( $\beta$  fraîche)

A vous de le découvrir en TD!

( $\beta, x$  fraîches)

( $\lambda x. v$ )



	Monde de la logique	Monde des programmes
$F$	formule	type
$u$	preuve	$\lambda$ -terme (programme)
$u \rightarrow^* v$	élimination des détours	$\beta$ -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
```

# 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
```

```
abs =  $\exists \alpha .$ 
      ( $\alpha \Rightarrow \alpha \Rightarrow \alpha$ )  $\wedge$  (* and_intro *)
      ( $\alpha \Rightarrow \alpha$ )  $\wedge$            (* and_elim1 *)
      ...

 $\iota(\text{formule}, \langle \lambda x, y. x \wedge y, \dots \rangle) : \text{abs}$ 
```



# 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
```

```
abs =  $\exists \alpha .$ 
      ( $\alpha \Rightarrow \alpha \Rightarrow \alpha$ )  $\wedge$  (* and_intro *)
      ( $\alpha \Rightarrow \alpha$ )  $\wedge$            (* and_elim1 *)
      ...
```

```
 $\iota(\text{formule}, \langle \lambda x, y. x \wedge y, \dots \rangle) : \text{abs}$ 
```

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

---

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

---

# Le type **nat** des entiers

---

- ❖ On définit **nat** =  $\forall \alpha . (\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$
- ❖ L'entier de Church  $[n] = \Lambda \alpha . \lambda f, x. f^n(x) : \mathbf{nat}$



---

# Le type **nat** des entiers

---

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

- ❖ On définit **nat** =  $\forall \alpha . (\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$
- ❖ L'entier de Church  $[n] = \Lambda \alpha . \lambda f, x. f^n(x) : \mathbf{nat}$

# Le type **nat** des entiers

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

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

❖ L'entier de Church  $[n] = \lambda \alpha . \lambda f, x. f^n(x) : \mathbf{nat}$

« type de 0 » :  $\alpha$



# Le type **nat** des entiers

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

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

❖ L'entier de Church  $[n] = \lambda \alpha . \lambda f, x. f^n(x) : \mathbf{nat}$

« type de 0 » :  $\alpha$

❖ Le **récurseur**: si  $u:F, v:\mathbf{nat} \Rightarrow F \Rightarrow F$ , et  $w:\mathbf{nat}$ ,

$Ruvw = \pi_1 F (w (F * \mathbf{nat}))$

$(\lambda z. \langle v(\pi_2 \mathbf{nat} z)(\pi_1 F z), s(\pi_2 \mathbf{nat} z) \rangle)$

$\langle u, [0] \rangle) : F$  (~formule du prédécesseur)

# Le type **nat** des entiers

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

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

❖ L'entier de Church  $[n] = \lambda \alpha . \lambda f, x. f^n(x) : \mathbf{nat}$

« type de 0 » :  $\alpha$

❖ Le **récurseur**: si  $u:F, v:\mathbf{nat} \Rightarrow F \Rightarrow F$ , et  $w:\mathbf{nat}$ ,

$$Ruvw = \pi_1 F (w (F * \mathbf{nat}))$$

$$(\lambda z. \langle v(\pi_2 \mathbf{nat} z)(\pi_1 F z), s(\pi_2 \mathbf{nat} z) \rangle)$$

$$\langle u, [0] \rangle : F \quad (\sim \text{formule du prédécesseur})$$

❖  $Ruv[0] \rightarrow^+ u$

$$Ruv[n+1] =_{\beta} v[n](Ruv[n])$$

(voir exercice 39 du poly types.pdf)



---

# Le type $\mathbf{list}(F)$ des listes de $F$

---

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

---

# Le type $\mathbf{list}(F)$ des listes de $F$

---

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

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



---

# Le type $\mathbf{list}(F)$ 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** » :  $\alpha$

# Le type $\mathbf{list}(F)$ des listes de $F$

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

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

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

« type de **nil** » :  $\alpha$

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



# Le type $\mathbf{list}(F)$ des listes de $F$

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

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

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

« type de **nil** » :  $\alpha$

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

❖ Le récursueur sur les listes... exercice!

(voir 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  $\lambda$ -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?

- ❖ Rappel (logique du premier ordre)
- ❖ Fonction d'effacement:  
enlève tout ce qui est du premier ordre

$$\begin{array}{c}
 \frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{ (}\Rightarrow\text{E)} \quad \frac{\Gamma, x:F \vdash x : F}{\Gamma, x:F \vdash u : G} \text{ (Ax)} \\
 \frac{\Gamma \vdash u : \forall i . G}{\Gamma \vdash ue : G[i:=e]} \text{ (}\forall\text{E)} \quad \frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i . u : \forall i . G} \text{ (}\forall\text{I)} \\
 \text{(si } i \text{ pas libre dans [aucune des formules de] } \Gamma \text{)}
 \end{array}$$

- ❖  $E(P(e_1, \dots, e_n))=P$     $E(\forall i . F)=E(F)$     $E(F \Rightarrow G)=E(F) \Rightarrow E(G)$   
 $E(\Lambda i . u)=E(u)$     $E(ue)=E(u)$   
 $E(uv)=E(u)E(v)$     $E(\lambda x . u)=\lambda x . E(u)$     $E(x)=x$

$$\begin{array}{l}
 (\beta) \quad (\lambda x . u)v \rightarrow u[x:=v] \\
 (B) \quad (\Lambda i . u)e \rightarrow u[i:=e]
 \end{array}$$

- ❖ **Observation clé:** Si  $\Gamma \vdash u : F$  **au premier ordre**,  
alors  $E(\Gamma) \vdash E(u) : E(F)$  en types simples.

