Jean Goubault-Larrecq

λ-calcul

8. Logique classique propositionnelle

Aujourd'hui

- Suite de notre étude du λ-calcul simplement typé: logique **classique**
- … c'est tout! Ca va nous occuper bien assez, allez…

Logique classique

La logique propositionnelle classique

- Juste deux opérateurs ici: $\Rightarrow$ et $\bot$
- En classique (pas en intuitionniste), on peut définir les autres ($\wedge$, $\vee$, $\neg$) à partir de ces deux-là:

$$\neg F = F \Rightarrow \bot$$
$$F \vee G = (\neg F) \Rightarrow G$$
$$F \wedge G = \neg(F \Rightarrow \neg G)$$

- mais on pourrait aussi utiliser les règles de la dernière fois

Rappelez-vous: logique minimale+faux

- Types (formules):

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

  $\mid F \Rightarrow G$

  $\mid \bot$ (« faux »)

- Termes:

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

  $\mid uv$

  $\mid \lambda x.u$

  $\mid \nabla u$

- Réduction:

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

  $(\nabla)\quad (\nabla u)v \to \nabla u$

du faux on peut déduire n'importe quoi

$$\frac{\Gamma \vdash u : \bot}{\Gamma \vdash \nabla u : G}\ (\bot\,\mathrm{E})$$

élimination du faux
(… et pas d'intro du faux!)

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

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

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

… logique minimale+faux → …

- Types (formules):

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

  $\mid F \Rightarrow G$

  $\mid \bot$ (« faux »)

- Termes:

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

  $\mid uv$

  $\mid \lambda x.u$

  $\mid \mathbf{C}u$

- Réduction:

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

  $(\nabla)\quad (\nabla u)v \to \nabla u$

$$\frac{\Gamma \vdash u : \bot}{\Gamma \vdash \nabla u : G}\,(\bot\,\mathrm{E})$$

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

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

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

… → logique classique

- Types (formules):

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

  $\mid F \Rightarrow G$

  $\mid \bot$ (« faux »)

- Termes:

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

  $\mid uv$

  $\mid \lambda x.u$

  $\mid \mathbf{C}u$

- Réduction:

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

  $(\nabla)\quad (\nabla u)v \to \nabla u$

raisonnement **par l'absurde**

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\,(\neg\neg\mathrm{E})$$

élimination de la double négation (!)
$(\neg\neg G = (G \Rightarrow \bot) \Rightarrow \bot)$

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

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

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

… → logique classique

- Types (formules):

  $F, G, \ldots ::= A \mid F \Rightarrow G \mid \bot$ (« faux »)

- Termes:

  $u, v, \ldots ::= x \mid uv \mid \lambda x.u \mid \mathbf{C}u$

- Réduction:

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

  $(\mathbf{C})\quad (\mathbf{C}u)v \to \mathbf{?}$

raisonnement **par l'absurde**

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

élimination de la double négation (!)
$(\neg\neg G = (G \Rightarrow \bot) \Rightarrow \bot)$

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

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

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

Raisonnement par l'absurde

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\,(\neg\neg\mathrm{E})$$

- Avant de découvrir la règle $(\mathbf{C}u)v \to ?$, parlons de $(\neg\neg\mathrm{E})$
- « pour prouver G, supposons $\neg G$, et déduisons $\bot$ »
  → raisonnement **par l'absurde**
- différent de:
  « pour prouver $\neg G$, supposons G, et déduisons $\bot$ »
  → … qui est prouvable en intuitionniste
  (rappel: $\neg G = G \Rightarrow \bot$, et utilisez $(\Rightarrow\mathrm{I})$)

La règle $(\mathbf{C}u)v \to \ldots$

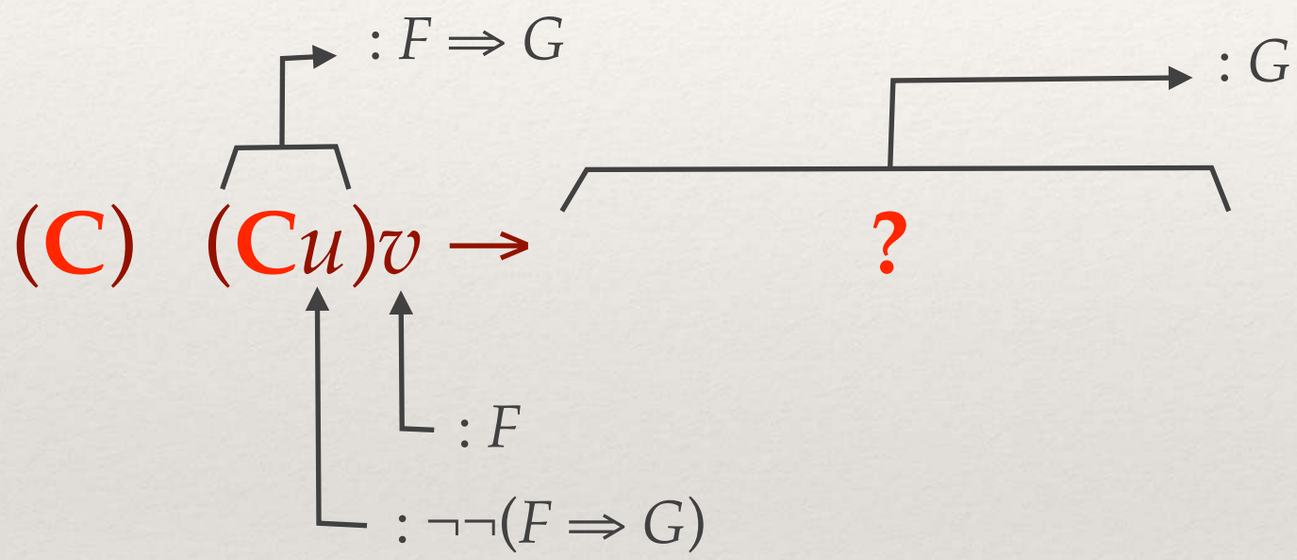
$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\,(\neg\neg\mathrm{E})$$

- Je vous propose de tenter de trouver ‘…’ en cherchant un terme du même type que le côté gauche
- On verra bien ce que ça donne!

La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

- Si on veut typer le côté gauche…

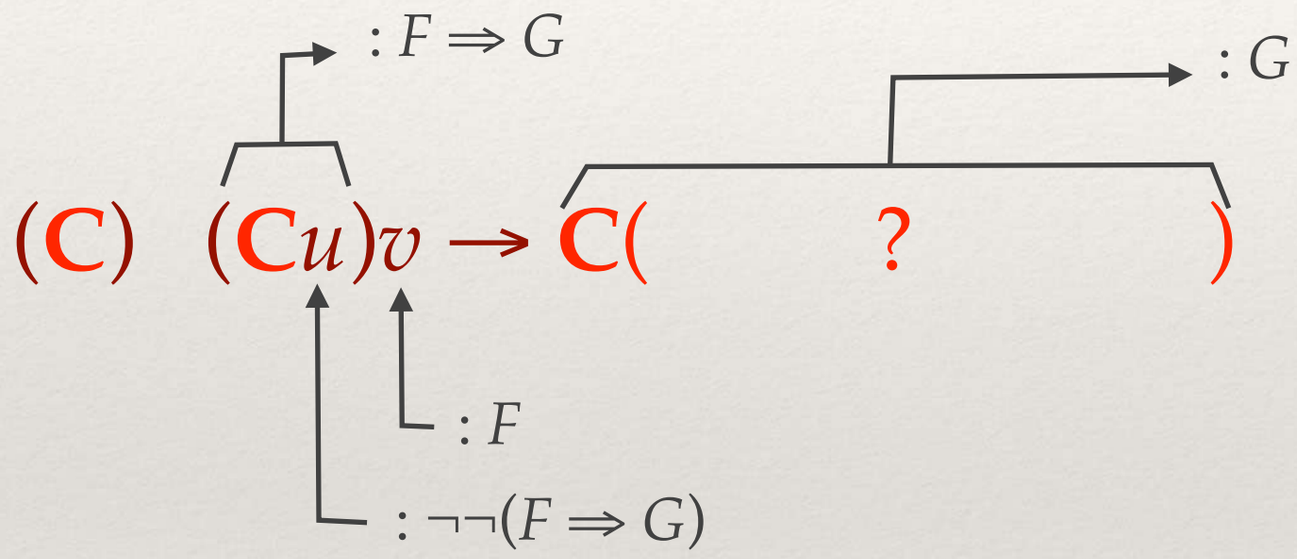


- alors le côté droit doit être de type G

La règle (Cu)$v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

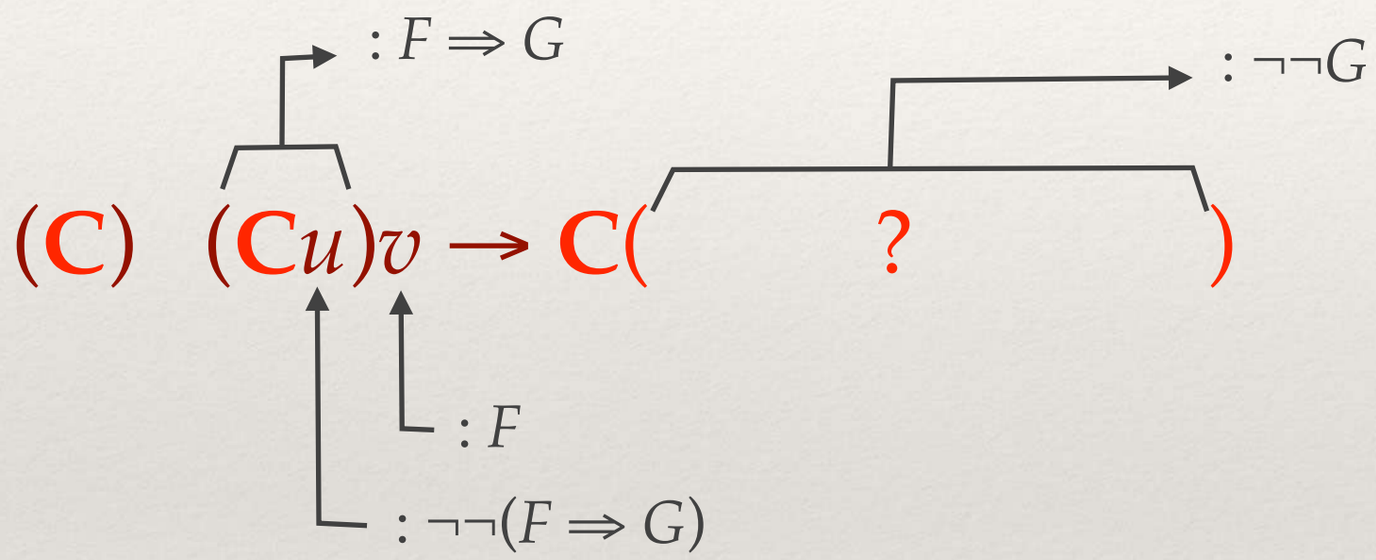
- Cherchons à faire remonter **C**:



