Jean Goubault-Larrecq

λ-calcul

11. Machines, interprètes

Aujourd'hui

- Comment implémenter le λ-calcul?
- Beaucoup de solutions possibles, on n'en verra que quelques-unes
- … et l'on finira par se concentrer sur quelques problèmes théoriques.

Interprètes naïfs en Caml

Représentation des λ-termes

```
type lambda_terme = VAR of string
                  | APPL of lambda_terme * lambda_terme
                  | ABS of string * lambda_terme
```

Par exemple, $\lambda x \,.\, y(\lambda z \,.\, yxzx)$ serait:

```
ABS ("x", APPL (VAR "y",
                ABS ("z",
                     APPL (APPL (APPL (VAR "y", VAR "x"),
                                 VAR "z"),
                           VAR "x"))))
```

Un premier int

```
type lambda_terme = VAR of string
                  | APPL of lambda_ter    lambda_terme
                  | ABS of string * l mbda_terme

let rec norm (t : lambda_terme) = (* calcule la forme normale de t *
    match t
        | VA
        | AB                 ABS (x, norm u)
        | APPL (u, v) ->
            let u' = norm u in
            match u' with
                | ABS (y, u1) -> norm (subst u1 y v)
                | _ -> APPL (u', norm v)
```

Réduction complète.
Pour une réduction faible,
écrire `t` à la place
de `ABS (x, norm u)`

Ce code est-il correct?

Réduction
gauche

À définir…
(attention au **α-renommage**!)

Efficacité?
`u1` est normal,
mais on va quand même
normaliser `subst u1 y v`

Efficacité?
(pas terrible…)

L'astuce de Krivine

Jean-Louis Krivine

Gandvine.jpg

- Pour éviter la r… on garde une li… des argument… en attente

```
let rec apply_head (t : lambda_terme) …                    t) =
    match args with
          | nil -> t
          | v::rest -> apply_head (APPL (t, u …

let rec no…                     ne) (args : lam…           ) =
    match …
          | VAR x -> apply_head t args
          | ABS (x, u) -> (match args with
                              | nil -> ABS (x, norm u nil)
                              | v::rest -> norm (subst u x v) rest)
          | APPL (u, v) -> norm u (v::args)
```

Si `args`=[v_1; …; v_n], calcule la forme normale de `t` v_1 … v_n

Exploration du terme: recherche du rédex de tête

Réduction complète. Pour une réduction faible, écrire `t` à la place de `ABS (x, norm u nil)`

La β-réduction est ici

Une machine de Krivine

Reste quand même à définir la substitution… ou à la remplacer (efficacité)

- On abstrait tout ça sous forme de sémantique opérationnelle

(ici, pour la réduction de tête faible)

https://www.irif.fr/~padovani/Images/Gandvine.jpg

- (explore) $uv, args \rightsquigarrow u, v{::}args$
  (β) $\lambda x\,.\,u, v{::}rest \rightsquigarrow u[x{:=}v], rest$
- **Thm (correction).** Si $u, [v_1; \ldots; v_n] \rightsquigarrow^* u', [v'_1; \ldots; v'_m]$
  alors $u\ v_1 \ldots v_n \rightarrow_{\mathrm{wh}}{}^* u'\ v'_1 \ldots v'_m$.
- **Thm (progrès).** Les configurations stoppées
  sont celles de la forme $x, [v_1; \ldots; v_n]$ ou $\lambda x\,.\,u, []$
  (NB: $x\ v_1 \ldots v_n$ et $\lambda x\,.\,u$ sont les formes normales de tête faibles)

Logique combinatoire

Logique combinatoire

- On peut éviter les problèmes d'α-renommage en passant à la logique combinatoire, qui est un langage **sans variable liée**

Règles de réduction:

$\texttt{S}MNP \to MP(NP)$

$\texttt{K}MN \to M$

$\texttt{I}M \to M$

- Termes combinatoires:

$M, N, P, \ldots ::=$	$\texttt{S} \mid \texttt{K} \mid \texttt{I}$	**combinateurs** (constantes)
	$\mid x$	variables
	$\mid MN$	applications

… et pas de lambda-abstractions

- Mais tout λ-terme u aura une traduction en un terme combinatoire u^*!

Traductions en termes combinatoires

$$\begin{aligned} M, N, P, \ldots ::= &\ \mathtt{S} \mid \mathtt{K} \mid \mathtt{I} && \textbf{combinateurs} \text{ (constantes)} \\ &\mid x && \text{variables} \\ &\mid MN && \text{applications} \end{aligned}$$

- On définit:

$$\begin{aligned} x^* &\stackrel{\text{def}}{=} x \\ (uv)^* &\stackrel{\text{def}}{=} u^*\, v^* \\ (\lambda x\, .\, u)^* &\stackrel{\text{def}}{=} [x]\, u^* \end{aligned}$$

- L'opération $[x]\, M$ (« bracket abstraction ») reste à définir
- Elle **ne fait pas partie** du langage des termes combinatoires (c'est une méta-opération, comme la substitution en λ-calcul)

Traductions en termes combinatoires

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$		**combinateurs** (constantes)
$\mid x$		variables
$\mid MN$		applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\,.\,u) \stackrel{\text{def}}{=} [x]\, u^*$$

Règles de réduction:

$\mathtt{S}MNP \to MP(NP)$

$\mathtt{K}MN \to M$

$\mathtt{I}M \to M$

- On va définir $[x]\, M$
  de sorte que, pour tout N,
  $([x]\, M)\, N \to^* M[x{:=}N]$

- C'est comme ça
  qu'on verra apparaître $\mathtt{S}$, $\mathtt{K}$, $\mathtt{I}$ et leurs règles de réduction

- On va donc démontrer le:

  Lemme. $([x]\, M)\, N \to^* M[x{:=}N]$

  par récurrence sur M, en ajoutant les règles dont on a besoin au fur et à mesure.

Traductions en termes combinatoires

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$	**combinateurs** (constantes)
$\mid x$	variables
$\mid MN$	applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\, .\, u) \stackrel{\text{def}}{=} [x]\, u^*$$

Règles de réduction:

$\mathtt{S}MNP \to MP(NP)$

$\mathtt{K}MN \to M$

$\mathtt{I}M \to M$

- **Lemme.** $([x]\, M)\, N \to^* M[x{:=}N]$
- Cas 1/3: $M{=}x$
- On cherche un terme $X = [x]\, M$ tel que
  $$X\, N \to^* M[x{:=}N] = N$$
- On pose $X \stackrel{\text{def}}{=} \mathtt{I}$,
  et on ajoute la règle $\mathtt{I}N \to N$
- Résumé du cas 1: si $M{=}x$, on pose $[x]\, M \stackrel{\text{def}}{=} \mathtt{I}$.

Traductions en termes combinatoires

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$ **combinateurs** (constantes)
$\mid x$ variables
$\mid MN$ applications

$x^* \stackrel{\text{def}}{=} x$
$(uv)^* \stackrel{\text{def}}{=} u^* v^*$
$(\lambda x\, .\, u) \stackrel{\text{def}}{=} [x]\, u^*$

Règles de réduction:
$\mathtt{S}MNP \to MP(NP)$
$\mathtt{K}MN \to M$
$\mathtt{I}M \to M$

$[x]\, x \stackrel{\text{def}}{=} \mathtt{I}$

- **Lemme.** $([x]\, M)\, N \to^* M[x{:=}N]$
- Cas 2/3: $x \notin \mathrm{fv}(M)$
- On cherche un terme $X = [x]\, M$ tel que
  $X\, N \to^* M[x{:=}N] = M$
- On pose $X \stackrel{\text{def}}{=} \mathtt{K}M$,
  et on ajoute la règle $\mathtt{K}MN \to M$

Traductions en termes combinatoires

- **Lemme.** $([x]\, M)\, N \to^* M[x{:=}N]$
- <u>Cas 3/3</u>: $x \in \mathrm{fv}(M)$ et $M \neq x$
- Alors M est une application $M_1 M_2$
- On cherche un terme $X = [x]\, M$ tel que

$$X\, N \to^* M[x{:=}N] = (M_1[x{:=}N])(M_2[x{:=}N])$$

- X sera construit à partir de $[x]M_1$ et de $[x]M_2$, obtenus par hypothèse de récurrence, donc posons

$$X \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2)$$

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$	**combinateurs** (constantes)
$\mid x$	variables
$\mid MN$	applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\, .\, u) \stackrel{\text{def}}{=} [x]\, u^*$$

<u>Règles de réduction:</u>

$\mathtt{S}MNP \to MP(NP)$

$\mathtt{K}MN \to M$

$\mathtt{I}M \to M$

$[x]\, x \stackrel{\text{def}}{=} \mathtt{I}$

$[x]\, M \stackrel{\text{def}}{=} \mathtt{K}M$ si $x \notin \mathrm{fv}(M)$

Traductions en termes combinatoires

- **Lemme.** $([x]\, M)\, N \to^* M[x{:=}N]$
- <u>Cas 3/3</u>: $x \in \mathrm{fv}(M)$ et $M \neq x$
- Alors M est une application $M_1 M_2$
- On cherche un terme $X = [x]\, M$ tel que
  $$X\, N \to^* M[x{:=}N] = (M_1[x{:=}N])(M_2[x{:=}N])$$
- … $X \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2)$ (résumé)
- Alors $XN = \mathtt{S}([x]M_1)([x]M_2)N \to \ldots$?
  $$\ldots$$
  $$\to^* (M_1[x{:=}N])(M_2[x{:=}N])$$
  $$= M[x{:=}N]$$

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$	**combinateurs** (constantes)
$\mid x$	variables
$\mid MN$	applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\, .\, u) \stackrel{\text{def}}{=} [x]\, u^*$$

<u>Règles de réduction:</u>

$\mathtt{S}MNP \to MP(NP)$

$\mathtt{K}MN \to M$

$\mathtt{I}M \to M$

$$[x]\, x \stackrel{\text{def}}{=} \mathtt{I}$$
$$[x]\, M \stackrel{\text{def}}{=} \mathtt{K}M \quad \text{si } x \notin \mathrm{fv}(M)$$
$$[x]\, (M_1 M_2) \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2) \quad \text{si } x \in \mathrm{fv}(M_1 M_2)$$

Traductions en termes combinatoires

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$	**combinateurs** (constantes)
$\mid x$	variables
$\mid MN$	applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\,.\,u) \stackrel{\text{def}}{=} [x]\, u^*$$

<u>Règles de réduction:</u>

$\mathtt{S}MNP \to MP(NP)$

$\mathtt{K}MN \to M$

$\mathtt{I}M \to M$

$[x]\, x \stackrel{\text{def}}{=} \mathtt{I}$

$[x]\, M \stackrel{\text{def}}{=} \mathtt{K}M$ si $x \notin \mathrm{fv}(M)$

$[x]\, (M_1M_2) \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2)$ si $x \in \mathrm{fv}(M_1M_2)$

- **Lemme.** $([x]\, M)\, N \to^* M[x:=N]$
- <u>Cas 3/3</u>: $x \in \mathrm{fv}(M)$ et $M \neq x$
- Alors M est une application M_1M_2
- On cherche un terme X … $X\,N \to^* M[x:=N]$ …])
- … $X \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2)$ (resu…)
- Alors $XN = \mathtt{S}([x]M_1)([x]M_2)N \to \ldots$?