Si  $u \rightarrow v$  par  $(\beta)$ , alors  $E(u) \rightarrow E(v)$  par  $(\beta)$

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



# Effacement, comme au premier ordre?

❖ Rappel

❖ Fonction

enlève

❖  $E(P(e_1, \dots, e_n)) = E(P)(E(e_1), \dots, E(e_n))$

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E) \quad \frac{\Gamma, x:F \vdash x : F}{\Gamma, x:F \vdash u : G} (\Rightarrow I) \quad \frac{\Gamma \vdash u : \forall \alpha . F}{\Gamma \vdash uG : F[\alpha := G]} (\forall^2 E) \quad \frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda \alpha . u : \forall \alpha . F} (\forall^2 I)$$

(si  $\alpha$  pas libre dans [aucune des formules de]  $\Gamma$ )

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} (\Rightarrow E) \quad \frac{\Gamma, x:F \vdash x : F}{\Gamma, x:F \vdash u : G} (\Rightarrow I) \quad \frac{\Gamma \vdash u : \forall i . G}{\Gamma \vdash ue : G[i := e]} (\forall E) \quad \frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i . u : \forall i . G} (\forall I)$$

(si  $i$  pas libre dans [aucune des formules de]  $\Gamma$ )

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

( $\beta$ )  $(\lambda x . u)v \rightarrow u[x := v]$   
 ( $B$ )  $(\Lambda \alpha . u)G \rightarrow u[\alpha := G]$

( $\beta$ )  $(\lambda x . u)v \rightarrow u[x := v]$   
 ( $B$ )  $(\Lambda i . u)e \rightarrow u[i := e]$

❖ Observation

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

Si  $u \rightarrow v$  par ( $\beta$ ), alors  $E(u) = E(v)$

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

---

# Candidats de réductibilité

---

- ❖ ... candidats de réductibilité, comme en  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$



---

# Candidats de réductibilité

---

- ❖ ... candidats de réductibilité, comme en  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$
- ❖ Juste un petit souci: pourquoi ceci est-il  
une définition valide? (par récurrence sur quoi?)

---

# Candidats de réductibilité

---

- ❖ ... candidats de réductibilité, comme en  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$
- ❖ 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**



---

# Candidats de réductibilité

---

- ❖ ... candidats de réductibilité, comme en  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$
- ❖ 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  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$



---

# Candidats de réductibilité

---

- ❖ ... candidats de réductibilité, comme en  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$
- ❖ Prenons le type « faux »  $\forall \alpha . \alpha$ :  
la définition de  $\text{RED}_{\forall \alpha . \alpha}$  dépend de  $\text{RED}_G$   
pour tous les types  $G$  (y compris « faux » lui-même)

---

# Candidats de réductibilité

---

- ❖ ... candidats de réductibilité, comme en  $\mathbf{HA}_1$
- ❖  $\text{RED}_\alpha = \text{RED}_\perp = \text{SN}$ ,  $\text{RED}_{F \Rightarrow G} = \text{RED}_F \Rightarrow \text{RED}_G$   
 $\text{RED}_{\forall \alpha . F} = \{u \mid \text{pour tout } G, uG \in \text{RED}_{F[\alpha := G]}\}$
- ❖ Prenons le type « faux »  $\forall \alpha . \alpha$ :  
la définition de  $\text{RED}_{\forall \alpha . \alpha}$  dépend de  $\text{RED}_G$   
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  $\llbracket F \rrbracket$  au lieu de  $RED_F$

---

# Sémantique ...

---

- ❖ Vous vous souvenez de mon explication des candidats de réductibilité comme valeurs de vérité particulières?
- ❖ J'avais écrit  $\llbracket F \rrbracket$  au lieu de  $RED_F$
- ❖ Ce qu'on vient de faire, c'est de tenter de donner une sémantique aux formules par:



# Sémantique ...

- ❖ Vous vous souvenez de mon explication des candidats de réductibilité comme valeurs de vérité particulières?
- ❖ J'avais écrit  $\llbracket F \rrbracket$  au lieu de  $RED_F$
- ❖ Ce qu'on vient de faire, c'est de tenter de donner une sémantique aux formules par:
  - ❖  $\llbracket \alpha \rrbracket = SN$                        $\llbracket \perp \rrbracket = SN$
  - $\llbracket F \Rightarrow G \rrbracket = \llbracket F \rrbracket \Rightarrow \llbracket G \rrbracket$
  - $\llbracket \forall \alpha . F \rrbracket = \{u \mid \text{pour tout } G, uG \in \llbracket F[\alpha := G] \rrbracket\}$

Ça n'a aucun sens...

# Sémantique ...

❖ Vous vous souvenez de mon explication des candidats de réductibilité comme valeurs de vérité particulières?

❖ J'avais écrit  $\llbracket F \rrbracket$  au lieu de  $\text{RED}_F$

❖ Ce qui est bizarre, c'est de donner une sémantique à une formule sans donner une

Mais ça, c'est bizarre, non?

❖  $\llbracket \alpha \rrbracket = \text{SN}$                        $\llbracket \perp \rrbracket = \text{SN}$

$\llbracket F \Rightarrow G \rrbracket = \llbracket F \rrbracket \Rightarrow \llbracket G \rrbracket$

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

Ça n'a aucun sens...



---

# Sémantique à la Tarski...

---

- ❖ Dans une sémantique à la Tarski, on se donnerait un **environnement**  $\varrho$  tel que  $\varrho(\alpha)$  serait une valeur de vérité pour toute variable de type  $\alpha$



# Sémantique à la Tarski...

❖ Dans une sémantique à la Tarski, on se donnerait un **environnement**  $\varrho$  tel que  $\varrho(\alpha)$  serait une valeur de vérité pour toute variable de type  $\alpha$

❖  $\llbracket \alpha \rrbracket \varrho = \varrho(\alpha)$                        $\llbracket \perp \rrbracket \varrho = \text{SN}$

$\llbracket F \Rightarrow G \rrbracket \varrho = \llbracket F \rrbracket \varrho \Rightarrow \llbracket G \rrbracket \varrho$

$\llbracket \forall \alpha . F \rrbracket \varrho = \text{infimum (« et ») des } \llbracket F \rrbracket (\varrho[\alpha \mapsto S]),$

lorsque  $S$  parcourt les valeurs de vérité