La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

- Cherchons à faire remonter **C**:

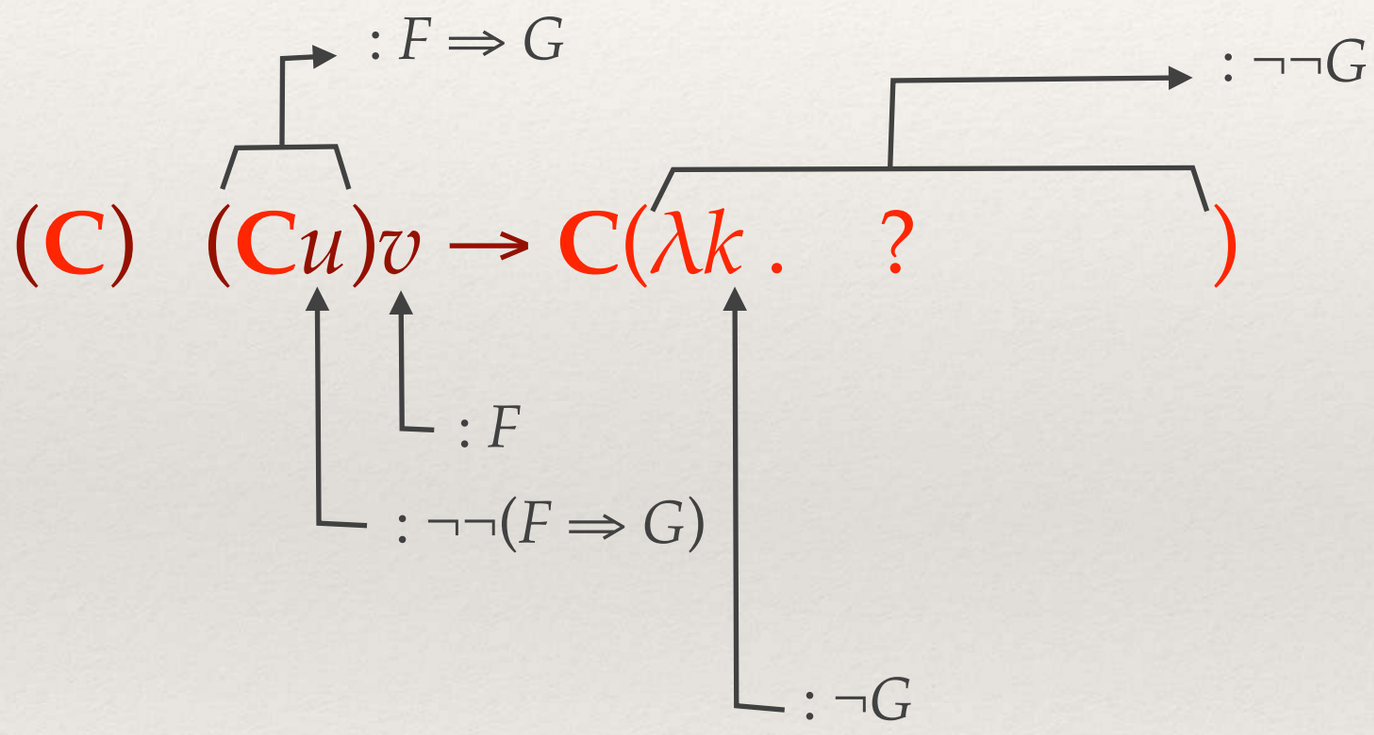


- donc le terme sous **C** doit être de type $\neg\neg G$

La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

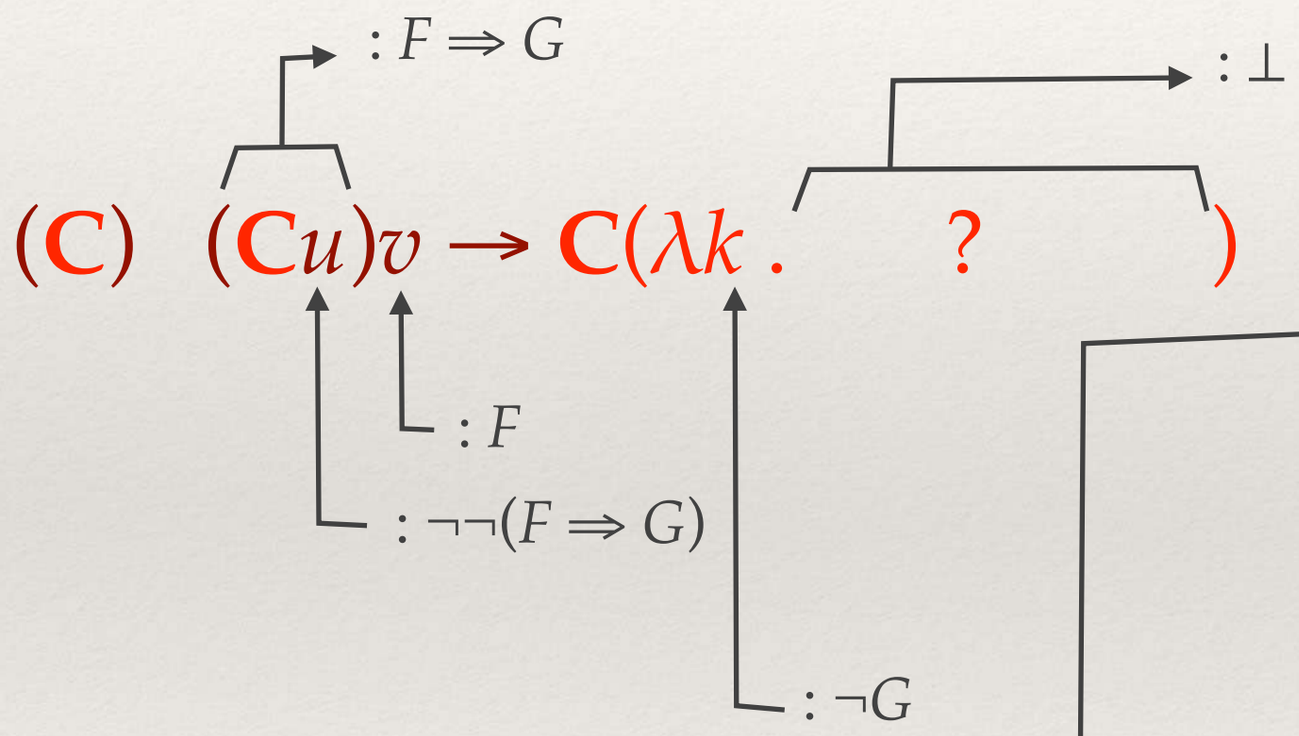
- Introduisons une abstraction



La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

- Introduisons une abstraction



ça, ça ne marchera pas:
il faudrait trouver un
qqch de type G,
et c'est justement
ce qu'on cherche
depuis le début

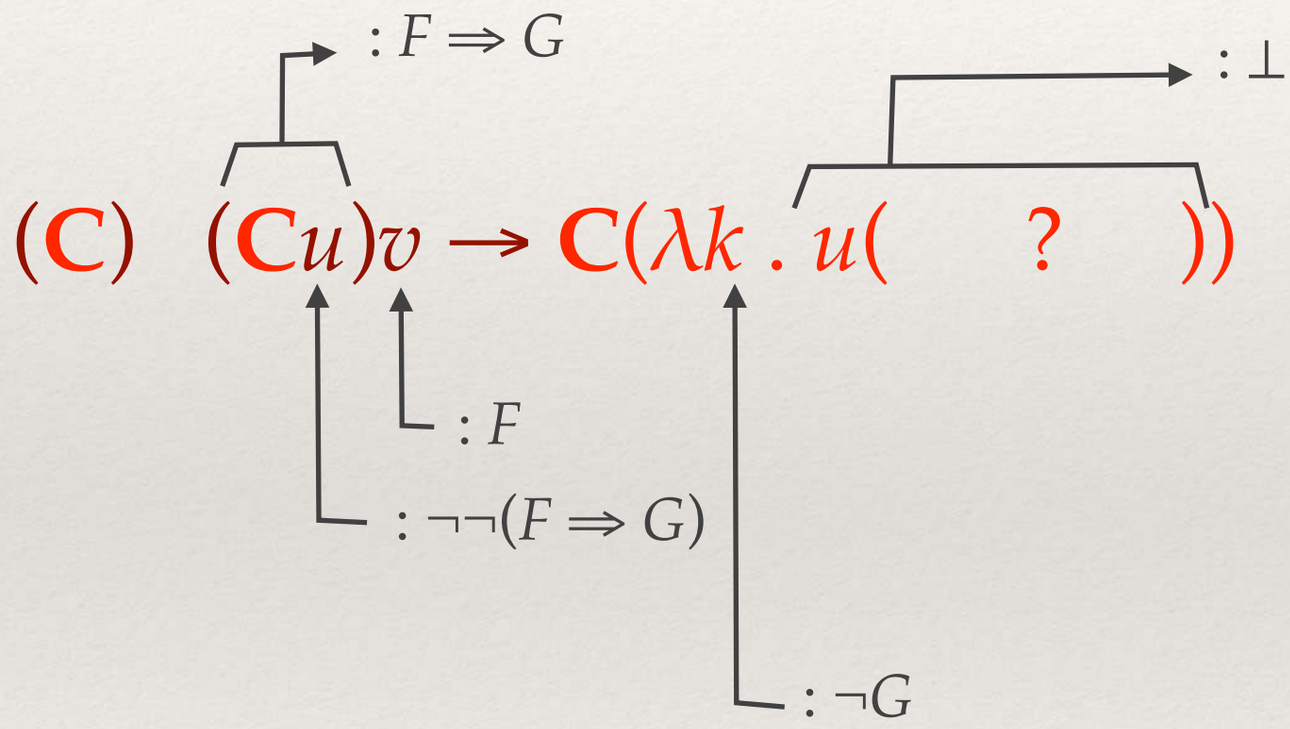
- Ici le terme inconnu peut être ~~k(qqch)~~ ou u(qqch)