  … $(([x]M_1)N)\,(([x]M_2)N)$

  $\to^* (M_1[x:=N])\,(M_2[x:=N])$ par hypothèse de récurrence

  $= M[x:=N]$

Il n'y a qu'à poser la nouvelle règle: $\mathtt{S}MNP \to MP(NP)$!

Traductions en termes combinatoires

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$ **combinateurs** (constantes)
$\mid x$ variables
$\mid MN$ applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\,.\,u) \stackrel{\text{def}}{=} [x]\, u^*$$

<u>Règles de réduction:</u>

$\mathtt{S}MNP \to MP(NP)$
$\mathtt{K}MN \to M$
$\mathtt{I}M \to M$

$$[x]\, x \stackrel{\text{def}}{=} \mathtt{I}$$
$$[x]\, M \stackrel{\text{def}}{=} \mathtt{K}M \quad \text{si } x \notin \mathrm{fv}(M)$$
$$[x]\, (M_1M_2) \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2) \quad \text{si } x \in \mathrm{fv}(M_1M_2)$$

- **Lemme.** $([x]\, M)\, N \to^* M[x:=N]$
- <u>Cas 3/3</u>: $x \in \mathrm{fv}(M)$ et $M \neq x$
- Alors M est une application M_1M_2
- On cherche un terme $X = [x]\, M$ tel que
  $$X\, N \to^* M[x:=N] = (M_1[x:=N])(M_2[x:=N])$$
- … $X \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2)$ (résumé)
- Alors $XN = \mathtt{S}([x]M_1)([x]M_2)N$
  $\to (([x]M_1)N)\,(([x]M_2)N)$
  $\to^* (M_1[x:=N])\,(M_2[x:=N])$ par hypothèse de récurrence
  $= M[x:=N]$. $\square$

Traductions en termes combinatoires

$M, N, P, \ldots ::= \mathtt{S} \mid \mathtt{K} \mid \mathtt{I}$ — **combinateurs** (constantes)
$\mid x$ — variables
$\mid MN$ — applications

$$x^* \stackrel{\text{def}}{=} x$$
$$(uv)^* \stackrel{\text{def}}{=} u^*\, v^*$$
$$(\lambda x\, .\, u) \stackrel{\text{def}}{=} [x]\, u^*$$

Règles de réduction:

$$\mathtt{S}MNP \to MP(NP)$$
$$\mathtt{K}MN \to M$$
$$\mathtt{I}M \to M$$

$$[x]\, x \stackrel{\text{def}}{=} \mathtt{I}$$
$$[x]\, M \stackrel{\text{def}}{=} \mathtt{K}M \quad \text{si } x \notin \text{fv}(M)$$
$$[x]\, (M_1M_2) \stackrel{\text{def}}{=} \mathtt{S}([x]M_1)([x]M_2) \quad \text{si } x \in \text{fv}(M_1M_2)$$

- **Lemme.** $([x]\, M)\, N \to^* M[x:=N]$
- On peut ensuite démontrer:
  Lemme. $u^*[x:=v^*] = (u[x:=v])^*$
- Donc $((\lambda x\, .\, u)\, v)^* = ([x]\, u^*)\, v^*$
  $\to^* u^*[x:=v^*] = (u[x:=v])^*$
  On en déduit que la logique combinatoire implémente correctement la β-réduction:

  ou presque!

- **Lemme.** Si $u \to_{\text{tf}}^* v$, alors $u^* \to^* v^*$

  Arg: réduction de tête **faible** seulement (ou un peu plus)… mais ne préserve pas la réduction **sous les λ**

- Détails: voir TD.

Logique combinatoire

- Il est maintenant facile d'implémenter une machine pour réduire les termes combinatoires:

Règles de réduction:

$$\mathrm{S}MNP \to MP(NP)$$
$$\mathrm{K}MN \to M$$
$$\mathrm{I}M \to M$$

```
type ski_terme = S | K | I
               | VAR of string
               | APPL of ski_terme * ski_terme;;

let rec ski_norm m =
  match m with
    S | K | I -> m
  | VAR x -> m
  | APPL (m0, m1) ->
    match ski_norm m0 with
      I -> ski_norm m1
    | APPL (K, m') -> m'
    | APPL (APPL (S, m3), m2) -> ski_norm (APPL (APPL (m3, m1), APPL (m2, m1)))
    | m'0 -> APPL (m'0, ski_norm m1);;
```