# Sémantique à la Tarski...

- ❖ Dans une sémantique à la Tarski, on se donne un langage  $\mathcal{L}$  et un ensemble  $\mathcal{V}$  de valeurs de vérité. On définit un **interprétation**  $\varrho$  tel que  $\varrho(\perp) = \text{SN}$  (« vrai »)! pour de vérité pour toute



- ❖  $\llbracket \alpha \rrbracket \varrho = \varrho(\alpha)$        $\llbracket \perp \rrbracket \varrho = \text{SN}$   
 $\llbracket F \Rightarrow G \rrbracket \varrho = \llbracket F \rrbracket \varrho \Rightarrow \llbracket G \rrbracket \varrho$   
 $\llbracket \forall \alpha . F \rrbracket \varrho = \text{infimum (« et ») des } \llbracket F \rrbracket (\varrho[\alpha \mapsto S]),$   
lorsque  $S$  parcourt les valeurs de vérité

# Sémantique à la Tarski...



- ❖ Dans une sémantique à la Tarski, on se donne un langage  $\mathcal{L}$  et un ensemble  $S$  de valeurs de vérité. On définit alors une interprétation  $\varrho$  tel que  $\varrho(\perp) = \text{SN}$  (« vrai »)! pour de vérité pour toute formule  $F$  de  $\mathcal{L}$ .

- ❖  $\llbracket \alpha \rrbracket \varrho = \varrho(\alpha)$   $\llbracket \perp \rrbracket \varrho = \text{SN}$  récurrence sur la formule!  
 $\llbracket F \Rightarrow G \rrbracket \varrho = \llbracket F \rrbracket \varrho \Rightarrow \llbracket G \rrbracket \varrho$   
 $\llbracket \forall \alpha . F \rrbracket \varrho = \text{infimum (« et ») des } \llbracket F \rrbracket (\varrho[\alpha \mapsto S]),$   
lorsque  $S$  parcourt les valeurs de vérité



---

# Sémantique à la Tarski-Girard

---

- ❖  $\llbracket \alpha \rrbracket \varrho = \varrho(\alpha)$                        $\llbracket \perp \rrbracket \varrho = \text{SN}$
- $\llbracket F \Rightarrow G \rrbracket \varrho = \llbracket F \rrbracket \varrho \Rightarrow \llbracket G \rrbracket \varrho$
- $\llbracket \forall \alpha . F \rrbracket \varrho = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S,$   
 $uG \in \llbracket F \rrbracket (\varrho[\alpha \mapsto S])\}$

# En notation « RED »

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

- ❖  $RED_{\alpha}^{\mathcal{C}} = \mathcal{C}(\alpha)$        $RED_{\perp}^{\mathcal{C}} = SN$

$$RED_{F \Rightarrow G}^{\mathcal{C}} = RED_F^{\mathcal{C}} \Rightarrow RED_G^{\mathcal{C}}$$

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

- ❖ (Comparez:

$$[[\alpha]]\varrho = \varrho(\alpha) \qquad [[\perp]]\varrho = SN$$

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

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

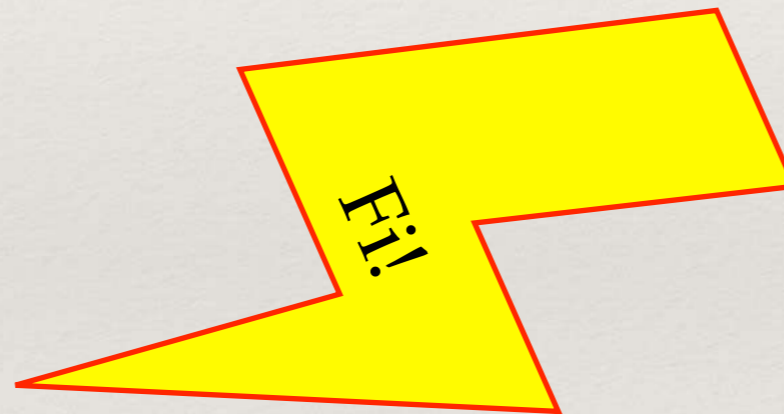


---

# L'ironie du sort

---

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



---

# L'opérateur $\Pi$

---

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

❖ Autrement dit,

$$\text{RED}_{\forall\alpha}.F^c = \Pi(S \mapsto \text{RED}_F^{c[\alpha \mapsto S]}),$$

❖ où pour toute fonction  $f: \text{candidats} \rightarrow \text{candidats}$ ,

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



---

# (CR1)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (CR1): pour toute  $f: \text{candidats} \rightarrow \text{candidats}$ ,  
 $\Pi(f) \subseteq \text{SN}$

---

# (CR1)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (CR1): pour toute  $f: \text{candidats} \rightarrow \text{candidats}$ ,  
 $\Pi(f) \subseteq \text{SN}$
- ❖ En effet, pour tout  $u \in \Pi(f)$ ,  
choisissons un type  $G$  quelconque (il en existe)  
**posons  $S = \text{SN}$  — car  $\text{SN}$  est un candidat**  
alors  $uG \in f(S)$



---

# (CR1)

---

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

---

# (CR2)

---

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



---

# (CR2)

---

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

---

# (CR2)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (CR2): Si  $u \in \Pi(f)$  et  $u \rightarrow u'$  alors  $u' \in \Pi(f)$ .
- ❖ Pour tout type  $G$ , pour tout candidat  $S$ ,  $uG \in f(S)$
- ❖ Or  $uG \rightarrow u'G$   
et  $f(S)$  candidat, donc  $u'G \in f(S)$  par (CR2) sur  $f(S)$



---

# (CR2)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (CR2): Si  $u \in \Pi(f)$  et  $u \rightarrow u'$  alors  $u' \in \Pi(f)$ .