rappel: ¬truc = truc ⇒ ⊥,

et on veut un résultat de type ⊥

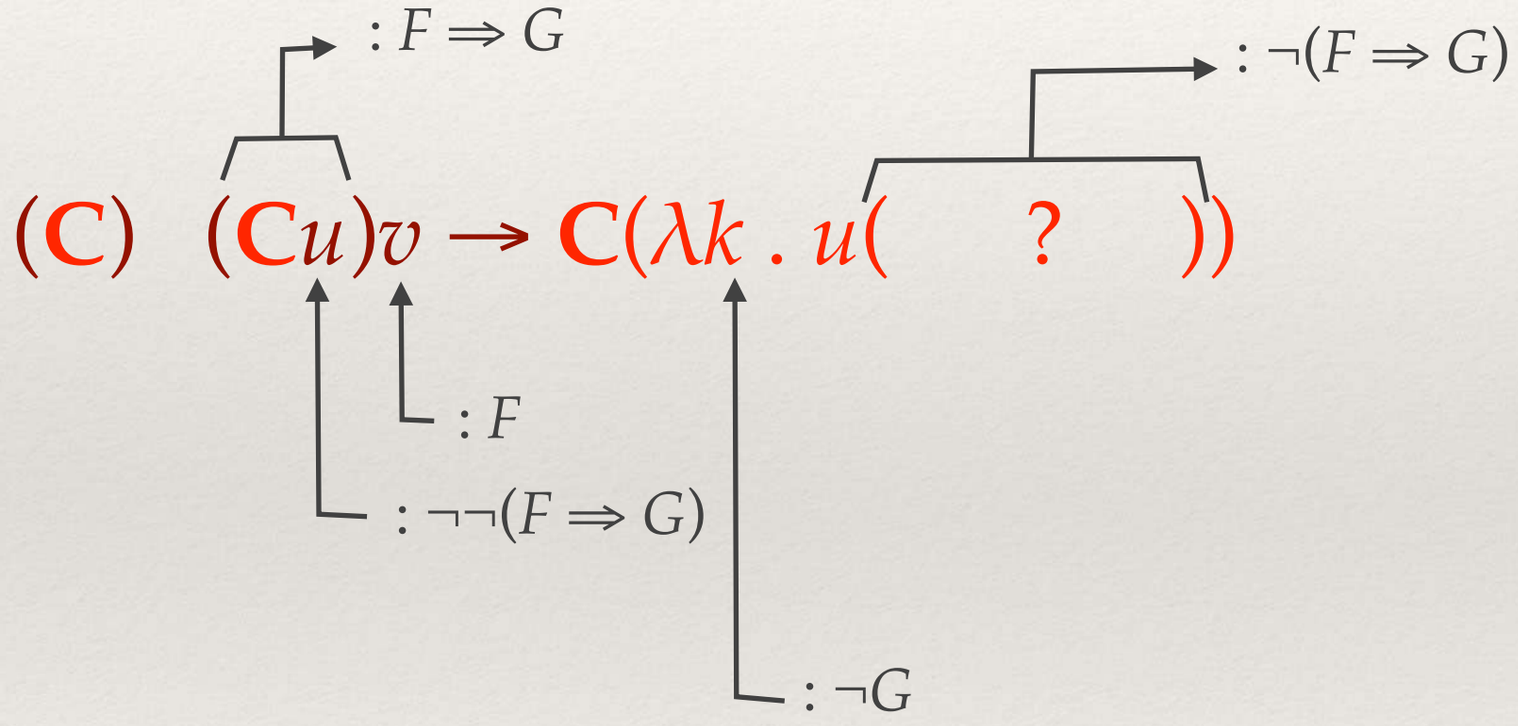
La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$



La règle $(\mathbf{C}u)v \to \ldots$

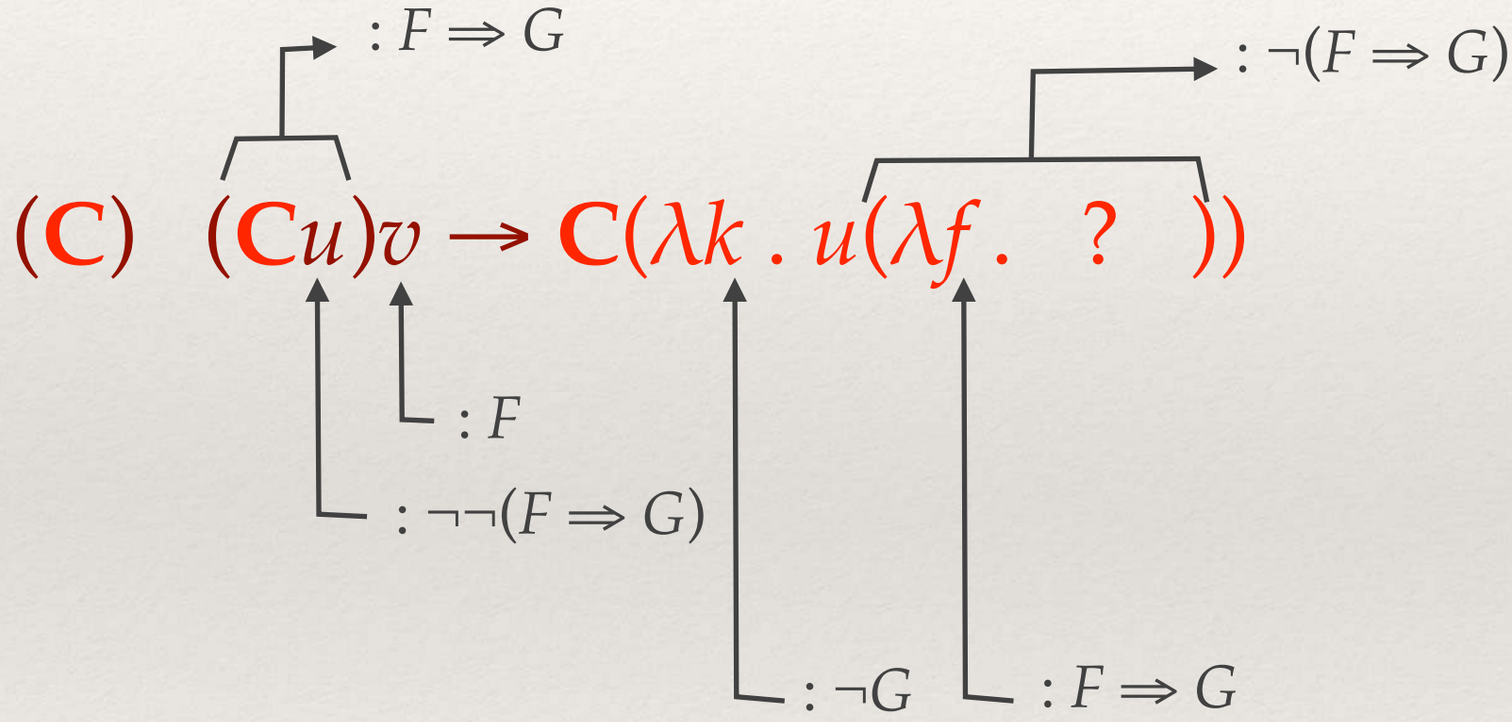
$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$



La règle $(\mathbf{C}u)v \to \ldots$

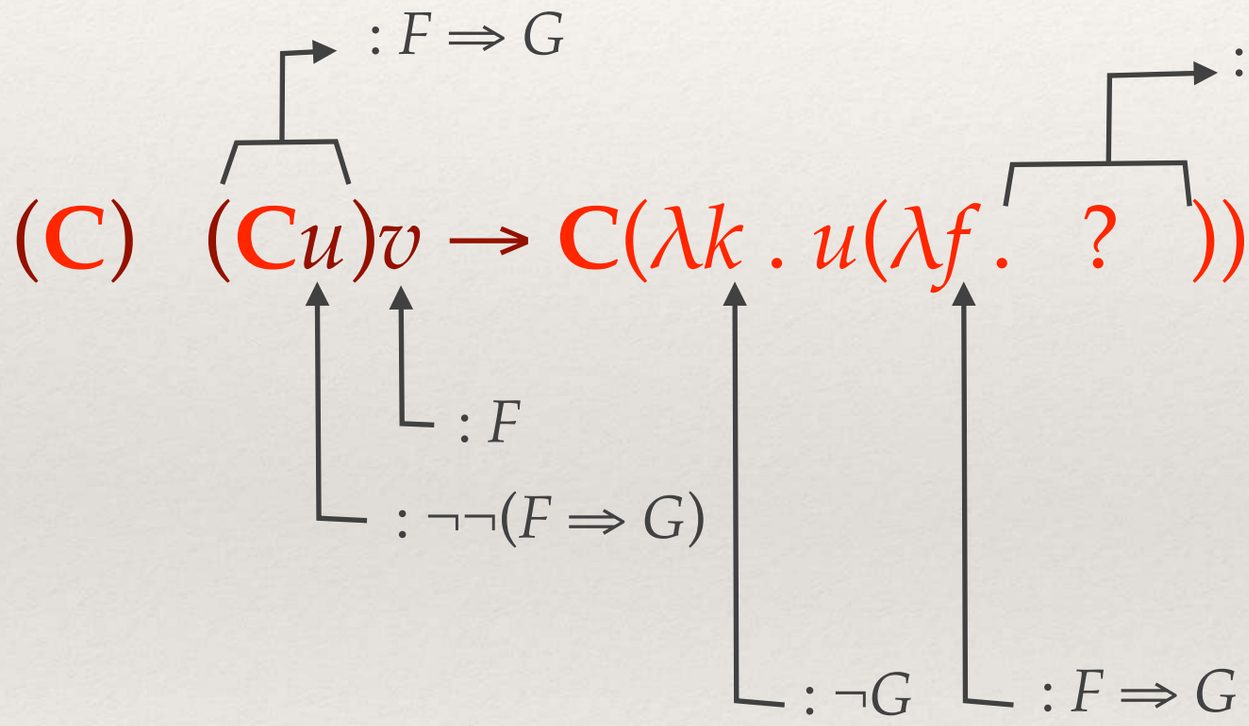
$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

- Introduisons une abstraction



La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$



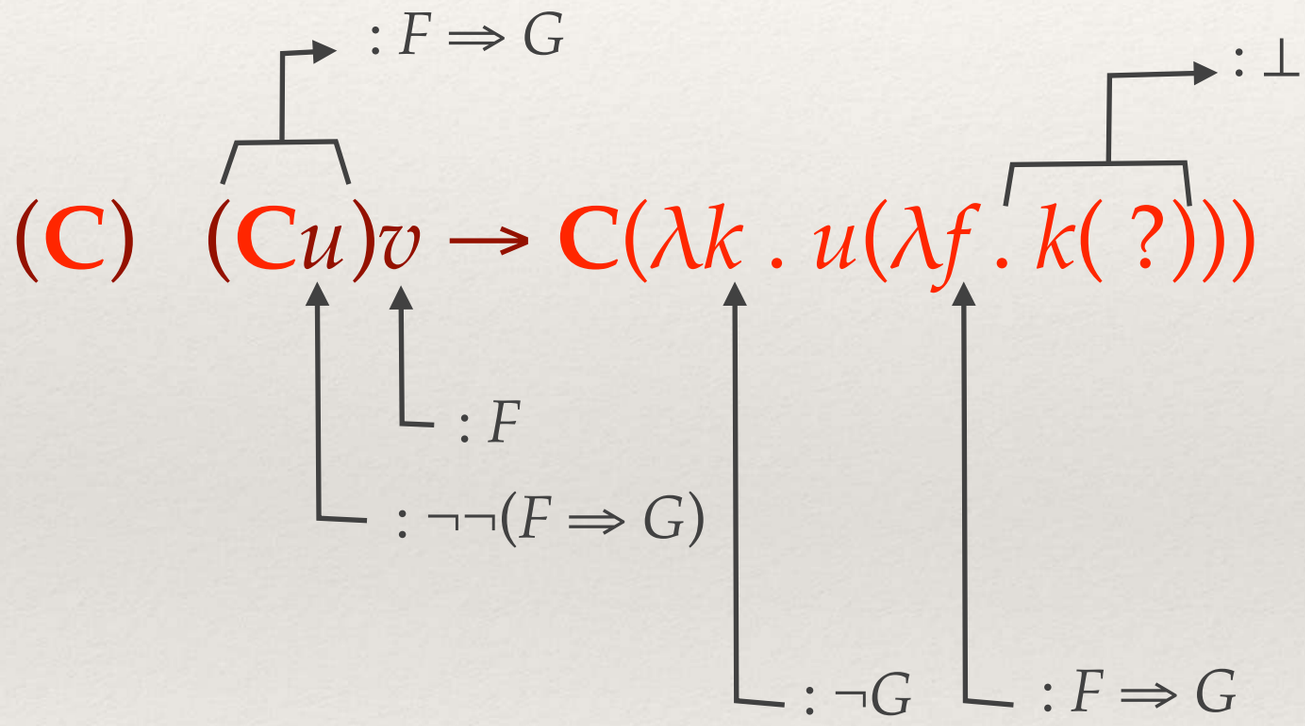
- Ici le terme inconnu peut être k(qqch) par ex., non?