Ne réduit pas sous les λ

Une machine de Krivine pour SKI

https://www.irif.fr/~padovani/Images/Gandvine.jpg

- (explore) $uv, args \rightsquigarrow u, v::args$

  (`I`) `I`, $v::rest \rightsquigarrow v, rest$

  (`K`) `K`, $v_1::v_2::rest \rightsquigarrow v_1, rest$

  (`S`) `S`, $v_1::v_2::v_3::rest \rightsquigarrow v_1, v_3::(v_2 v_3)::rest$

<u>Règles de réduction:</u>

$\texttt{S}MNP \to MP(NP)$

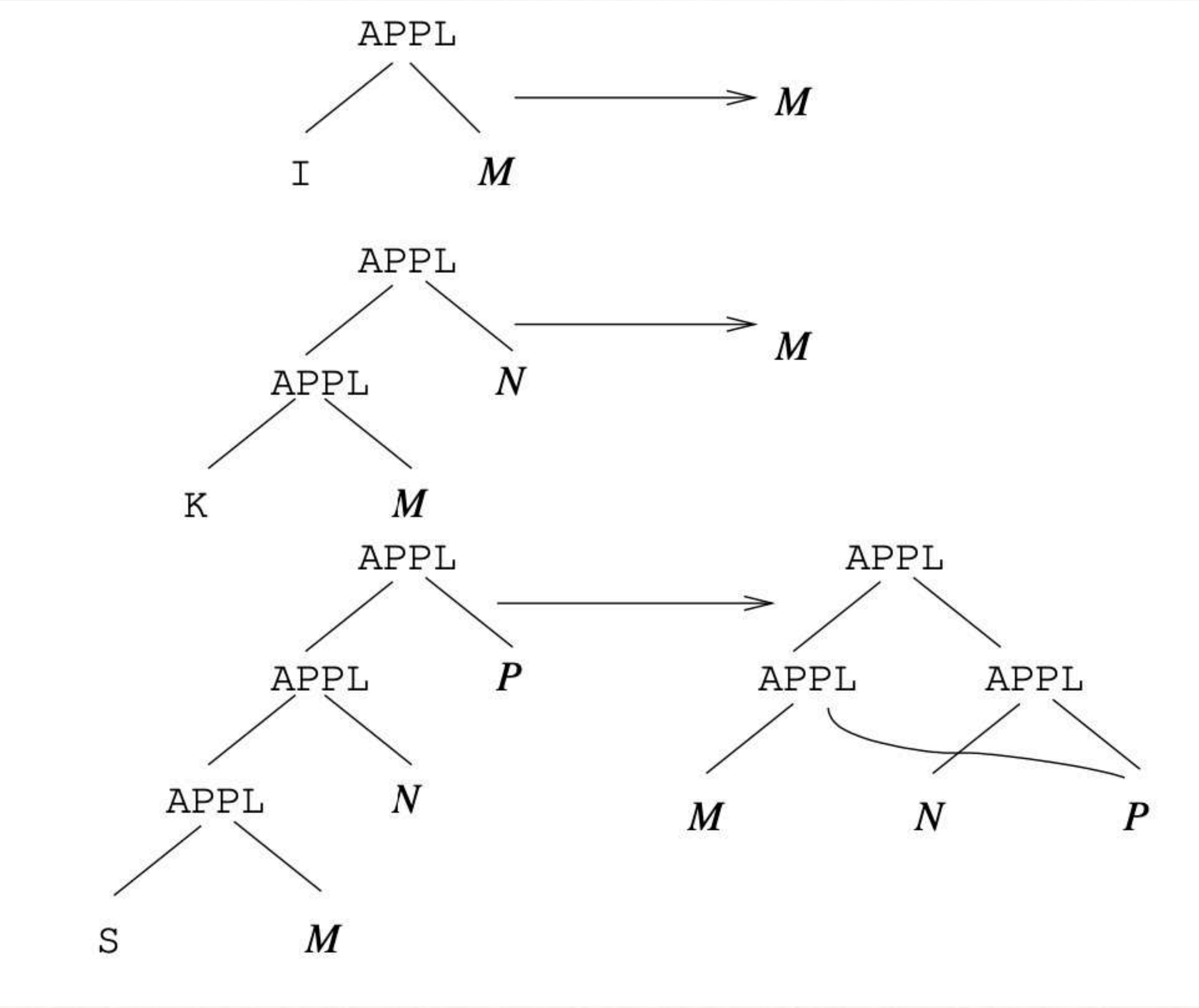
$\texttt{K}MN \to M$

$\texttt{I}M \to M$

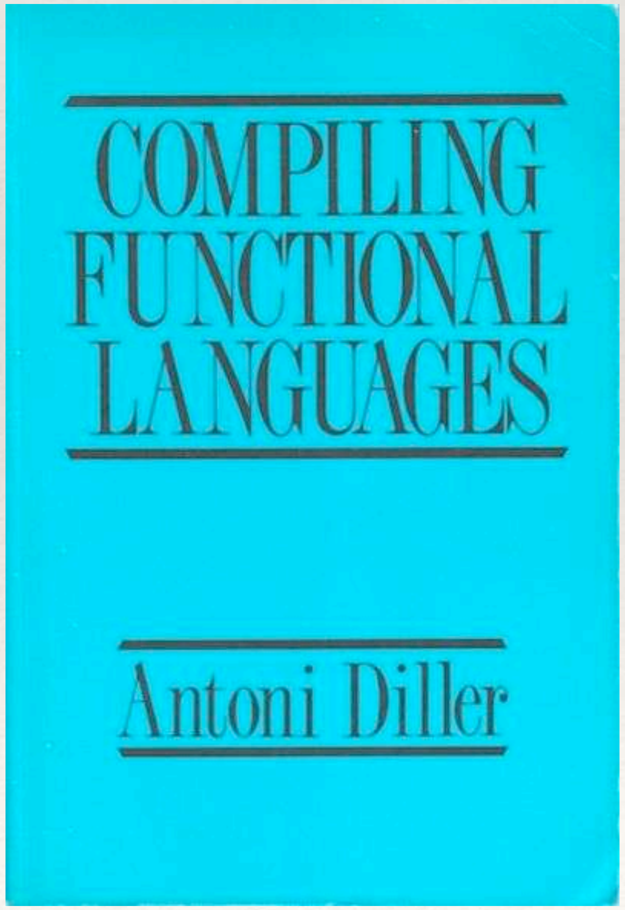
- Théorèmes de correction, de progrès: voir poly `machines.pdf`

Machines à réduction de graphes

- Permettent d'éviter la réévaluation d'un même argument copié plusieurs fois



Pour toutes les variantes de l'idée, lire:



Machines à environnement

Le problème de la substitution

- (explore) $uv, args \leadsto u, v{::}args$
  (β) $\lambda x\ .\ u, v{::}rest \leadsto u[x{:=}v], rest$
- Comment implémenter $u[x{:=}v]$?
- … sans faire d'erreur d'**α-renommage**?
- … **paresseusement**?

Machines à environnement

- On maintient un **environnement** ρ qui à chaque variable x associe un terme par lequel x aurait dû être remplacée

plutôt $v\rho$, non?

problème ici aussi, non?

- (explore) $uv, \rho, args \rightsquigarrow u, \rho, v{::}args$
  (β) $\lambda x\,.\,u, \rho, v{::}rest \rightsquigarrow u, \rho[x \mapsto v], rest$
  (var) $x, \rho, args \rightsquigarrow \rho(x), \rho, args$ si $x \in \mathrm{dom}\ \rho$
- … modulo ***α*-renommage** (ignoré ici!)
- … **correct**?

Les calculs à substitutions explicites

- … à commencer par le **$\lambda\sigma$-calcul**
- gère les **environnements** à l'intérieur du calcul
- résout les difficultés d'**α-renommage** en suivant une ancienne idée de N. G. de Bruijn

Explicit Substitutions

Martín Abadi, Luca Cardelli, Pierre-Louis Curien, Jean-Jacques Lévy

February 6th, 1990

La notation de de Bruijn

- Idée: remplacer les **variables liées**
- par des **pointeurs** vers les abstractions qui les lient
- puis noter ces pointeurs par des **entiers**

Nicolaas Govert de Bruijn

https://www.tuencyclopedie.nl/images/thumb/9/9c/Lemma_28_Foto_1.jpg/200px-Lemma_28_Foto_1.jpg

MATHEMATICS

LAMBDA CALCULUS NOTATION WITH NAMELESS DUMMIES, A TOOL FOR AUTOMATIC FORMULA MANIPULATION, WITH APPLICATION TO THE CHURCH-ROSSER THEOREM

BY

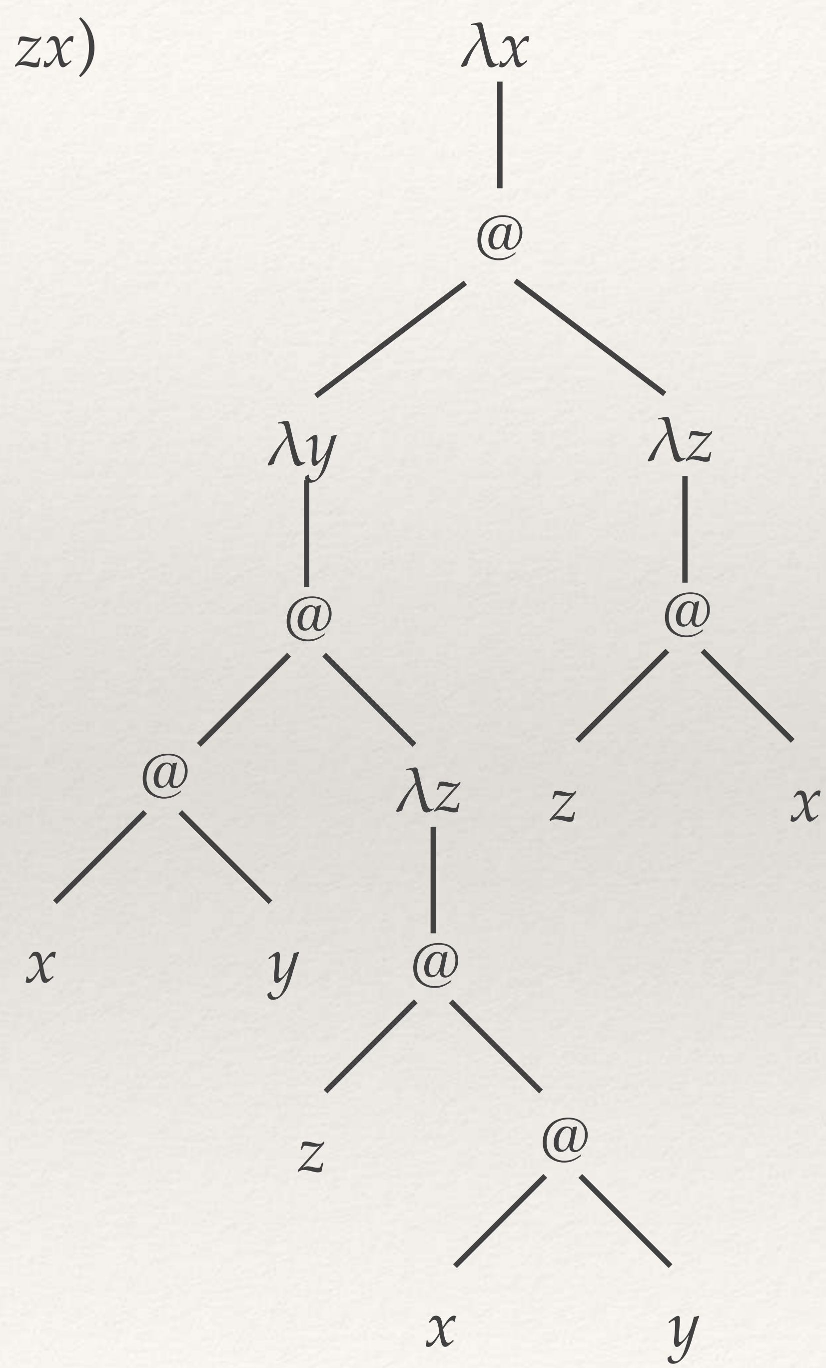
N. G. DE BRUIJN

(Communicated at the meeting of June 24, 1972)