- ❖ Pour tout type  $G$ , pour tout candidat  $S$ ,  $uG \in f(S)$
- ❖ Or  $uG \rightarrow u'G$   
et  $f(S)$  candidat, donc  $u'G \in f(S)$  par (CR2) sur  $f(S)$
- ❖ Donc  $u' \in \Pi(f)$ .  $\square$

---

# (CR3)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (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  $\lambda$  ni par  $\Lambda$ )



---

# (CR3)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (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  $\lambda$  ni par  $\Lambda$ )
- ❖ Pour tout type  $G$ , **pour tout candidat**  $S$ , les réduits en une étape de  $uG$  sont les  $u'G$

---

# (CR3)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (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  $\lambda$  ni par  $\Lambda$ )
- ❖ 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



---

# (CR3)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (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  $\lambda$  ni par  $\Lambda$ )
- ❖ 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  $f(S)$

---

# (CR3)

---

- ❖  $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$
- ❖ (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  $\lambda$  ni par  $\Lambda$ )
- ❖ 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  $u \in \Pi(f)$ .  $\square$



# Candidats de réductibilité

$$\diamond \text{RED}_{\alpha}^{\mathcal{C}} = \mathcal{C}(\alpha) \quad \text{RED}_{\perp}^{\mathcal{C}} = \text{SN}$$

$$\text{RED}_{F \Rightarrow G}^{\mathcal{C}} = \text{RED}_F^{\mathcal{C}} \Rightarrow \text{RED}_G^{\mathcal{C}}$$

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

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

$\diamond$  Donc, par récurrence sur le type  $F$ :

**Lemme B.** Pour tout type  $F$ ,

pour tout contexte de candidats  $\mathcal{C}$ ,

$\text{RED}_F^{\mathcal{C}}$  est un candidat.

# Lemmes utiles (1)

- ❖  $\text{RED}_{\alpha}^{\mathcal{C}} = \mathcal{C}(\alpha)$        $\text{RED}_{\perp}^{\mathcal{C}} = \text{SN}$   
 $\text{RED}_{F \Rightarrow G}^{\mathcal{C}} = \text{RED}_F^{\mathcal{C}} \Rightarrow \text{RED}_G^{\mathcal{C}}$   
 $\text{RED}_{\forall \alpha . F}^{\mathcal{C}} = \Pi(S \mapsto \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]})$   
( $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$ )
- ❖ Comme avant, mais **pour tout contexte de candidats  $\mathcal{C}$** :
- ❖ **Lemme C.** Si (pour tout  $v \in \text{RED}_F^{\mathcal{C}}, s[x:=v] \in \text{RED}_G^{\mathcal{C}}$ )  
alors  $\lambda x.s \in \text{RED}_{F \Rightarrow G}^{\mathcal{C}}$



# Lemmes utiles (2)

- ❖  $\text{RED}_{\forall\alpha . F^c} = \Pi(S \mapsto \text{RED}_F^{c[\alpha \mapsto S]})$   
( $\Pi(f) = \{u \mid \text{pour tout } G, \text{ pour tout candidat } S, uG \in f(S)\}$ )
- ❖ **Lemme C'**. Si (pour tout  $G$  et pour tout candidat  $S$ ,  
 $u[\alpha := G] \in \text{RED}_F^{c[\alpha \mapsto S]}$ )  
alors  $\Lambda\alpha . u \in \text{RED}_{\forall\alpha . F^c}$
- ❖ Pour tous  $G$  et  $S$ , on regarde les réduits en une étape de  $(\Lambda\alpha . u)G \dots$
- ❖ et on applique (CR3) + récurrence sur  $u$  le long de  $\rightarrow$ .  $\square$

---

# Quelques lemmes triviaux

---

- ❖ Dans une sémantique à la Tarski, on a toujours:





# Quelques lemmes triviaux

---



- ❖ Dans une sémantique à la Tarski, on a toujours:
- ❖ **(Substitution.)**  $\llbracket F[\alpha:=G] \rrbracket \varrho = \llbracket F \rrbracket (\varrho[\alpha \mapsto \llbracket G \rrbracket \varrho])$

# Quelques lemmes triviaux



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

(Récurrences triviales sur  $F$ .)



---

# Quelques lemmes utiles

---

- ❖ De même, ici nous avons:

---

# Quelques lemmes utiles

---

❖ De même, ici nous avons:

❖ **Lemme S.**  $\text{RED}_{F[\alpha:=G]}^c = \text{RED}_F^{c[\alpha \mapsto S]}$  où  $S = \text{RED}_G^c$



---

# Quelques lemmes utiles

---

❖ De même, ici nous avons:

❖ **Lemme S.**  $\text{RED}_{F[\alpha:=G]}^c = \text{RED}_F^{c[\alpha \mapsto S]}$  où  $S = \text{RED}_G^c$

❖ **Lemme L.** Si  $\alpha$  pas libre dans  $F$ ,  $\text{RED}_F^{c[\alpha \mapsto S]} = \text{RED}_F^c$

(Récurrences triviales sur  $F$ .)

---

# Normalisation forte

---

- ❖ **Def:**  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$  ssi pour tout  $x:F$  dans  $\Gamma$ ,  $\theta(x) \in \text{RED}_F^{\mathcal{C}}$   
+  $\theta(\alpha)$  est un type pour tout  $\alpha$
- ❖ **Thm.** si  $\Gamma \vdash u : G$  dérivable alors pour tout contexte  $\mathcal{C}$ ,  
pour toute  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$ ,  $u\theta \in \text{RED}_G^{\mathcal{C}}$ .



# Normalisation forte

- ❖ **Def:**  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$  ssi pour tout  $x:F$  dans  $\Gamma$ ,  $\theta(x) \in \text{RED}_F^{\mathcal{C}}$   
+  $\theta(\alpha)$  est un type pour tout  $\alpha$
- ❖ **Thm.** si  $\Gamma \vdash u : G$  dérivable alors pour tout contexte  $\mathcal{C}$ ,  
pour toute  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$ ,  $u\theta \in \text{RED}_G^{\mathcal{C}}$ .
- ❖ **Preuve:** par récurrence sur la dérivation de  $\Gamma \vdash u : G$ ,  
comme les dernières fois;  
on utilise la déf. de  $\text{RED}_{F \Rightarrow G}^{\mathcal{C}}$  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  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$ ,  $u\theta \in \text{RED}_G^{\mathcal{C}}$ .

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

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

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

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



# Normalisation forte

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

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

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

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

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

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

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

# Normalisation forte

- ❖ **Thm.** si  $\Gamma \vdash u : G$  dérivable alors **pour tout contexte  $\mathcal{C}$ ,**  
pour toute  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$ ,  $u\theta \in \text{RED}_G^{\mathcal{C}}$ .