  rappel: ¬truc = truc ⇒ ⊥,

  et on veut un résultat de type ⊥

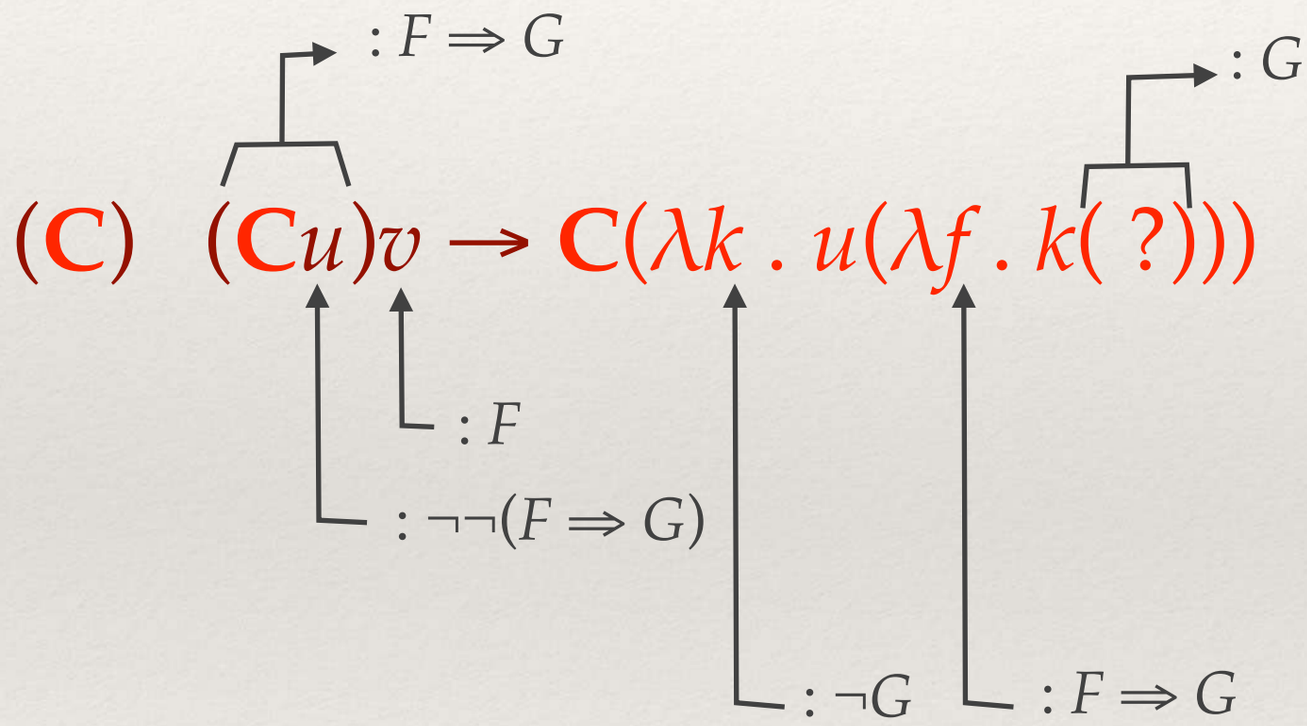
La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$



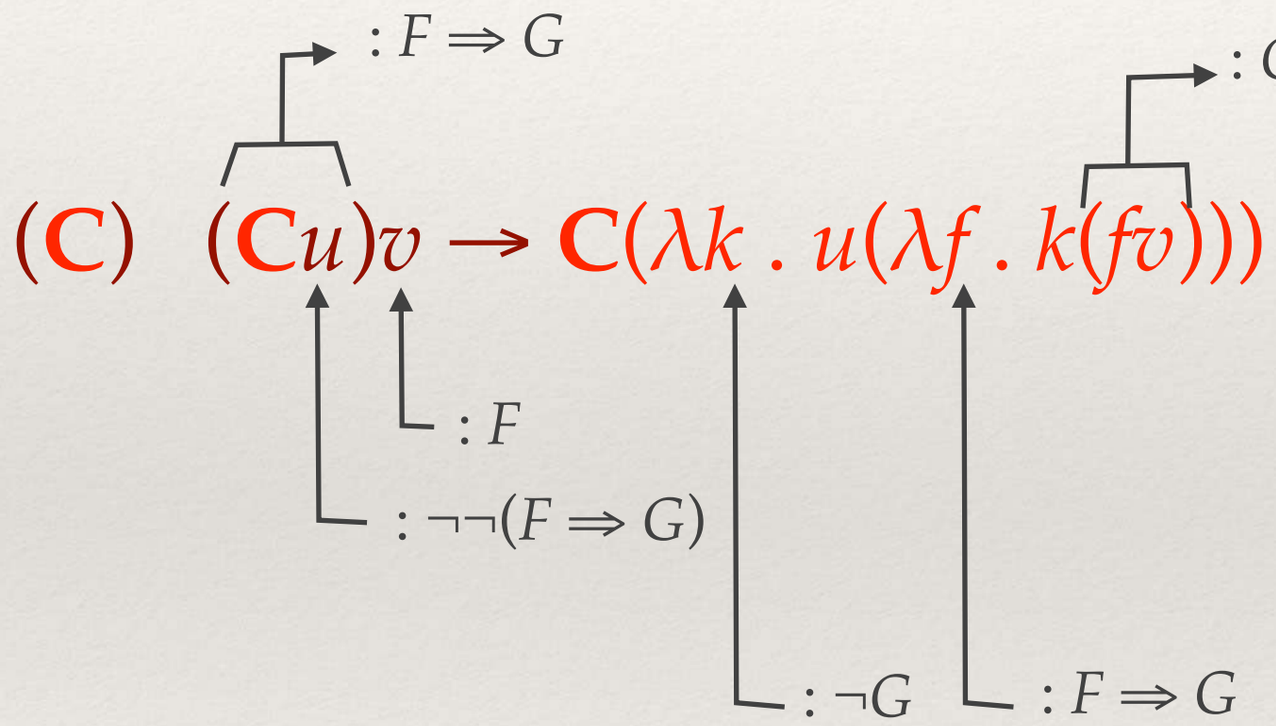
La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$



La règle $(\mathbf{C}u)v \to \ldots$

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$



- Voilà, on a trouvé!

Logique classique

Le λ**C**-calcul

- Types (formules):

  $F, G, \ldots ::= A \mid F \Rightarrow G \mid \bot$ (« faux »)

- Termes:

  $u, v, \ldots ::= x \mid uv \mid \lambda x.u \mid \mathbf{C}u$

- Réduction:

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

  (C) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

!

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\,(\neg\neg\mathrm{E})$$

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

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

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

La η-règle pour **C**

- Dans le poly, en plus de (**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

  j'ai ajouté (η**C**) $\mathbf{C}(\lambda k\,.\,ku) \to u$

  (k non libre dans u)

- Je vais me concentrer sur la première

  par souci de simplicité.

L'élimination du faux

- Avant, nous avions la règle:

$$\frac{\Gamma \vdash u : \bot}{\Gamma \vdash \nabla u : G}\,(\bot\,\mathrm{E})$$

- Elle n'est plus nécessaire.

- On peut **définir**

$$\nabla u = \mathbf{C}(\lambda k.u)$$

(k variable fraîche)

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash u : \bot}{\Gamma, k{:}\neg G \vdash u{:}\bot}\,(\text{affaiblissement})}{\Gamma \vdash \lambda k.u : \neg\neg G}\,(\Rightarrow\mathrm{I})}{\Gamma \vdash \mathbf{C}(\lambda k.u) : G}\,(\neg\neg\mathrm{E})$$

L'élimination du faux

- On peut **définir**

$$\nabla u = \mathbf{C}(\lambda k.u)$$

(k variable fraîche)

(C) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

- et même la réduction est la bonne:

$$\begin{aligned}(\nabla u)v &= \mathbf{C}\,(\lambda k'.\,u)\,v \\ &\to \mathbf{C}\,(\lambda k\,.\,(\lambda k'.\,u)\,(\lambda f\,.\,k(fv))) \quad (\mathbf{C}) \\ &\to \mathbf{C}\,(\lambda k\,.\,u) \quad (\beta)\ (k' \text{ fraîche!}) \\ &= \nabla u\end{aligned}$$

Une parenthèse historique

La découverte de l'opérateur C

- Pendant longtemps, on a cru qu'il n'y avait pas de λ-calcul adapté à la logique classique:

Lemme de Joyal. Toute catégorie cartésienne close (~modèle catégorique de la logique intuitionniste) munie d'un objet dualisant (~ le morphisme $F \to \neg\neg F$ est un isomorphisme) est un préordre (~ **toutes les preuves sont convertibles**).

Oups!

- En ~~1988~~, Timothy G. Griffin s'aperçoit que l'opérateur **C** de Felleisen répond à la [illegible] (± [illegible]lions)

Ben oui, on ne va pas accepter un papier qui résout un problème qu'on croyait impossible…

https://www.cnn.group.cam.ac.uk/directory/dr-timothy-g-griffin/image_normal

On a Formal Correspondence Betw

December 20, 1988

Dear Author:

I am sorry to have to tell you that your paper

On a Formal Correspondence Between A-C-Terms + Classical Proofs

was not among those selected by the program committee for presentation in June. Far more good papers were submitted than the program could possibly accomodate and we were forced to reject papers that were clearly publishable in a journal.

Thank you anyhow for submitting your paper to LICS 89 and I hope that you will attend the meeting in June.