ABSTRACT

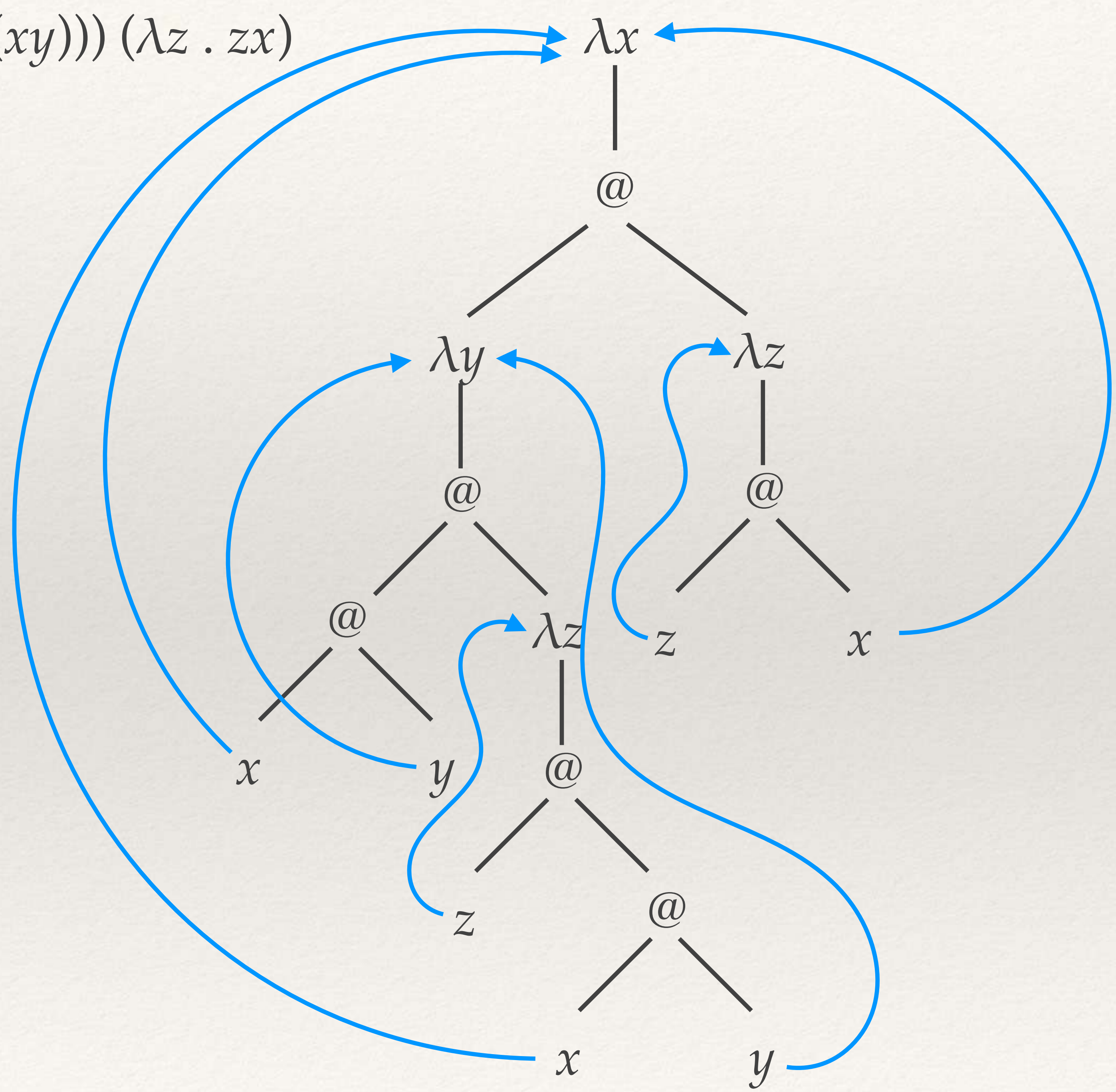
Un exemple

$\lambda x \,.\, (\lambda y \,.\, xy(\lambda z \,.\, z(xy)))\,(\lambda z \,.\, zx)$



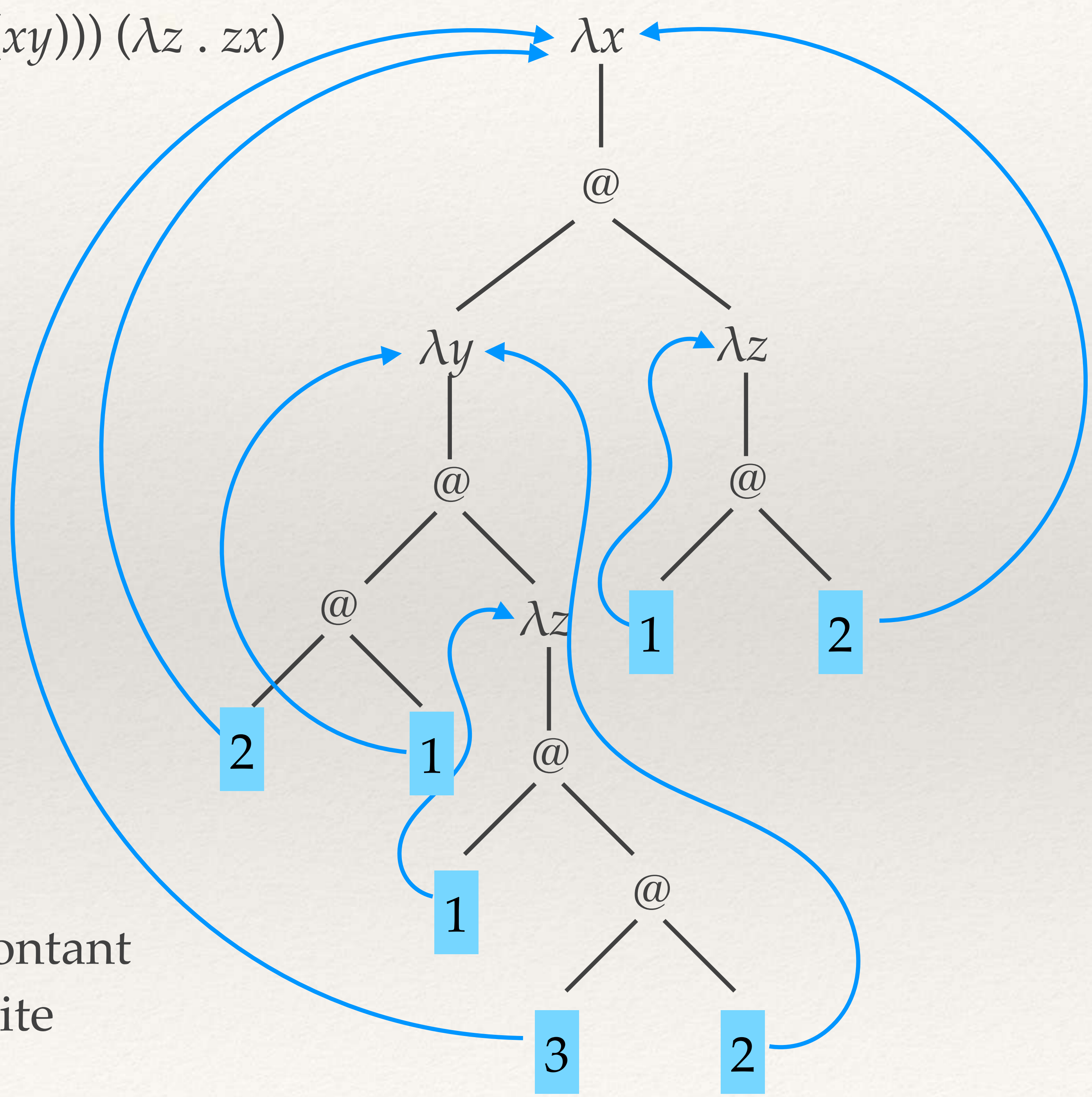
Pointeurs…

$\lambda x\,.\,(\lambda y\,.\,xy(\lambda z\,.\,z(xy)))\,(\lambda z\,.\,zx)$



Pointeurs… codés par des entiers

$\lambda x\,.\,(\lambda y\,.\,xy(\lambda z\,.\,z(xy)))\,(\lambda z\,.\,zx)$



On code
chaque pointeur
par un entier
comptant
le **nombre de λ**
à traverser en remontant
pour retrouver le site
de liaison

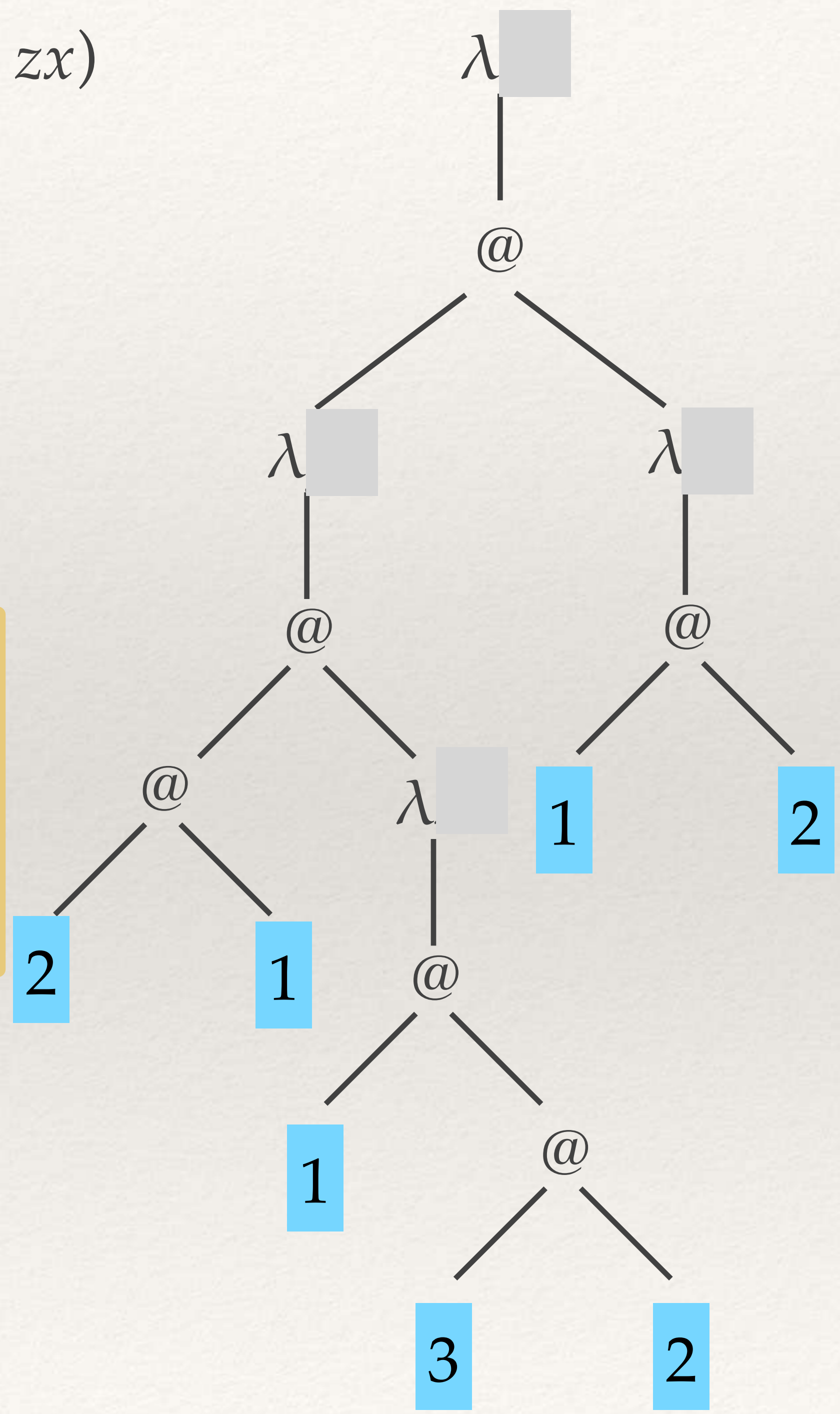
Les variables sont inutiles!