- ❖ **Cas ( $\forall^2\text{I}$ ).** Par h.r., (pour tous  $\mathcal{C}, G, S$ ),  
**pour tout contexte de la forme  $\mathcal{C}[\alpha \mapsto S]$ ,**  
 $u(\theta + [\alpha := G]) \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$ .

$$\frac{\Gamma \vdash u : F}{\Gamma \vdash \lambda \alpha . u : \forall \alpha . F} (\forall^2\text{I})$$

(si  $\alpha$  pas libre dans aucune des formules de  $\Gamma$ )



# Normalisation forte

- ❖ **Thm.** si  $\Gamma \vdash u : G$  dérivable alors **pour tout contexte  $\mathcal{C}$ ,**  
pour toute  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$ ,  $u\theta \in \text{RED}_G^{\mathcal{C}}$ .
- ❖ **Cas ( $\forall^2\text{I}$ ).** Par h.r., (pour tous  $\mathcal{C}, G, S$ ),  
**pour tout contexte de la forme  $\mathcal{C}[\alpha \mapsto S]$ ,**  
 $u(\theta + [\alpha := G]) \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$ .
- ❖ I.e., pour tout  $\mathcal{C}$ , pour tous  $G$  et  $S$ ,  $u\theta[\alpha := G] \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$

$$\frac{\Gamma \vdash u : F}{\Gamma \vdash \lambda \alpha . u : \forall \alpha . F} (\forall^2\text{I})$$

(si  $\alpha$  pas libre dans aucune des formules de  $\Gamma$ )

# Normalisation forte

- ❖ **Thm.** si  $\Gamma \vdash u : G$  dérivable alors **pour tout contexte  $\mathcal{C}$ ,**  
pour toute  $\theta \in \underline{\text{RED}}_{\Gamma}^{\mathcal{C}}$ ,  $u\theta \in \text{RED}_G^{\mathcal{C}}$ .
- ❖ **Cas ( $\forall^2\text{I}$ ).** Par h.r., (pour tous  $\mathcal{C}, G, S$ ),  
**pour tout contexte de la forme  $\mathcal{C}[\alpha \mapsto S]$ ,**  
 $u(\theta + [\alpha := G]) \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$ .
- ❖ I.e., pour tout  $\mathcal{C}$ , pour tous  $G$  et  $S$ ,  $u\theta[\alpha := G] \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$
- ❖ Donc  $\Lambda\alpha . u\theta \in \text{RED}_{\forall\alpha . F}^{\mathcal{C}}$ .  $\square$

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

(si  $\alpha$  pas libre dans aucune des formules de  $\Gamma$ )

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



# Normalisation forte

- ❖ **Thm.** si  $\Gamma \vdash u : F$  pour tout contexte  $\mathcal{C}$ ,  
 pour tout  $G$  et tout candidat  $S$ ,  
 $u\theta + [\alpha := G] \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$ .  
 Je vous laisse vérifier les conditions sur les variables libres et liées...

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

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

(si  $\alpha$  pas libre dans aucune des formules de  $\Gamma$ )

- ❖ I.e., pour tout  $\mathcal{C}$ , pour tous  $G$  et  $S$ ,  $u\theta[\alpha := G] \in \text{RED}_F^{\mathcal{C}[\alpha \mapsto S]}$

- ❖ Donc  $\Lambda \alpha . u\theta \in \text{RED}_{\forall \alpha . F}^{\mathcal{C}}$ .  $\square$

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

---

# Normalisation forte

---

- ❖ **Thm.** Tout terme typable en système  $F_2$  est fortement normalisable.
- ❖ Voilà!



# Normalisation forte

- ❖ **Thm.** Tout terme typable en système  $F_2$  est fortement normalisable.
- ❖ Voilà!
- ❖ Ceci implique la cohérence de  $HA_2$ ...



Jean-Yves Girard

**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  $\rightarrow$  sont considérés comme implicationnels) et de quatre schémas de construction de fonctionnelles 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**  $A ::= P(e_1, \dots, e_n)$   
mais ici  $P/n$  peut être quantifiée!
- ❖ Formules  $F, G, \dots ::= A$  (formules atomiques)
  - |  $F \Rightarrow G$  (comme en logique prop.)
  - |  $\forall i . F$  (premier ordre)
  - |  $\forall P/n . F$  (second ordre)

---

# Substitution

---

- ❖ **Substitution du 1er ordre:**  $F[i:=e]$  (comme d'habitude)



# Substitution

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

- ❖ **Substitution du 1er ordre:**  $F[i:=e]$  (comme d'habitude)
- ❖ **Substitution du 2nd ordre:**  $F[P:=(i_1, \dots, i_n) \mapsto G]$   
définie par récurrence sur  $F$ ...

# Substitution

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

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



# Substitution

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

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

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

# Substitution

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

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

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)

$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{ (}\Rightarrow\text{E)}$	$\frac{}{\Gamma, x:F \vdash x : F} \text{ (Ax)}$
$\frac{\Gamma \vdash u : \forall i . G}{\Gamma \vdash ue : G[i:=e]} \text{ (}\forall\text{E)}$	$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{ (}\Rightarrow\text{I)}$
$\frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i . u : \forall i . G} \text{ (}\forall\text{I)}$	$\frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i . u : \forall i . G} \text{ (}\forall\text{I)}$ <p style="text-align: center; font-size: small;">(si <math>i</math> pas libre dans [aucune des formules de] <math>\Gamma</math>)</p>

$\frac{\Gamma \vdash u : \forall P_{/n} . F}{\Gamma \vdash uG : F[P:=(i_1, \dots, i_n) \mapsto G]} \text{ (}\forall^2\text{E)}$	$\frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda P_{/n} . u : \forall P_{/n} . G} \text{ (}\forall^2\text{I)}$ <p style="text-align: center; font-size: small;">(si <math>P</math> pas libre dans [aucune des formules de] <math>\Gamma</math>)</p>
---	--

---

# Réductions

---

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

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

$$(B^2) \quad (\Lambda P . u)G \rightarrow u[P:=(i_1, \dots, i_n) \vdash G]$$

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



# Effacement du 1er ordre

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

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

(si  $i$  pas libre dans [aucune des formules de]  $\Gamma$ )

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

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )

# Effacement du 1er ordre

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

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

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )



# Effacement du 1er ordre

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

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

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )

# Effacement du 1er ordre

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

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

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )



# Effacement du 1er ordre

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

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

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )

# Effacement du 1er ordre

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

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

Systeme F!