Yours sincerely,

R. Parikh

Rohit Parikh
Program chair

La découverte de l'opérateur C

- Pendant longtemps, on a cru qu'il n'y avait pas de λ-calcul adapté à la logique classique:

Lemme de Joyal. Toute catégorie cartésienne close (~modèle catégorique de la logique intuitionniste) munie d'un objet dualisant (~ le morphisme $F \to \neg\neg F$ est un isomorphisme) est un préordre (~ **toutes les preuves sont convertibles**).

1990

- En ~~1988~~, Timothy G. Griffin s'aperçoit que l'opérateur **C** de Felleisen répond à la question (± celle que nous étudions)

Ouf, publié!

https://www.cnn.group.cam.ac.uk/directory/dr-timothy-g-griffin/image_normal

A Formulae-as-Types Notion of Control

Timothy G. Griffin*
Department of Computer Science
Rice University
Houston, TX 77251-1892

Matthias Felleisen

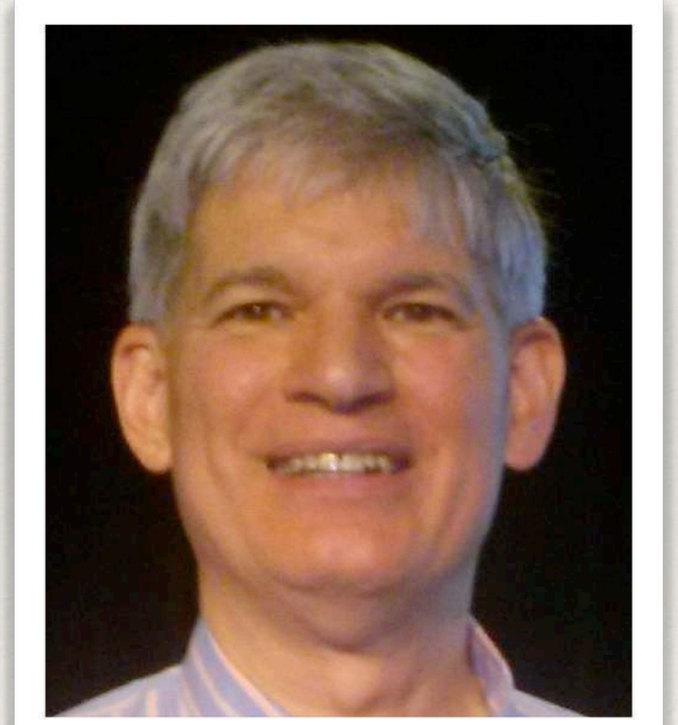
By David Van Horn - DSC_1455, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=29059937

- Ce n'est pas T. Griffin qui a inventé l'opérateur **C**!
- C'est M. Felleisen, qui a passé des années à l'étudier, pour formaliser l'opérateur `call/cc` de Scheme…

Scheme

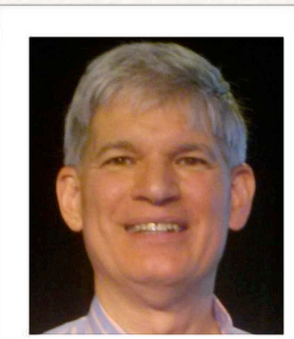
- Scheme est un dialecte Lisp à portée lexicale (donc en clair un λ-calcul étendu)
- inventé par Gerald P. Sussman et Guy L. Steele Jr. dans les années 1970
- qui contenait une primitive mystérieuse `call-with-current-continuation` (`call/cc` pour les intimes)

GLS l'a inventée comme suit…



Par George Ruban — Travail personnel, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45507579

let/cc



Par George Ruban — Travail personnel, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45507

- GLS dit: imaginons que je puisse écrire

  `let/cc` $k\ u$

  pour dire: exécuter u après avoir lié localement la variable k à la **continuation courante**

- … = le contexte d'exécution courant E

  (un terme à un trou, noté entre crochets)

- `let/cc` le **réifie** en une nouvelle constante c_E:

  $$E[\texttt{let/cc}\ \ k\ \ u] \rightarrow E[u[k{:=}c_E]]$$

  (règle exécutée au sommet du terme uniquement)

let/cc

- $E[\texttt{let/cc}\ k\ u] \rightarrow E[u[k{:=}c_E]]$

  (règle exécutée au sommet du terme uniquement)

- … et c_E fonctionne comme une **exception**

  qu'on peut **lancer** par `throw`:

  $E'[\texttt{throw}\ c_E\ a] \rightarrow E[a]$

  (on oublie le contexte d'exécution courant E', et on réinstalle E)

Exemple

$E[\texttt{let/cc}\ k\ (+\ 1\ (\texttt{throw}\ k\ 3))]$

$\rightarrow E[(+\ 1\ (\texttt{throw}\ c_E\ 3))]$

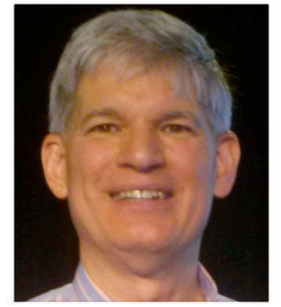
$= E'[\texttt{throw}\ c_E\ 3]$ où $E' = E[(+\ 1\ [])]$

$\rightarrow E[3]$

… autrement dit, l'exception k est lancée avec argument 3 avant qu'on ait la chance d'additionner 1

$E[\texttt{let/cc}\ \ k\ \ u] \rightarrow E[u[k{:=}c_E]]$
$E'[\texttt{throw}\ c_E\ a] \rightarrow E[a]$

Simplification (1/3)



Par George Ruban — Travail personnel, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=45507579

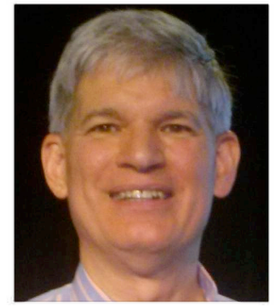
- GLS: plutôt que de créer une nouvelle constante c_E, on peut juste former le terme $\lambda x\ .\ E[x]$

$$E[\texttt{let/cc}\ k\ u] \to E[u[k{:=}c_E]]$$
$$E'[\texttt{throw}\ c_E\ a] \to E[a]$$

- mais alors, on peut changer les règles en:

$$E[\texttt{let/cc}\ k\ u] \to E[u[k{:=}\ \lambda x\ .\ E[x]]]$$
$$E'[\texttt{throw}\ u\ a] \to ua$$

Simplification (2/3)



Par George Ruban — Travail personnel, CC BY-SA 4.0,
https://commons.wikimedia.org/w/index.php?curid=45507579

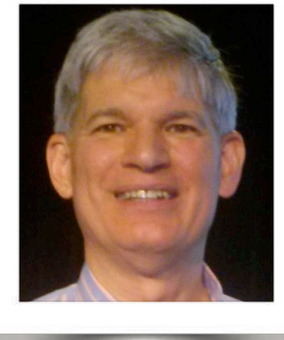
- GLS: `let/cc` lie la variable *k*, mais il serait plus élégant d'utiliser λ pour ça:
  on réécrit `let/cc` $k\ u$
  en `call/cc` $(\lambda k\ .\ u)$

$$E[\texttt{let/cc}\ \ k\ \ u] \rightarrow E[u[k{:=}\ \lambda x\ .\ E[x]]]$$
$$E'[\texttt{throw}\ u\ a] \rightarrow ua$$

- Les règles deviennent:

$$E[\texttt{call/cc}\ \ (\lambda k\ .\ u)] \rightarrow E[u[k{:=}\ \lambda x\ .\ E[x]]]$$
$$E'[\texttt{throw}\ u\ a] \rightarrow ua$$

Simplification (3/3)



Par George Ruban — Travail personnel, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=45507579

- GLS: `call/cc` est maintenant une fonction, donc on peut l'appliquer à un terme quelconque, pas juste une abstraction

$$E[\texttt{call/cc}\ \ (\lambda k\ .\ u)] \to E[u[k{:=}\ \lambda x\ .\ E[x]]]$$
$$E'[\texttt{throw}\ u\ a] \to ua$$

- Les règles deviennent:

$$E[\texttt{call/cc}\ \ u] \to E[u(\lambda x\ .\ E[x])]$$
$$E'[\texttt{throw}\ u\ a] \to ua$$

Un cas particulier

$E[\texttt{call/cc}\ u] \to E[u(\lambda x\,.\,E[x])]$
$E'[\texttt{throw}\ u\ a] \to ua$

- En appel par nom
  (réduction de tête faible),
  les contextes E sont des compositions de contextes élémentaires $[\,]\ v$ d'application à des arguments v

By David Van Horn - DSC_1455, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=29059937

- Felleisen décompose la première règle en:

$$(\texttt{call/cc}\ u)\ v \to \texttt{call/cc}(\lambda k.u(\lambda f.k(fv))v)$$

(E=[ ] v, x est renommée en f; on garde un $\texttt{call/cc}(\lambda k\ldots)$

à droite qui mangera le reste du contexte)

plus une règle qui effacera le `call/cc` au sommet du terme

(je ne décrirai pas toutes les étapes par lesquelles il passe…)

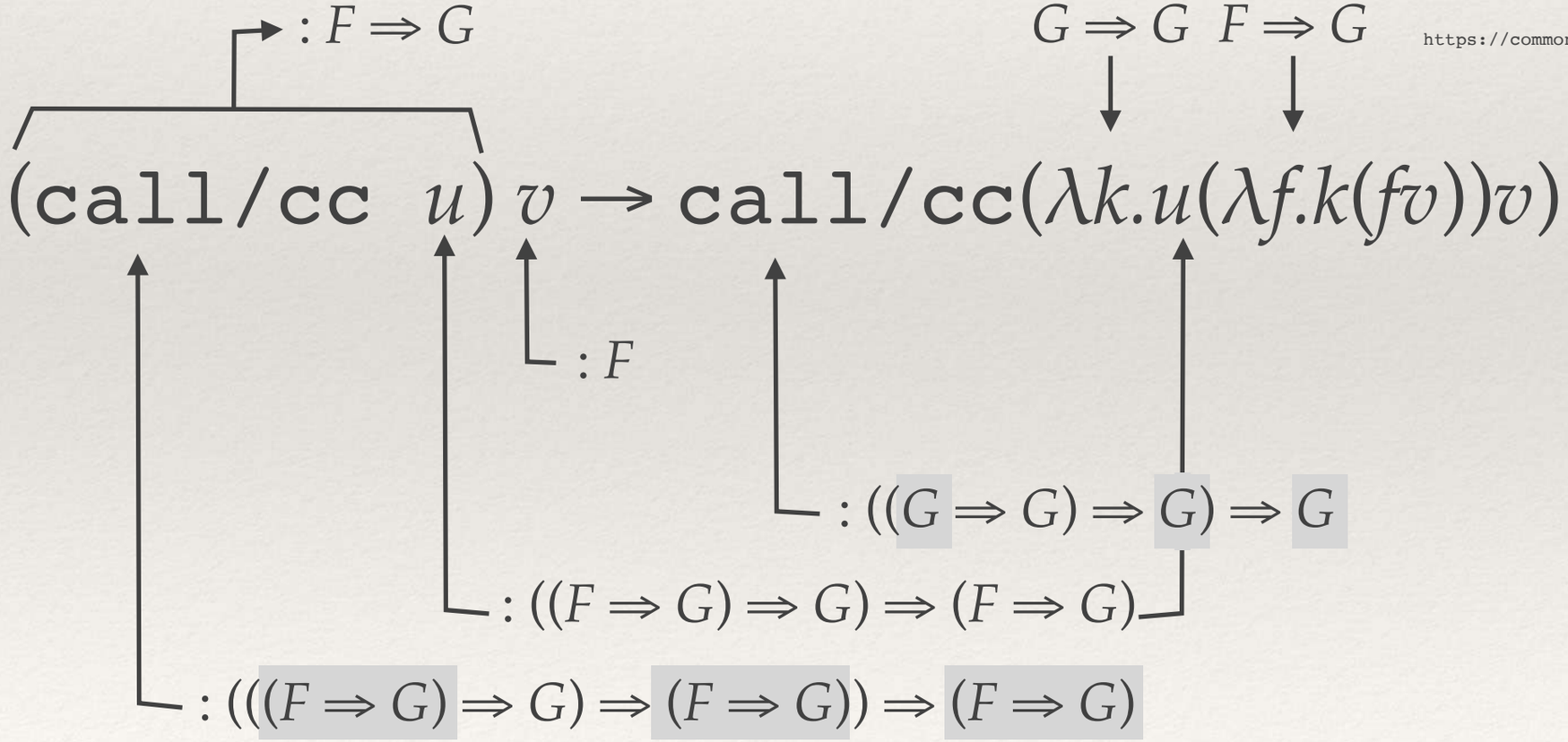
On peut typer call/cc!