Note 2: le même numéro peut représenter des variables différentes

$\lambda x\,.\,(\lambda y\,.\,xy(\lambda z\,.\,z(xy)))\,(\lambda z\,.\,zx)$

Notation de de Bruijn:
$\lambda\,.\,(\lambda\,.\,21(\lambda\,.\,1(32)))\,(\lambda\,.\,12)$

Note 1: une même variable peut apparaître avec plusieurs numéros

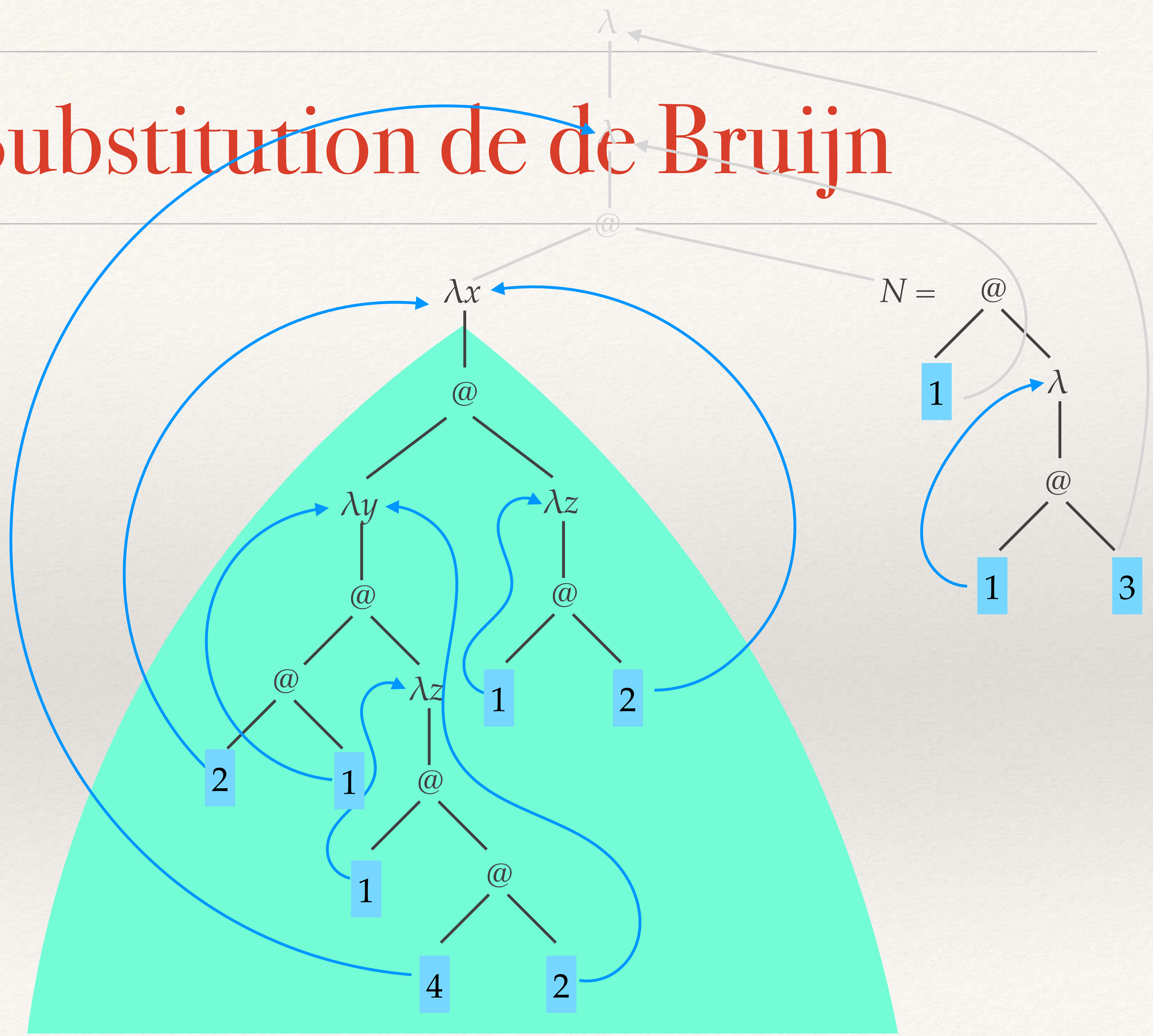


Théorème. Deux λ-termes sont α-équivalents
ssi leurs notations de de Bruijn sont **identiques**.

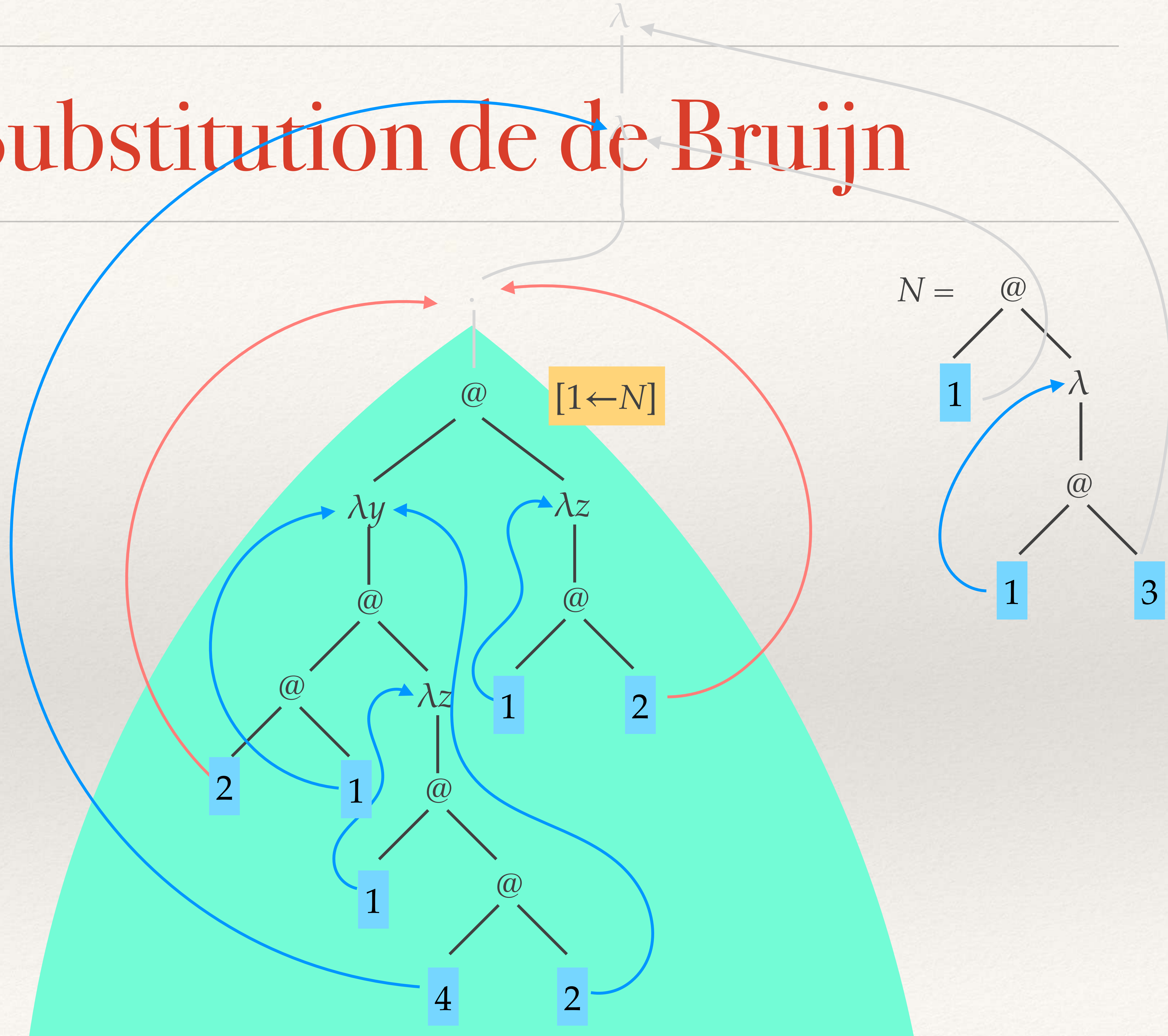
Termes de de Bruijn

- Syntaxe: $M, N, \ldots ::= x$ (variables **libres**)
  $\mid 1 \mid 2 \mid \ldots$ (indices de de Bruijn)
  $\mid MN$
  $\mid \lambda M$ (plutôt que $\lambda . M$)
- La β-réduction devient:
  $$(\lambda M)N \to M[1 \leftarrow N]$$
- … on doit définir le « remplacement de 1 par N »