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

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )



---

# Normalisation forte

---

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

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

$$(B) \quad (\lambda i. u)e \rightarrow u[i:=e]$$

$$(B^2) \quad (\lambda P. u)G \rightarrow u[P:=(i_1, \dots, i_n) \mapsto G]$$

# Normalisation forte

- ❖ **Observation clé:** Si  $\Gamma \vdash u : F$  au 2nd ordre,  
alors  $E(\Gamma) \vdash E(u) : E(F)$  en **systeme F**.
- ❖ Si  $u \rightarrow v$  par  $(\beta)$  ou  $(B^2)$ , alors  $E(u) \rightarrow E(v)$  par  $(\beta)$  ou  $(B^2)$   
Si  $u \rightarrow v$  par  $(B)$ , alors  $E(u) = E(v)$

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

$$(B) \quad (\lambda i . u)e \rightarrow u[i:=e]$$

$$(B^2) \quad (\lambda P . u)G \rightarrow u[P:=(i_1, \dots, i_n) \mapsto G]$$



# Normalisation forte

- ❖ **Observation clé:** Si  $\Gamma \vdash u : F$  **au 2nd ordre**,  
alors  $E(\Gamma) \vdash E(u) : E(F)$  en **systeme F**.
- ❖ Si  $u \rightarrow v$  par  $(\beta)$  ou  $(B^2)$ , alors  $E(u) \rightarrow E(v)$  par  $(\beta)$  ou  $(B^2)$   
Si  $u \rightarrow v$  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  $(\beta)+(B^2)\dots$  car le systeme F termine

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

$(B) \quad (\lambda i . u)e \rightarrow u[i:=e]$

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

# Normalisation forte

- ❖ **Observation clé:** Si  $\Gamma \vdash u : F$  au 2nd ordre,  
alors  $E(\Gamma) \vdash E(u) : E(F)$  en **systeme F**.
- ❖ Si  $u \rightarrow v$  par  $(\beta)$  ou  $(B^2)$ , alors  $E(u) \rightarrow E(v)$  par  $(\beta)$  ou  $(B^2)$   
Si  $u \rightarrow v$  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  $(\beta)+(B^2)\dots$  car le systeme F termine
- ❖ A partir d'un certain rang, seulement  $(B)\dots$  qui termine.  $\square$

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

$$(B) \quad (\lambda i . u)e \rightarrow u[i:=e]$$

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



---

# Cohérence

---

- ❖ **Thm.** Les  $\lambda$ -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  $\vdash \perp$ , où  $\perp = \forall P_{/0} . P()$ .)

# Arithmétique d'ordre 2 ( $PA_2$ , $HA_2$ )

(sans les détails)



# PA<sub>2</sub> = 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,  $\forall i, j . i \approx j \Rightarrow F \Rightarrow F[i:=j]$ )

❖  $\forall i . \neg 0 \approx s(i)$

❖  $\forall i, j . s(i) \approx s(j) \Rightarrow i \approx j$

❖  $\forall i . i + 0 \approx i$

❖  $\forall i, j . i + s(j) \approx s(i + j)$

❖  $\forall i . i * 0 \approx 0$

❖  $\forall i, j . i * s(i) \approx i + i$

pour chaque  
formule  $F$

❖ Principe de récurrence:  $F(0) \Rightarrow (\forall j . F(j) \Rightarrow F(s(j))) \Rightarrow \forall j . F(j)$

---

# Note

---

- ❖  $\mathbf{PA}_2$  = arithmétique de Peano du **2nd** ordre  
= log. classique du **2nd** ordre + Peano + récurrence
- ❖  $\mathbf{HA}_2$  = 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  $\mathbf{PA}_2$ !



# HA<sub>2</sub>:

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

$$\frac{\Gamma \vdash u : F \Rightarrow G \quad \Gamma \vdash v : F}{\Gamma \vdash uv : G} \text{(}\Rightarrow\text{E)}$$

$$\frac{\Gamma, x:F \vdash u : G}{\Gamma \vdash \lambda x.u : F \Rightarrow G} \text{(}\Rightarrow\text{I)}$$

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

$$\frac{\Gamma \vdash u : G}{\Gamma \vdash \Lambda i . u : \forall i . G} \text{(}\forall\text{I)}$$

(si  $i$  pas libre dans [aucune des formules de]  $\Gamma$ )

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

$$\frac{\Gamma \vdash u : F}{\Gamma \vdash \Lambda P_{/n} . u : \forall P_{/n} . G} \text{(}\forall^2\text{I)}$$

(si  $P$  pas libre dans [aucune des formules de]  $\Gamma$ )

+

$$\frac{}{\Gamma \vdash r_0 : 0 \approx 0} \text{(Refl}_0\text{)}$$

$$\frac{\Gamma \vdash u : F_1 \quad F_1 \leftrightarrow_{\mathbb{N}}^* F_2}{\Gamma \vdash u : F_2} \text{(}\leftrightarrow_{\mathbb{N}}^*\text{)}$$

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

---

# 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



---

# 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:

$$\forall i . \mathbf{nat}(i) \Rightarrow G$$

---

# 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:
$$\forall i . \mathbf{nat}(i) \Rightarrow G$$
- ❖ où  $\mathbf{nat}(e)$  abrège:  $\forall P_{/1} . (\forall j . P(j) \Rightarrow P(s(j))) \Rightarrow P(0) \Rightarrow P(e)$



# 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:  
 $\forall i . \mathbf{nat}(e)$   
où  $\mathbf{nat}(e)$  abrège:  $\forall P_{/1} . (\forall j . P(j) \Rightarrow P(s(j))) \Rightarrow P(0) \Rightarrow P(e)$   
Un entier, c'est un objet  $e$  qui vérifie un **principe de récurrence local**

---

# La 1ère astuce de Girard

---

la relativisée de  $F$

- ❖ Girard observe que si  $\vdash F$  prouvable en  $\mathbf{HA}_2$ , alors  $\vdash \text{rel}(F)$  est prouvable en logique du 2nd ordre + ax. Peano (sans principe de récurrence)
- ❖ **Note:**  $\text{rel}(\ll \text{faux} \gg) = \ll \text{faux} \gg$



# La 1ère astuce de Girard

la relativisée de  $F$