- Ce que T. Griffin a découvert, en réalité:

$$\texttt{call/cc} : ((F \Rightarrow G) \Rightarrow F) \Rightarrow F$$

(loi de **Peirce**)

Domaine public,
https://commons.wikimedia.org/w/index.php?curid=128147



call/cc + $\nabla \cong \mathbf{C}$

$\texttt{call/cc} : ((F \Rightarrow G) \Rightarrow F) \Rightarrow F$

$\mathbf{C} : \neg\neg F \Rightarrow F$

- **C** est une version simplifiée de `call/cc` + ∇ due à Felleisen (encore):

- ($\leftarrow$) On peut définir $\mathbf{C}u$ (pour $u{:}\neg\neg F$) comme

  $\texttt{call/cc}(\lambda k_{\neg F} \,.\, \nabla(uk))$

- ($\rightarrow$) On peut définir `call/cc` comme

  $\lambda z_{(F \Rightarrow G) \Rightarrow F} \,.\, \mathbf{C}(\lambda k_{\neg F} \,.\, k(z(\lambda x_F \,.\, \nabla(kx))))$

  (et ∇u comme $\mathbf{C}(\lambda k.u)$)

(Je vous laisse vérifier le typage; et les réductions — ça, je ne l'ai pas fait…)

Les autres connecteurs logiques en logique classique

Le « et »

- J'ai dit qu'on pouvait définir $F \wedge G = \neg(F \Rightarrow \neg G)$

  c'est-à-dire $F \wedge G = (F \Rightarrow G \Rightarrow \bot) \Rightarrow \bot$

- Le codage s'étend aux λ-termes:

  $\langle u,v \rangle = \lambda f_{F \Rightarrow G \Rightarrow \bot} \,.\, fuv$ (comme en λ-calcul non typé)

  $\pi_1\, u = \mathbf{C}(\lambda k_{\neg F} \,.\, u(\lambda x_F \,.\, \lambda y_G \,.\, kx))$ (… ou presque)

  $\pi_2\, u = \mathbf{C}(\lambda k_{\neg F} \,.\, u(\lambda x_F \,.\, \lambda y_G \,.\, ky$

  (**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k \,.\, u(\lambda f \,.\, k(fv)))$

  (η**C**) $\mathbf{C}(\lambda k \,.\, ku) \to u$

- et aux réductions!

  $\pi_1 \langle u,v \rangle = \mathbf{C}(\lambda k_{\neg F} \,.\, (\lambda f \,.\, fuv)(\lambda x.\, \lambda y.\, kx))$

  $\to \mathbf{C}(\lambda k_{\neg F} \,.\, (\lambda x.\, \lambda y.\, kx)uv)$

  $\to^2 \mathbf{C}(\lambda k_{\neg F} \,.\, ku) \to u$ par (η**C**)

Le « ou »

- On peut définir $F \vee G \;=\; (\neg F) \Rightarrow (\neg\neg G) \;=\; (F \Rightarrow \bot) \Rightarrow (G \Rightarrow \bot) \Rightarrow \bot$

  (ça fonctionne mieux que $F \vee G = (\neg F) \Rightarrow G$, que je vous avais proposé au début)

- Le codage s'étend aux λ-termes:

$$\iota_1\, u = \lambda k_{1\, F \Rightarrow \bot} \,.\, \lambda k_{2\, G \Rightarrow \bot} \,.\, k_1 u \qquad \text{(comme en } \lambda\text{-calcul non typé)}$$

$$\iota_2\, u = \lambda k_{1\, F \Rightarrow \bot} \,.\, \lambda k_{2\, G \Rightarrow \bot} \,.\, k_2 u$$

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

$$= \mathbf{C}(\lambda k_{\neg H} \,.\, u(\lambda x_1.kv_1)(\lambda x_2.kv_2))$$

(**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k \,.\, u(\lambda f \,.\, k(fv)))$

($\eta\mathbf{C}$) $\mathbf{C}(\lambda k \,.\, ku) \to u$

- et aux réductions!

$$\texttt{case}\ \iota_1\, u\ \texttt{of}\ \iota_1\, x_1 \mapsto v_1 \mid \iota_2\, x_2 \mapsto v_2 = \mathbf{C}(\lambda k.\ (\lambda k_1.\ \lambda k_2 \,.\, k_1 u)(\lambda x_1.kv_1)(\lambda x_2.kv_2))$$

$$\to^2 \mathbf{C}(\lambda k.\ (\lambda x_1.kv_1)u)$$

$$\to \mathbf{C}(\lambda k \,.\, k(v_1[x{:=}u])) \to v_1[x{:=}u] \qquad \text{par } (\eta\mathbf{C})$$

Normalisation forte

La règle (C)

$$(\mathbf{C}) \quad (\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$$
$$(\eta\mathbf{C}) \quad \mathbf{C}(\lambda k\,.\,ku) \to u$$

- Il peut paraître improbable qu'avec une « simplification » de preuve pareille, ça termine toujours

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash u : \neg\neg(F \Rightarrow G)} \quad \dfrac{\vdots}{\Gamma \vdash \mathbf{C}u : F \Rightarrow G} \quad \dfrac{\vdots}{\Gamma \vdash v : F}\,(\Rightarrow\mathrm{E})}{\Gamma \vdash (\mathbf{C}u)v : G}\,(\neg\neg\mathrm{E}) \qquad \longrightarrow$$

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash u : \neg\neg(F \Rightarrow G)}\,(\text{affaiblissement})}{\Gamma, k{:}\neg G \vdash u : \neg\neg(F \Rightarrow G)} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash v : F}\,(\text{affaiblissement})}{\ldots \vdash v : F} \quad \dfrac{}{\ldots \vdash f : F \Rightarrow G}\,(\mathrm{Ax})}{\ldots \vdash fv : G}\,(\Rightarrow\mathrm{E}) \quad \dfrac{}{\ldots \vdash k : \neg G}\,(\mathrm{Ax})}{\Gamma, k{:}\neg G, f{:}F \Rightarrow G \vdash k(fv) : \bot}\,(\Rightarrow\mathrm{E})}{\Gamma, k{:}\neg G \vdash \lambda f\,.\,k(fv) : \neg(F \Rightarrow G)}\,(\Rightarrow\mathrm{I})}{\Gamma, k{:}\neg G \vdash u(\lambda f\,.\,k(fv)) : \bot}\,(\Rightarrow\mathrm{E})$$

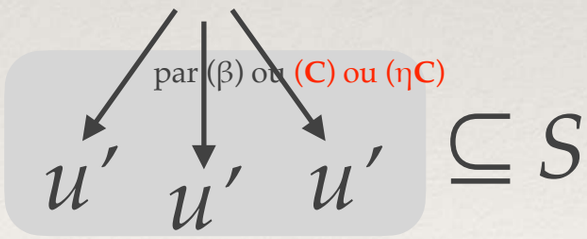
$$\dfrac{\dfrac{\Gamma, k{:}\neg G \vdash u(\lambda f\,.\,k(fv)) : \bot}{\Gamma \vdash \lambda k\,.\,u(\lambda f\,.\,k(fv)) : \neg\neg G}\,(\Rightarrow\mathrm{I})}{\Gamma \vdash \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv))) : G}\,(\neg\neg\mathrm{E})$$

Normalisation forte

- **Thm.** Tout terme typable (dans ce nouveau système de types) est fortement normalisable.
- La démonstration est comme en logique minimale… je vous montre juste les changements (en rouge).

Candidats de réductibilité

- Un ensemble de $\lambda\mathbf{C}$-termes S est un **candidat de réductibilité** si et seulement si:
- (CR1) $S \subseteq \mathrm{SN}$ *(SN = les fortement normalisables pour* $(\beta)+(\mathbf{C})+(\eta\mathbf{C})$*)*
- (CR2) si $u \in S \xrightarrow[\text{par } (\beta) \text{ ou } (\mathbf{C}) \text{ ou } (\eta\mathbf{C})]{} u'$ alors $u' \in S$ *(on suit les réductions en avant)*
- (CR3) si u neutre alors $u \in S$ *(réductions en arrière)*



où u **neutre** ssi ne commence ni par λ ni par $\mathbf{C}$.

Candidats de réductibilité

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$
$u'\ u'\ u' \subseteq S$

- **Lemme A.** Si S et S' sont des candidats de réductibilité, alors $S \Rightarrow S'$ aussi.
(Preuve: comme les dernières fois.)

$$S \Rightarrow S' = \{u \mid \forall\, v \in S,\ uv \in S'\}$$

- On pose $\mathrm{RED}_A = \mathrm{SN}$ pour toute formule atomique A,
$\mathrm{RED}_{F \Rightarrow G} = \mathrm{RED}_F \Rightarrow \mathrm{RED}_G$,
$\mathrm{RED}_\bot = \mathrm{SN}$

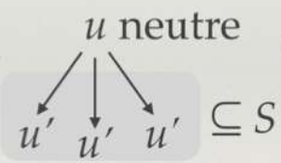
- SN vérifie (CR1)–(CR3) (facile). Donc:

- **Lemme B.** Pour tout type F, RED_F est un candidat.
(Preuve: par récurrence sur F.)