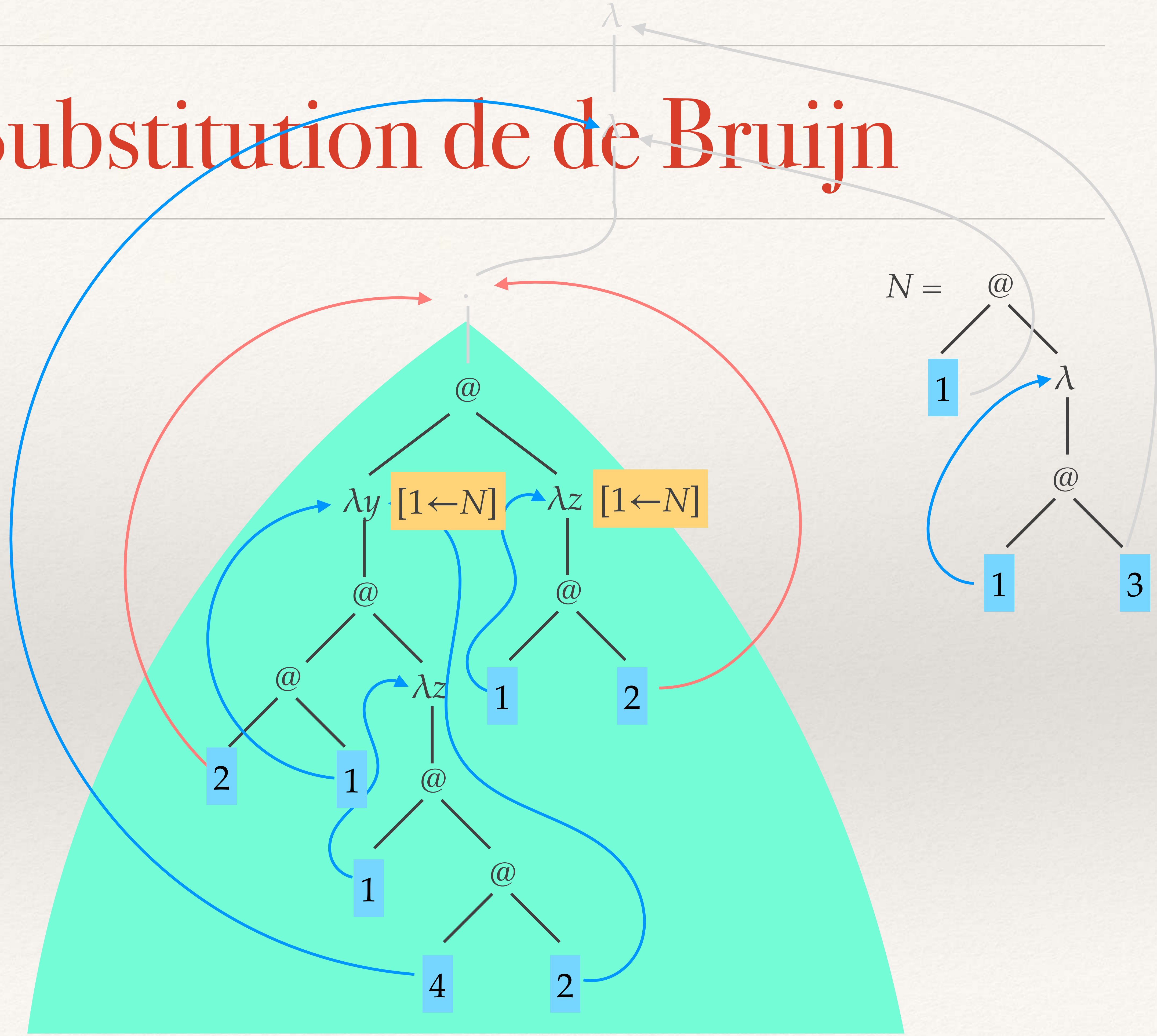
Substitution de de Bruijn



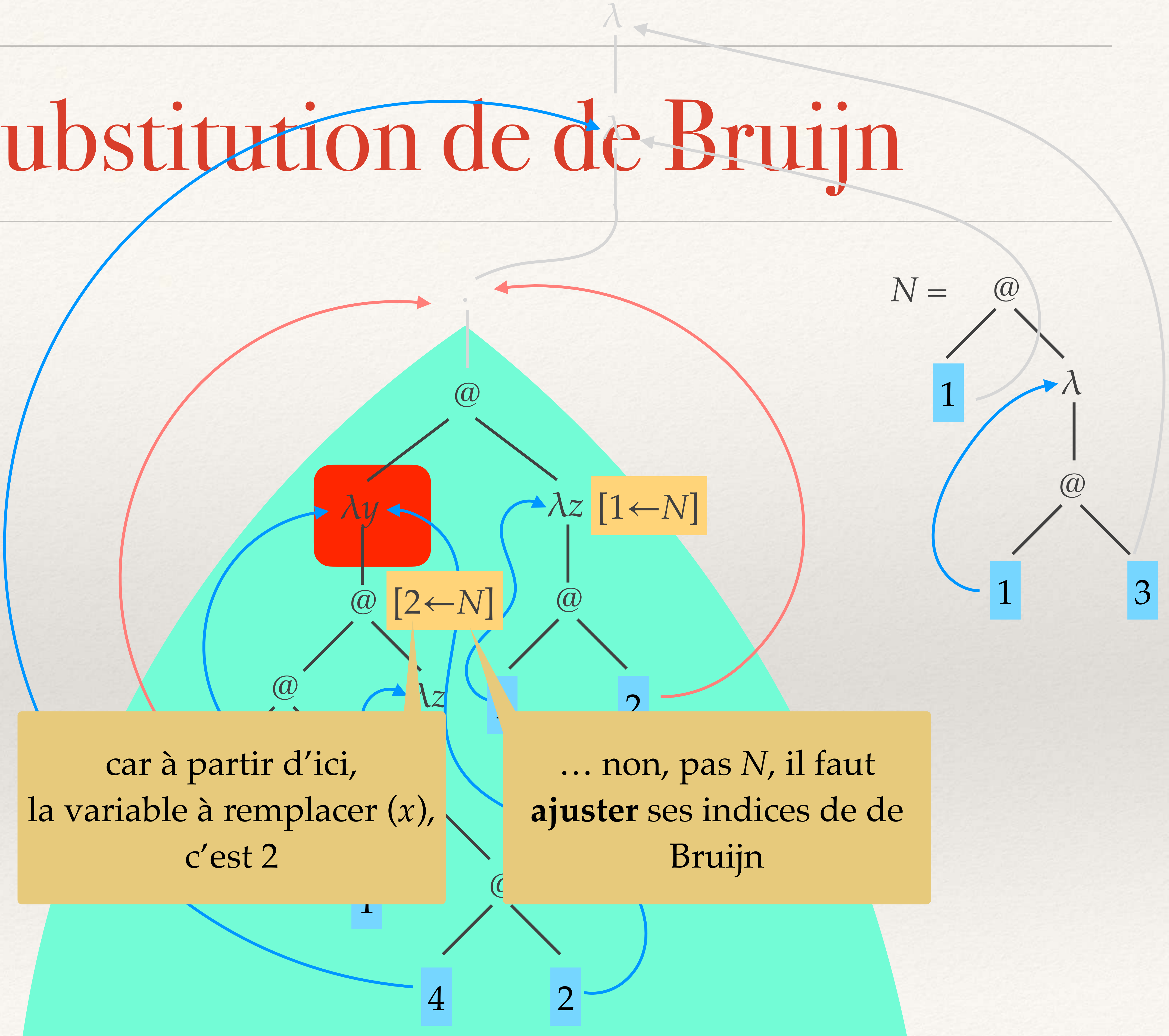
Substitution de de Bruijn



Substitution de de Bruijn