- ❖ Girard observe que si  $\vdash F$  prouvable en  $\mathbf{HA}_2$ , alors  $\vdash \text{rel}(F)$  est prouvable en logique du 2nd ordre + ax. Peano (sans principe de récurrence)
- ❖ **Note:**  $\text{rel}(\ll \text{faux} \gg) = \ll \text{faux} \gg$

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

# La 1ère astuce de Girard

la relativisée de  $F$

- ❖ Girard observe que si  $\vdash F$  prouvable en  $\mathbf{HA}_2$ , alors  $\vdash \text{rel}(F)$  est prouvable en logique du 2nd ordre + ax. Peano (sans principe de récurrence)

- ❖ **Note:**  $\text{rel}(\ll \text{faux} \gg) = \ll \text{faux} \gg$

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

$$\perp = \forall \alpha . \alpha$$



# La 1ère astuce de Girard

la relativisée de  $F$

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

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

- ❖ Note. Ici (« faux ») = « faux »

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

$$\perp = \forall \alpha . \alpha$$

---

# Effacement du 1er ordre

(la 2ème astuce de Girard)

---

- ❖ Mais, par effacement du premier ordre, les  $\lambda$ -termes de preuve de la logique du 2nd ordre + ax. de Peano (qui disparaissent tous!) **terminent**  
(Ça marche aussi avec  $(\text{Refl}_0)$  + preuve modulo  $(\longleftrightarrow^*_{\mathbb{N}})$  à la place des axiomes de Peano.)



---

# Effacement du 1er ordre

(la 2ème astuce de Girard)

---

- ❖ Mais, par effacement du premier ordre, les  $\lambda$ -termes de preuve de la logique du 2nd ordre + ax. de Peano (qui disparaissent tous!) **terminent**  
(Ça marche aussi avec  $(\text{Refl}_0)$  + preuve modulo  $(\longleftrightarrow^*_{\mathbb{N}})$  à la place des axiomes de Peano.)
- ❖ Par un argument usuel d'examen de formes normales, cette dernière est cohérente.

---

# Effacement du 1er ordre

(la 2ème astuce de Girard)

---

- ❖ Mais, par effacement du premier ordre, les  $\lambda$ -termes de preuve de la logique du 2nd ordre + ax. de Peano (qui disparaissent tous!) **terminent**  
(Ça marche aussi avec  $(\text{Refl}_0)$  + preuve modulo  $(\longleftrightarrow^*_{\mathbb{N}})$  à la place des axiomes de Peano.)
- ❖ Par un argument usuel d'examen de formes normales, cette dernière est cohérente.
- ❖ Donc **HA<sub>2</sub>** aussi!



# Cohérence de l'arithmétique

- ❖ **Thm.** Il n'y a pas de preuve de  $\vdash \perp$  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<sup>1)</sup>.

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 sendsten derzeit aufgestellten formalen Systeme sind das System Principia Mathematica (PM)<sup>2)</sup> einerseits, das Zermelo-Fraenkel'sche (von J. v. Neumann weiter ausgebildete) Axiomensystem Mengenlehre<sup>3)</sup> andererseits. Diese beiden Systeme sind so 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 Aussagen in den betreffenden Systemen überhaupt formal auslassen, auch zu entscheiden. Im folgenden wird gezeigt, daß nicht der Fall ist, sondern daß es in den beiden angeführten Systemen sogar relativ einfache Probleme aus der Theorie der natürlichen ganzen Zahlen gibt<sup>4)</sup>, die sich aus den Axiomen



---

# Quoi de finitiste dans notre preuve?

---

- ❖ C'est quelque chose dans notre preuve de terminaison qui ne l'est pas... mais quoi?



---

# 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**  $RED_F^e \dots$  mais ce n'est pas si simple.

---

# 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  $RED_F^c$ ...  
mais ce n'est pas si simple.
- ❖ Au lieu d'écrire 'soit  $u \in RED_{F \Rightarrow G}^c$ '  
on peut écrire  
'soit  $u$  tel que (pour tout  $v \in RED_F^c$ ,  $uv \in RED_G^c$ )'  
et ...



---

# Quoi de finitiste dans notre preuve?

---

❖ Et au lieu d'écrire 'soit  $u \in \text{RED}_{\forall\alpha . F^c}$ '

on peut écrire

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

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

# Quoi de finitiste dans notre preuve?

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

❖ Et au lieu d'écrire 'soit  $u \in \text{RED}_{\forall\alpha}$ .

on peut écrire

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

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



---

# Quoi de finitiste dans notre preuve?

---

- ❖ En fait, si je vous donne une dérivation d'un jugement **fixé**  $\Gamma \vdash u : F$  en  $\mathbf{HA}_2$ ,

---

# 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...



---

# 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 inclinant toutes les mentions d'ensembles  $\text{RED}_G^c$  et  $\text{SN}$

---

# 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 inclinant toutes les mentions d'ensembles  $\text{RED}_G^c$  et  $\text{SN}$
- ❖ et cette preuve est formalisable en  $\mathbf{HA}_2$



---

# Quoi de finitiste dans notre preuve?

---

- ❖ La seule chose non formalizable 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.

---

# Quoi de finitiste dans notre preuve?

---

- ❖ La seule chose non formalizable 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 **1-cohérente**:  
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  $\lambda$ -calcul typé **en système F**  
+ **récurrence primitive à tous les types**

---

# 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  $\lambda$ -calcul typé **en système F**  
~~+ récurrence primitive à tous les types~~

Pas besoin!  