Un lemme utile

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si  alors $u \in S$

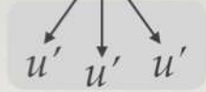
- **Lemme C.** Si (pour tout $v \in \mathrm{RED}_F$, $s[x:=v] \in \mathrm{RED}_G$) alors $\lambda x.s \in \mathrm{RED}_{F \Rightarrow G}$
- Preuve: comme la dernière fois, à coups de (CR3).
- Permettait de traiter de la règle ($\Rightarrow$I)
- Nous avons une nouvelle règle ($\neg\neg$E), pour laquelle nous démontrons le…

Un autre lemme utile

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$



$\subseteq S$

Preuve: par récurrence sur F

Lemme D. Pour tout type F, si $u \in \mathrm{RED}_{\neg\neg F}$, alors $\mathbf{C}u \in \mathrm{RED}_F$

$(\mathbf{C})\quad (\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

$(\eta\mathbf{C})\quad \mathbf{C}(\lambda k\,.\,ku) \to u$

- Si F atomique ou $\bot$, on veut démontrer $\mathbf{C}u \in \mathrm{SN}=\mathrm{RED}_F$
  Or toutes les réductions se passent dans u, sauf si on utilise $(\eta\mathbf{C})$.
  Imaginons une réduction infinie $\mathbf{C}u \to^\infty \ldots$ Deux cas:

- $\mathbf{C}u_0 \to \mathbf{C}u_1 \to \mathbf{C}u_2 \to \ldots \to \mathbf{C}u_n \to^\infty \ldots$
  où $u{=}u_0 \to u_1 \to u_2 \to \ldots \to u_n \to^\infty \ldots$: non car $u \in \mathrm{RED}_{\neg\neg F} \subseteq \mathrm{SN}$

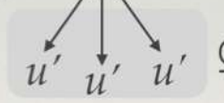
- $\mathbf{C}u_0 \to^* \mathbf{C}u_n = \mathbf{C}(\lambda k.kv) \overset{(\eta\mathbf{C})}{\to} v \to v_1 \to v_2 \to^\infty \ldots$
  où $u{=}u_0 \to^* u_n = \lambda k.kv$: non car sinon
  $u{=}u_0 \to^* u_n = \lambda k.kv \to \lambda k.kv_1 \to \lambda k.kv_2 \to^\infty \ldots$

Un autre lemme utile

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$



$\subseteq S$

(**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

($\eta\mathbf{C}$) $\mathbf{C}(\lambda k\,.\,ku) \to u$

- … reste à démontrer que

  $\mathbf{C}u \in \mathrm{RED}_{F\Rightarrow G}$, sachant que

  (H) $u \in \mathrm{RED}_{\neg\neg(F\Rightarrow G)}$

  (Hrec) pour tout terme $w \in \mathrm{RED}_{\neg\neg G}$, $\mathbf{C}w \in \mathrm{RED}_G$

- Par définition de $\mathrm{RED}_{F\Rightarrow G}$, il suffit de démontrer que pour tout $v \in \mathrm{RED}_F$, $(\mathbf{C}u)v$ est dans RED_G

- par **récurrence** sur (u, v) ordonné par $\to^* \times_{\text{lexicogr.}} \to^*$ (les deux sont dans SN, par (CR1))…. comme d'habitude

Un autre lemme utile

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$ dès que $u' \in S$ pour tout u'

- … nous devons donc démontrer que $(\mathbf{C}u)v \in \mathrm{RED}_G$, sachant que

(H) $u \in \mathrm{RED}_{\neg\neg(F \Rightarrow G)}$ (H′) $v \in \mathrm{RED}_F$

(Hrec) pour tout terme $w \in \mathrm{RED}_{\neg\neg G}$, $\mathbf{C}w \in \mathrm{RED}_G$

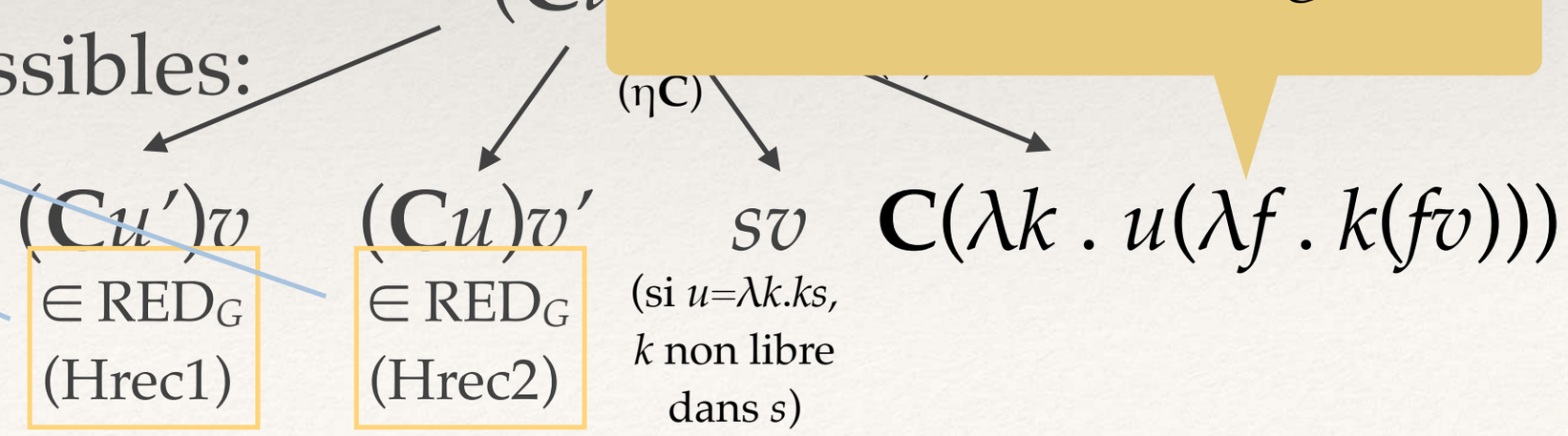
(Hrec1) pour tout réduit u' de u, $(\mathbf{C}u')v \in \mathrm{RED}_G$

(Hrec2) pour tout réduit v' de v, $(\mathbf{C}u)v' \in \mathrm{RED}_G$

(**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

(η**C**) $\mathbf{C}(\lambda k\,.\,ku) \to u$

- Il y a **quatre** cas de réduction possibles:



(C'est assez impressionnant à démontrer…)

$\mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$ est dans RED_G

… sous les hypothèses:

(H) $u \in \mathrm{RED}_{\neg\neg(F\Rightarrow G)}$ (H′) $v \in \mathrm{RED}_F$

(Hrec) pour tout terme $w \in \mathrm{RED}_{\neg\neg G}$, $\mathbf{C}w \in \mathrm{RED}_G$

(Hrec1) pour tout réduit u' de u, $(\mathbf{C}u')v \in \mathrm{RED}_G$

(Hrec2) pour tout réduit v' de v, $(\mathbf{C}u)v' \in \mathrm{RED}_G$

Vous avez sans doute vu le parallèle avec le **typage**:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots \vdash f : F \Rightarrow G}\,(\mathrm{Ax})}{\ldots \vdash k : \neg G}\,(\mathrm{Ax}) \quad \dfrac{\dfrac{\Gamma \vdash v : F}{\ldots \vdash v : F}\,(\text{affaiblissement})}{\ldots \vdash fv : G}\,(\Rightarrow\mathrm{E})}{\Gamma, k{:}\neg G, f{:}F \Rightarrow G \vdash k(fv) : \bot}\,(\Rightarrow\mathrm{E}) \quad \dfrac{\Gamma \vdash u : \neg\neg(F \Rightarrow G)}{\Gamma, k{:}\neg G \vdash u : \neg\neg(F \Rightarrow G)}\,(\text{affaiblissement})}{\Gamma, k{:}\neg G \vdash \lambda f\,.\,k(fv) : \neg(F \Rightarrow G)}\,(\Rightarrow\mathrm{I})}{\Gamma, k{:}\neg G \vdash u(\lambda f\,.\,k(fv)) : \bot}\,(\Rightarrow\mathrm{E})}{\Gamma \vdash \lambda k\,.\,u(\lambda f\,.\,k(fv)) : \neg\neg G}\,(\Rightarrow\mathrm{I})}{\Gamma \vdash \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv))) : G}\,(\neg\neg\mathrm{E})$$

Lemme C. Si $(\forall\, v \in \mathrm{RED}_F,\ s[x{:=}v] \in \mathrm{RED}_G)$ alors $\lambda x.s \in \mathrm{RED}_{F\Rightarrow G}$

Retour à la preuve…

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$

u' u' u' $\subseteq S$

- … nous devons donc démontrer que $(\mathbf{C}u)v \in \mathrm{RED}_G$, sachant que

(H) $u \in \mathrm{RED}_{\neg\neg(F \Rightarrow G)}$ (H′) $v \in \mathrm{RED}_F$

(Hrec) pour tout terme $w \in \mathrm{RED}_{\neg\neg G}$, $\mathbf{C}w \in \mathrm{RED}_G$

(Hrec1) pour tout réduit u' de u, $(\mathbf{C}u')v \in \mathrm{RED}_G$

(Hrec2) pour tout

- Il y a **quatre** cas de réduction possibles.

(**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$

(η**C**) $\mathbf{C}(\lambda k\,.\,ku) \to u$

montrons m nous venons de montrer

que ceci est c que ceci est dans RED_G

(η**C**)

$(\mathbf{C}u')v$ $\in \mathrm{RED}_G$ (Hrec1)

$(\mathbf{C}u)v'$ $\in \mathrm{RED}_G$ (Hrec2)

sv (si $u=\lambda k.ks$, k non libre dans s)

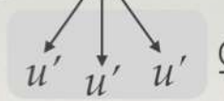
$\mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$ $\in \mathrm{RED}_G$ (!)

Le 4ème cas

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$



- … montrons que si $u = \lambda k.ks \in \mathrm{RED}_{\neg\neg H}$ (ici $H = F \Rightarrow G$)

  où k n'est pas libre dans s,

  alors $s \in \mathrm{RED}_H$

- Ceci est par récurrence sur H
- Dans le cas d'un type flèche, on utilise le Lemme C et la définition de RED, comme au 3ème cas

  (voir poly `types.pdf`, théorème 3, claim (***), p.17).

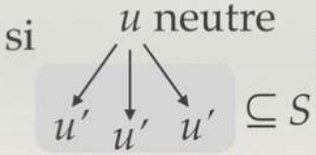
- Je vous l'épargne!