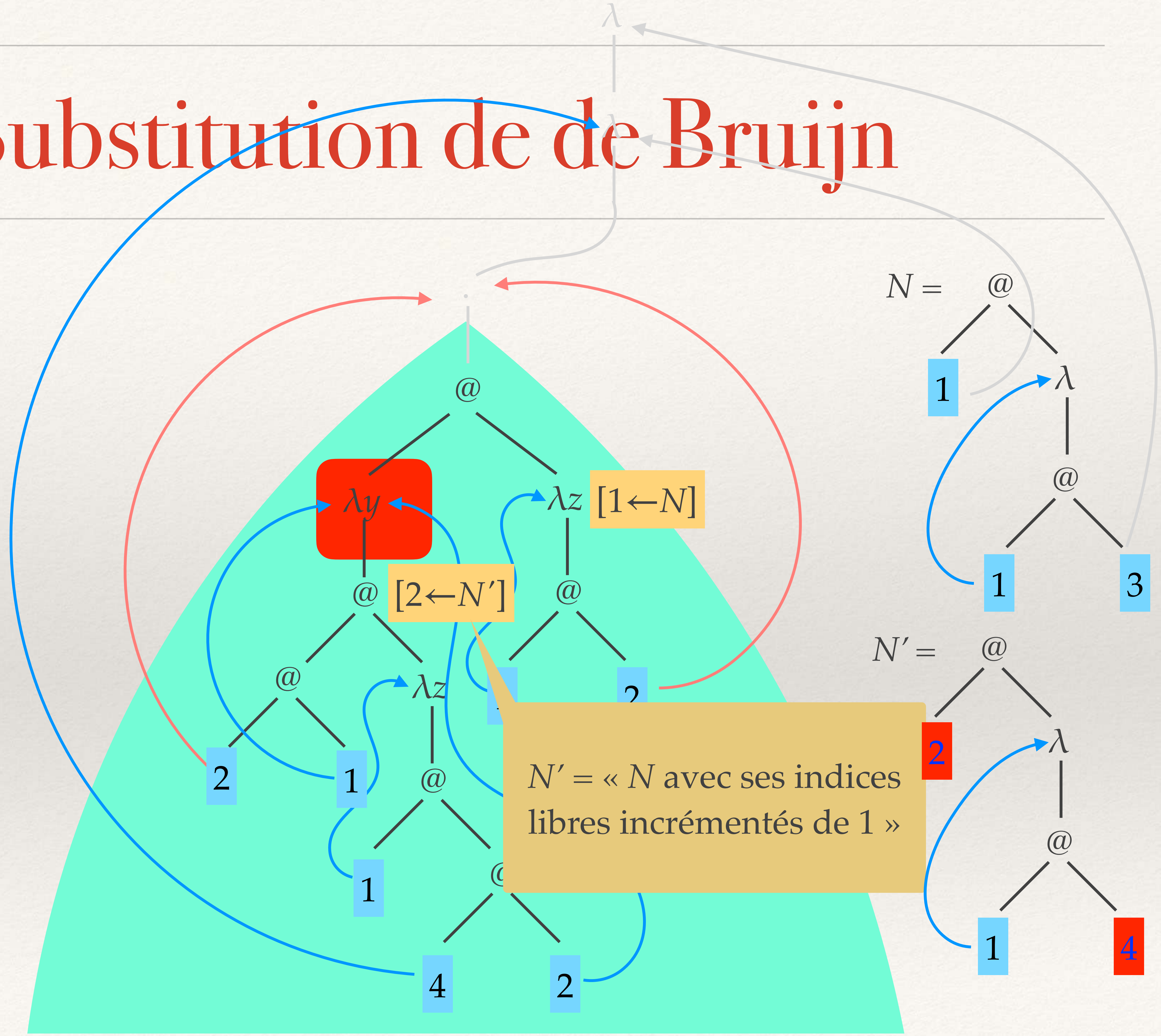
Substitution de de Bruijn



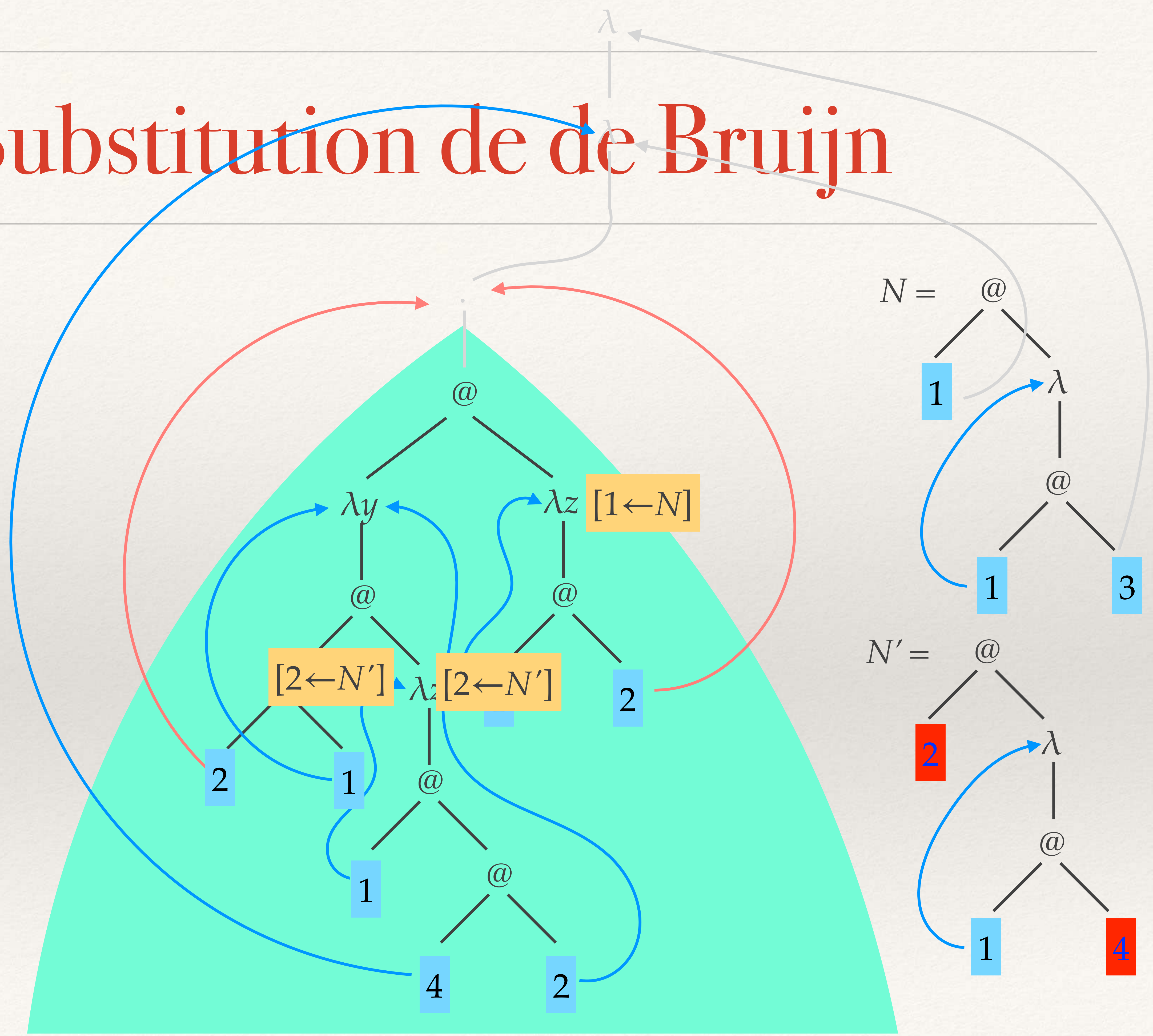
car à partir d'ici,
la variable à remplacer (x),
c'est 2