(1ère astuce de Girard)



---

# Note: « $\exists$ » en logique du 2nd ordre

---

- ❖ ... est définissable, comme en système F:

$$\exists P/n . F = \forall \beta . (\forall P/n . F \Rightarrow \beta) \Rightarrow \beta$$

- ❖ et pareil pour le « et », le « ou », le « non », etc.

---

# Fonctions prouvablement totales

---

- ❖ Une **fonction prouvablement totale** (en  $\mathbf{HA}_2$ )  
est une formule  $R(i,j)$   
telle qu'on peut prouver  $\vdash \forall i . \exists j . R(i,j)$  en  $\mathbf{HA}_2$ .



---

# Fonctions prouvablement totales

---

- ❖ Une **fonction prouvablement totale** (en  $\mathbf{HA}_2$ )  
est une formule  $R(i,j)$   
telle qu'on peut prouver  $\vdash \forall i . \exists j . R(i,j)$  en  $\mathbf{HA}_2$ .
- ❖ Alors on peut prouver sa **relativisée**:  
 $\vdash u : \forall i . \mathbf{nat}(i) \Rightarrow \exists j . \mathbf{nat}(j) \wedge \mathbf{rel}(R(i,j))$   
en logique du 2nd ordre + axiomes de Peano.

---

# Fonctions prouvablement totales

---

- ❖ Une **fonction prouvablement totale** (en  $\mathbf{HA}_2$ )  
est une formule  $R(i,j)$   
telle qu'on peut prouver  $\vdash \forall i . \exists j . R(i,j)$  en  $\mathbf{HA}_2$ .
- ❖ Alors on peut prouver sa **relativisée**:  
 $\vdash u : \forall i . \mathbf{nat}(i) \Rightarrow \exists j . \mathbf{nat}(j) \wedge \mathbf{rel}(R(i,j))$   
en logique du 2nd ordre + axiomes de Peano.
- ❖ Par effacement du premier ordre, on obtient un terme  
 $\vdash E(u) : \mathbf{nat} \Rightarrow \mathbf{nat} \wedge E(\mathbf{rel}(R(i,j)))$  typé **en système F**



# Fonctions provablement totales

- Eh oui,
- ❖ Une telle fonction  $\text{nat}$  est l'effacement du premier ordre du principe de récurrence local  
 $\text{nat} = \forall \alpha . (\alpha \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha)$  (système F)
  - ❖ Alors  $\text{nat}(e) = \forall P_{/1} . (\forall j . P(j) \Rightarrow P(s(j))) \Rightarrow P(0) \Rightarrow P(e)$
  - ❖  $\vdash u : \forall i . \text{nat}(i) \Rightarrow \exists j . \text{nat}(j) \wedge \text{rel}(R(i,j))$   
en logique du 2nd ordre + axiomes de Peano.
  - ❖ Par effacement du premier ordre, on obtient un terme  $\vdash E(u) : \text{nat} \Rightarrow \text{nat} \wedge E(\text{rel}(R(i,j)))$  typé **en système F**

---

# Fonctions provablement totales

---

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



---

# Fonctions provablement totales

---

- ❖ Par effacement du premier ordre, on obtient un terme  $\vdash E(u) : \mathbf{nat} \Rightarrow \mathbf{nat} \wedge E(\text{rel}(R(i,j)))$  typé **en système F**
- ❖ et donc un terme  $\vdash \lambda x . \pi_1(E(u)(x)) : \mathbf{nat} \Rightarrow \mathbf{nat}$

---

# Fonctions provablement totales

---

- ❖ Par effacement du premier ordre, on obtient un terme  $\vdash E(u) : \mathbf{nat} \Rightarrow \mathbf{nat} \wedge E(\text{rel}(R(i,j)))$  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  $i$ ,  
$$(\lambda x . \pi_1(E(u)(x))) \ulcorner i \urcorner \rightarrow^* \ulcorner j \urcorner$$
où  $\vdash R(i,j)$  est prouvable en  $\mathbf{HA}_2$ .



---

# Fonctions provablement totales

---

- ❖ Par effacement du premier ordre, on obtient un terme  $\vdash E(u) : \mathbf{nat} \Rightarrow \mathbf{nat} \wedge E(\text{rel}(R(i,j)))$  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  $i$ ,  
$$(\lambda x . \pi_1(E(u)(x))) \ulcorner i \urcorner \rightarrow^* \ulcorner j \urcorner$$
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  $u : \forall i . \mathbf{nat}(i) \Rightarrow \exists j . \mathbf{nat}(j) \wedge \text{rel}(R(i,j))$  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:



---

# 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.  $u^{\ulcorner i \urcorner} \rightarrow^* \ulcorner j \urcorner$ ,  
**provablement en HA<sub>2</sub>**:
  - la réduction  $u^{\ulcorner i \urcorner} \rightarrow^* \dots$  termine,  
provablement en **HA<sub>2</sub>** (voir argument d'inlining des RED<sub>F</sub><sup>e</sup>)
  - une forme normale close de type **nat**  
est un entier de Church (à un détail près, voir TD)

---

# Fonctions prouvablement 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.  $u \ulcorner i \urcorner \rightarrow^* \ulcorner j \urcorner$ ,  
**prouvablement en HA<sub>2</sub>**:
    - la réduction  $u \ulcorner i \urcorner \rightarrow^* \dots$  termine,  
prouvablement en **HA2** (voir argument d'inlining des RED<sub>F</sub><sup>e</sup>)
    - 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 HA<sub>2</sub>**.  
(Je rappelle: j'omets quelques détails...)



Vers l'infini et au-delà

---

# Arithmétique d'ordre $\omega$

---

- ❖ 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)

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





---

# 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  $\omega$  **polymorphe**)  
est **incohérent** (Coquand, 1994)



---

La prochaine fois

---



---

# La prochaine fois

---

- ❖ Nous parlerons de quelques stratégies d'implémentation du lambda-calcul

---

# La prochaine fois

---

- ❖ Nous parlerons de quelques stratégies d'implémentation du lambda-calcul
- ❖ (machines de Krivine, calculs à substitutions explicites)



---

# 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**

---

# 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. 😊