Retour à la preuve…

(CR1) $S \subseteq \mathrm{SN}$

(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$ $\subseteq S$



- … nous devons donc démontrer que
  que
) $v \in \mathrm{RED}_F$
  $\mathrm{RED}_{\neg\neg G}$, $\mathbf{C}w \in \mathrm{RED}_G$
  e u, $(\mathbf{C}u')v \in \mathrm{RED}_G$
  RED_G

(**C**) $(\mathbf{C}u)v \to \mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$
(η**C**) $\mathbf{C}(\lambda k\,.\,ku) \to u$

par (CR3) au type G,
comme $(\mathbf{C}u)v$ est neutre,
il est dans RED_G.
Ceci termine le cas $F \Rightarrow G$ de…

par le transparent précédent,
$s \in \mathrm{RED}_{F\Rightarrow G}$;
et $v \in \mathrm{RED}_F$ par (H'), donc…

ıs maintenant
est dans RED_G

(η**C**)

$(\mathbf{C}u')v$ $\in \mathrm{RED}_G$ (Hrec1)

$(\mathbf{C}u)v$ $\in \mathrm{RED}_G$ (Hrec2)

(si k ı

sv $\in \mathrm{RED}_G$ (!)

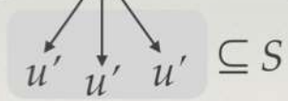
$\mathbf{C}(\lambda k\,.\,u(\lambda f\,.\,k(fv)))$ $\in \mathrm{RED}_G$ (!)

… l'autre lemme utile

(CR1) $S \subseteq \mathrm{SN}$

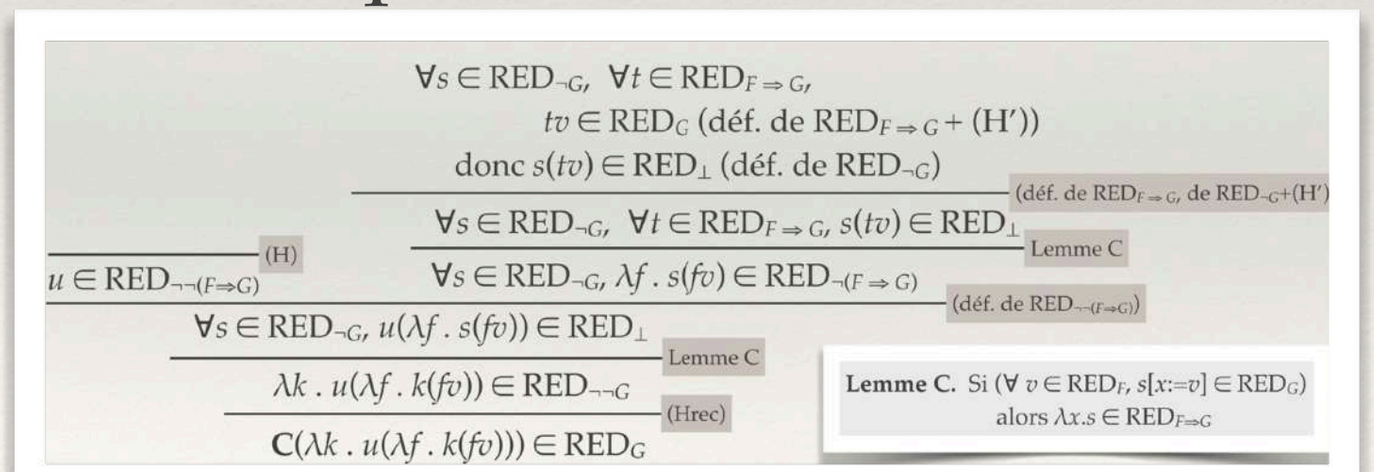
(CR2) si $u \in S \to u'$ alors $u' \in S$

(CR3) si u neutre alors $u \in S$

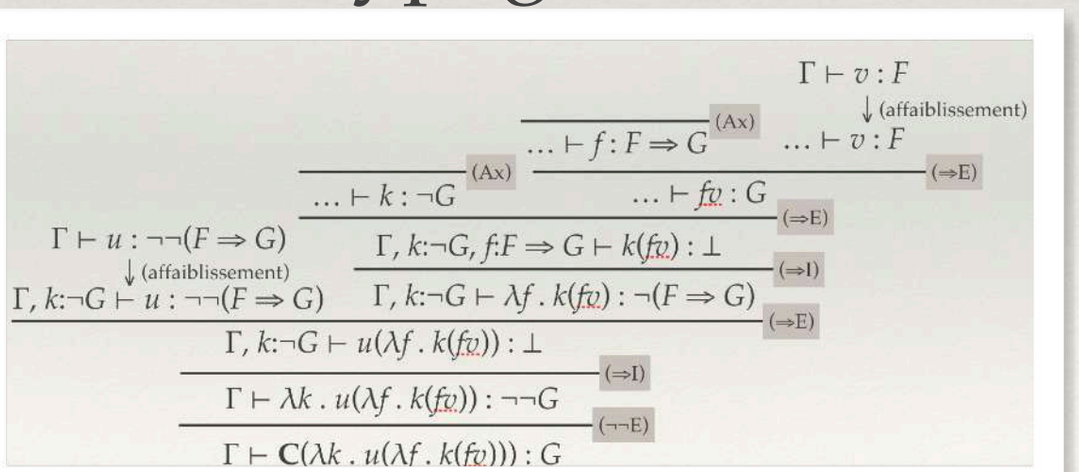


- **Lemme D.** Pour tout type F, si $u \in \mathrm{RED}_{\neg\neg F}$, alors $\mathbf{C}u \in \mathrm{RED}_F$

- Résumé: par récurrence sur F;
  pour les types flèches, on utilise (CR3):
  4 cas possibles, et un raisonnement infernal…
  qui ressemblait à une dérivation de typage



Le raisonnement infernal



La dérivation de typage analogue

Normalisation forte

- **Def:** $\theta \in \underline{\mathrm{RED}}_\Gamma$ ssi pour tout $x{:}F$ dans Γ, $\theta(x) \in \mathrm{RED}_F$
- **Thm.** si $\Gamma \vdash u : G$ dérivable alors
  pour toute $\theta \in \underline{\mathrm{RED}}_\Gamma$, $u\theta \in \mathrm{RED}_G$.
- Preuve: par récurrence sur la dérivation de $\Gamma \vdash u : G$,
  comme la dernière fois;
  on utilise la déf. de $\mathrm{RED}_{F \Rightarrow G}$ dans le cas de ($\Rightarrow$E), le Lemme C pour ($\Rightarrow$I),

  et le Lemme D pour ($\neg\neg$E).

$$\frac{\Gamma \vdash u : \neg\neg G}{\Gamma \vdash \mathbf{C}u : G}\ (\neg\neg\mathrm{E})$$

Lemme D. Pour tout type F, si $u \in \mathrm{RED}_{\neg\neg F}$, alors $\mathbf{C}u \in \mathrm{RED}_F$

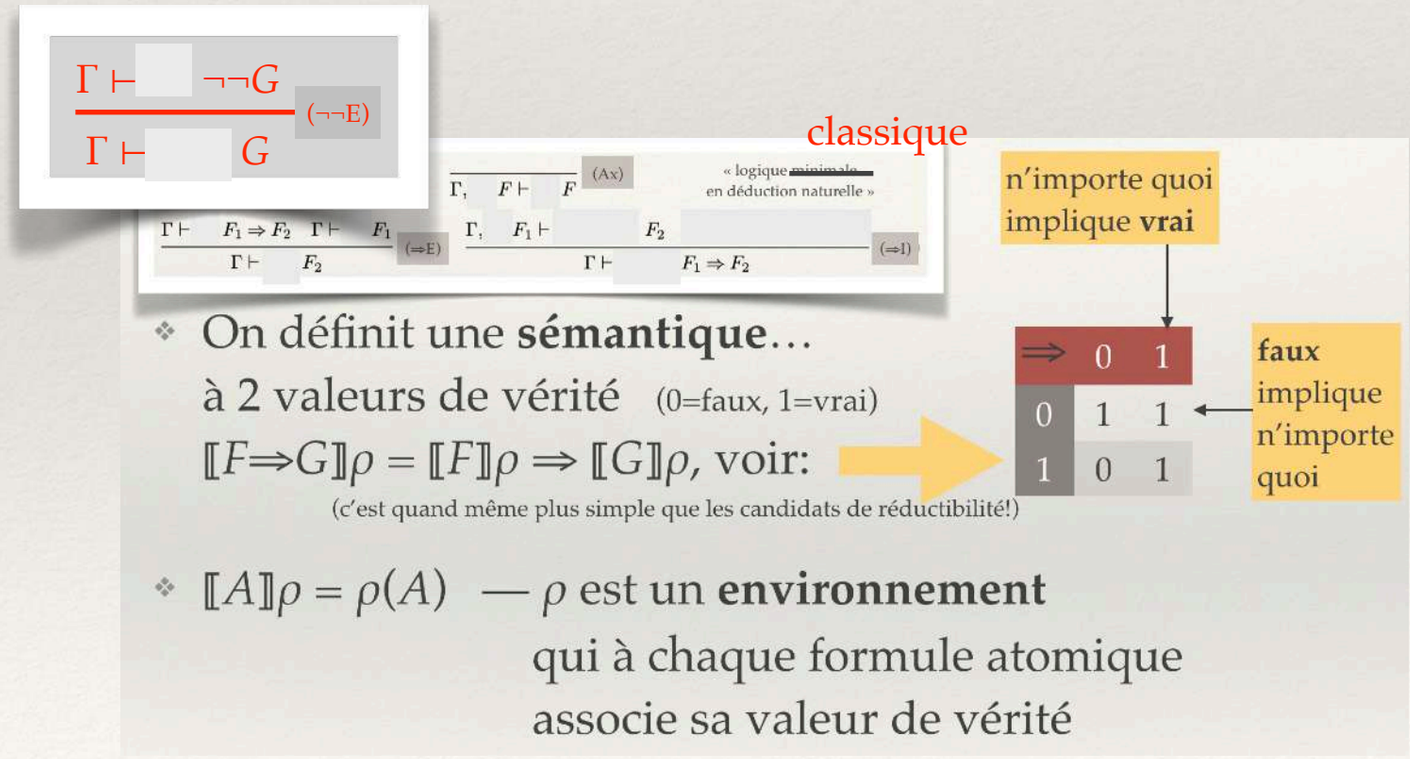
Normalisation forte

- **Def:** $\theta \in \underline{\mathrm{RED}}_\Gamma$ ssi pour tout $x{:}F$ dans Γ, $\theta(x) \in \mathrm{RED}_F$
- **Thm.** si $\Gamma \vdash u : G$ dérivable alors
  pour toute $\theta \in \underline{\mathrm{RED}}_\Gamma$, $u\theta \in \mathrm{RED}_G$.
- **Corollaire.** Tout λ**C**-terme simplement typable
  est fortement normalisable.

Pourquoi « classique »?

- On peut démontrer que le système de déduction naturelle sous-jacent à $\lambda\mathbf{C}$
  est **correct** et **complet** pour l'interprétation dans les booléens de la dernière fois

- Ca prend du temps…
  (omis)



Voilà

- … pour les logiques propositionnelles
- Les prochaines fois, nous passerons à:
  — la logique et l'arithmétique du 1er ordre
  — l'arithmétique de 2nd ordre et le système F
  — enfin à certains considérations théoriques sur l'implémentation de machines de réduction pour le λ-calcul