… non, pas N, il faut
ajuster ses indices de de
Bruijn

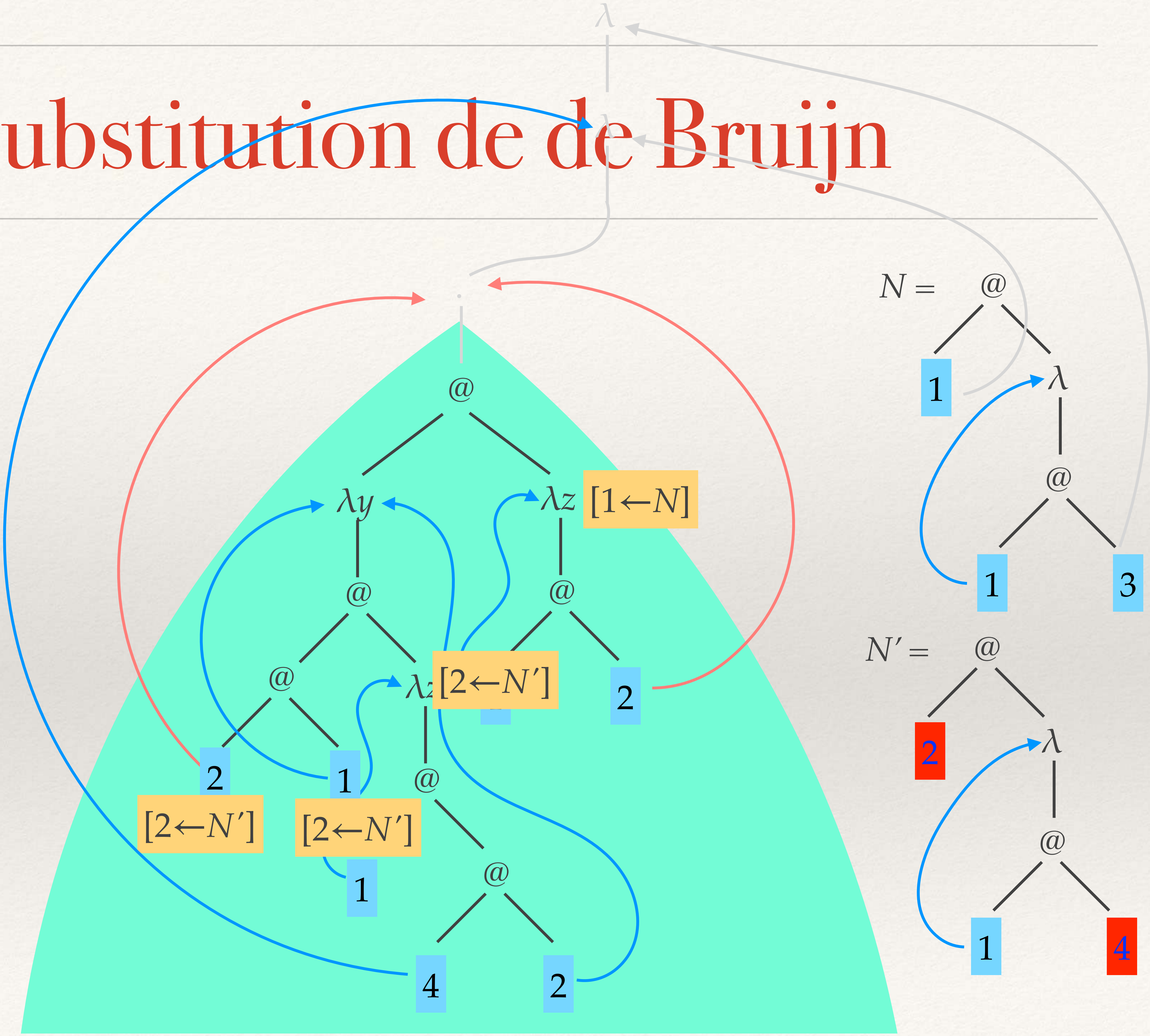
Substitution de de Bruijn



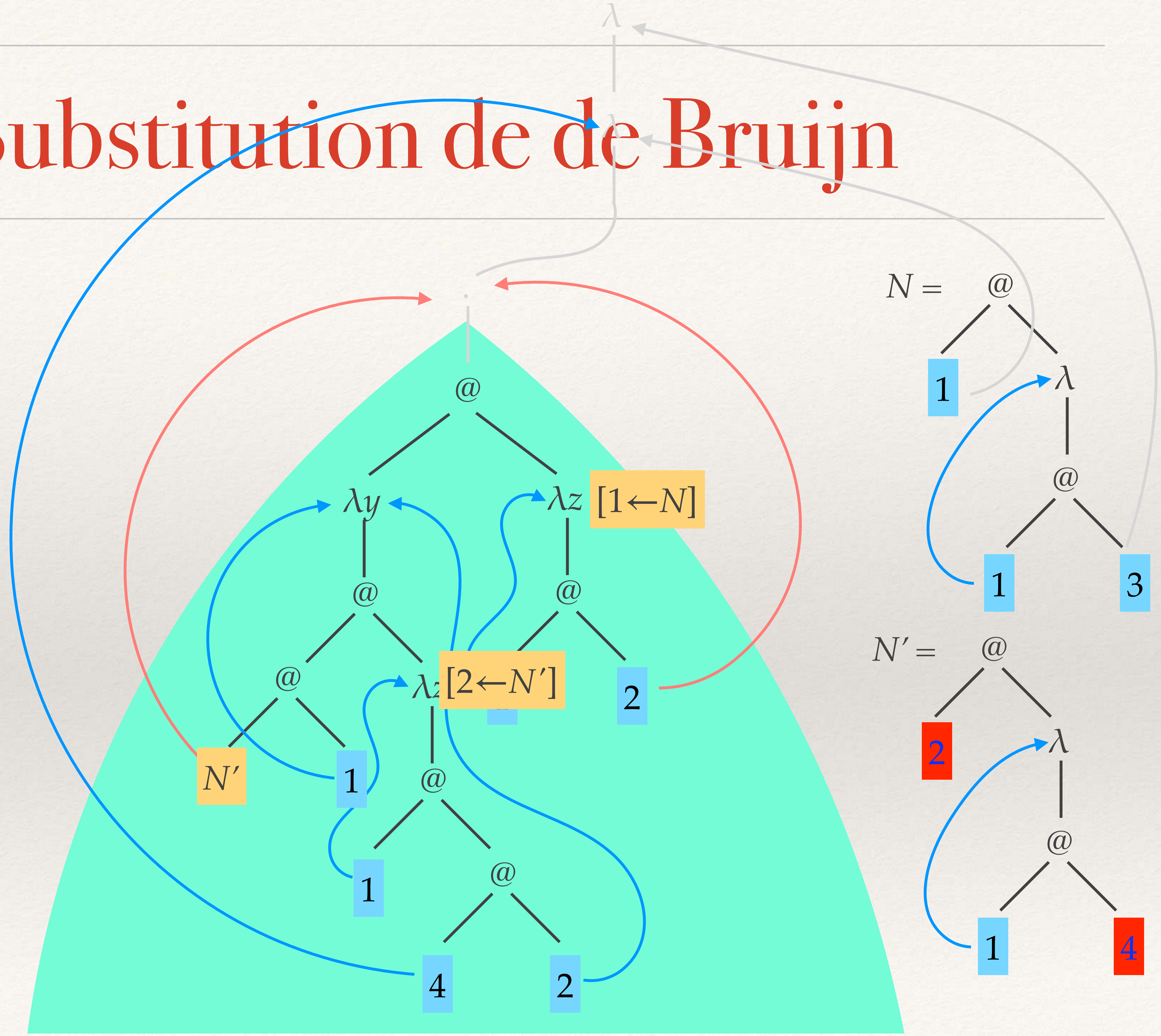
Substitution de de Bruijn



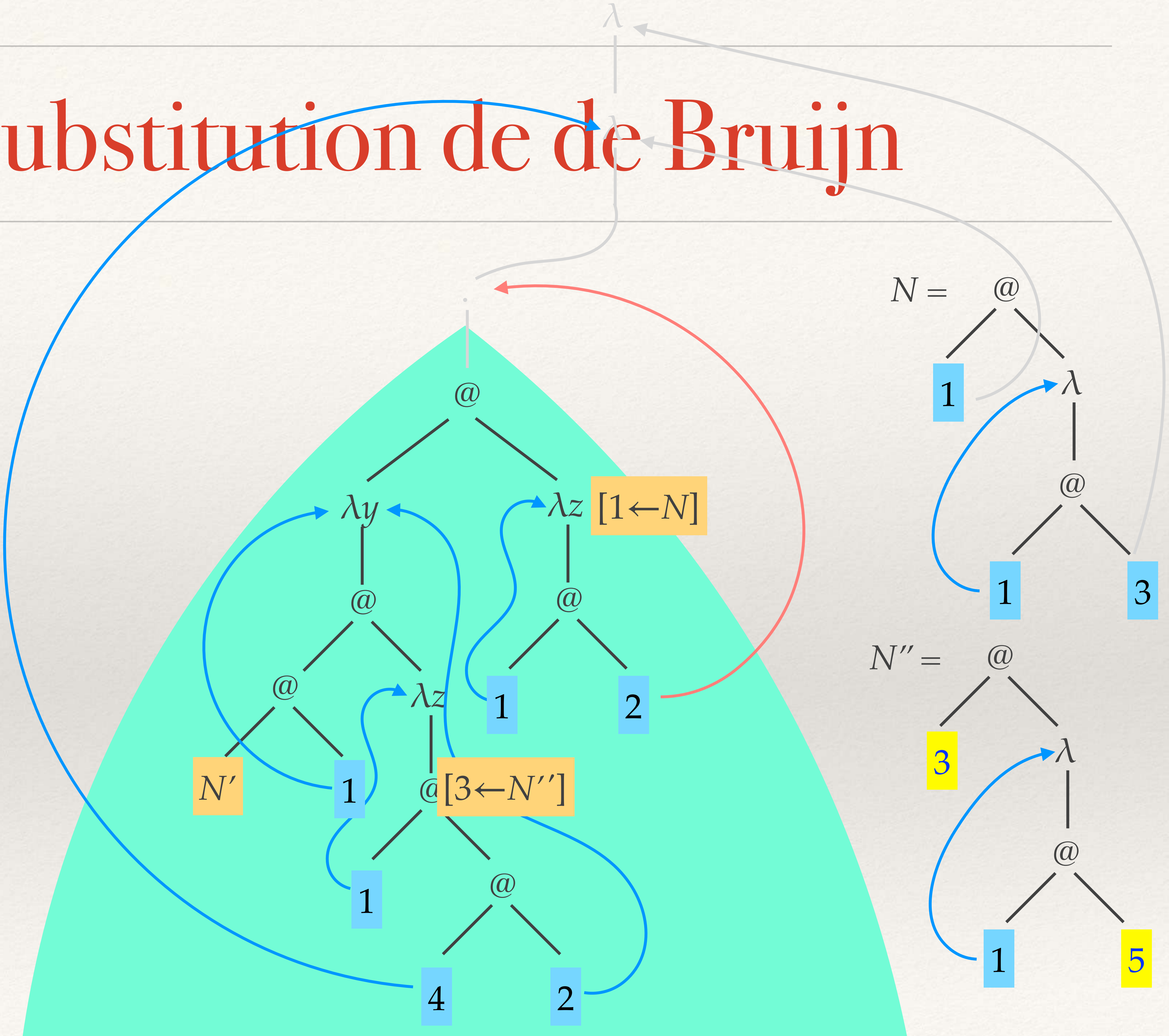
Substitution de de Bruijn



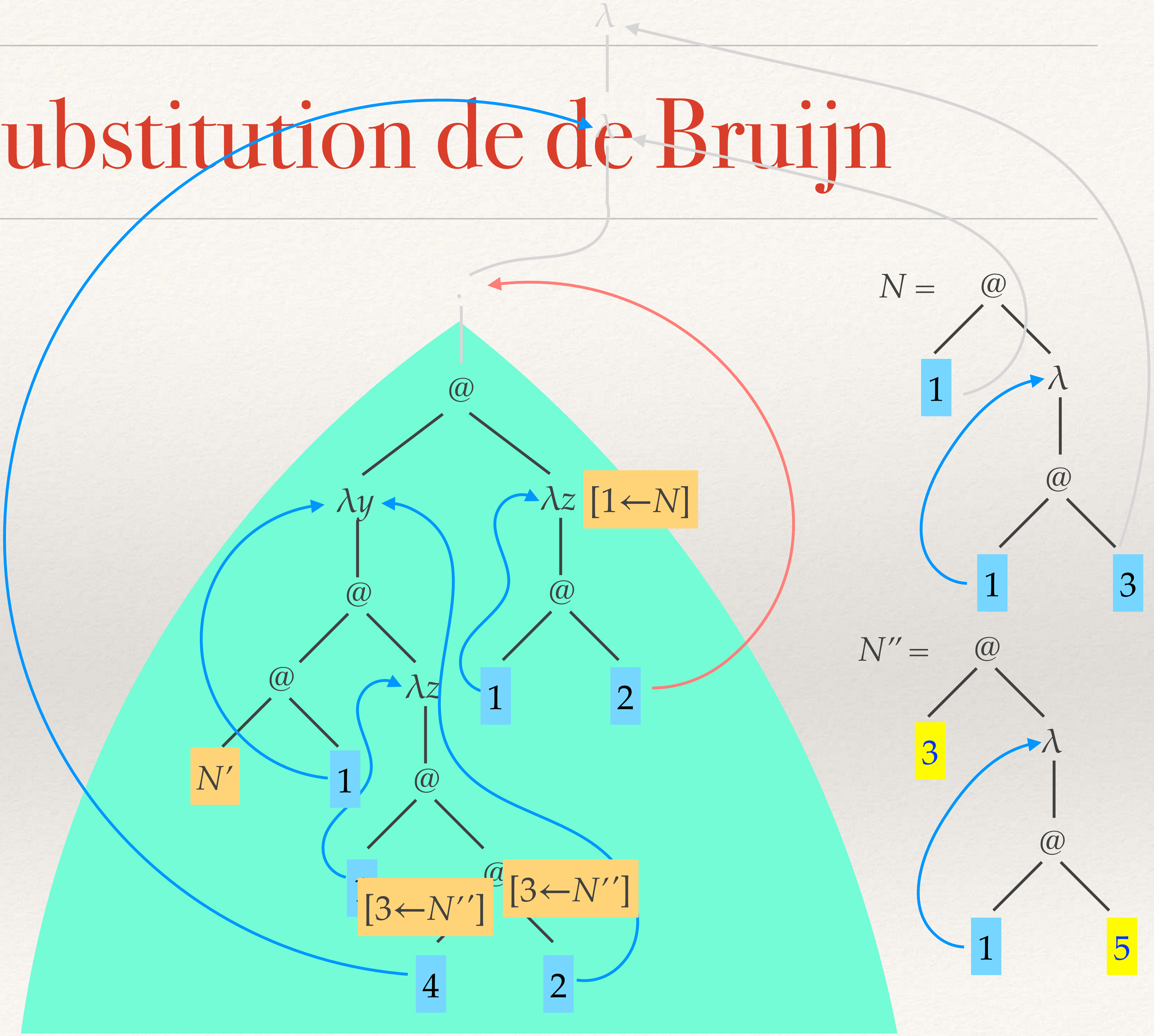
Substitution de de Bruijn



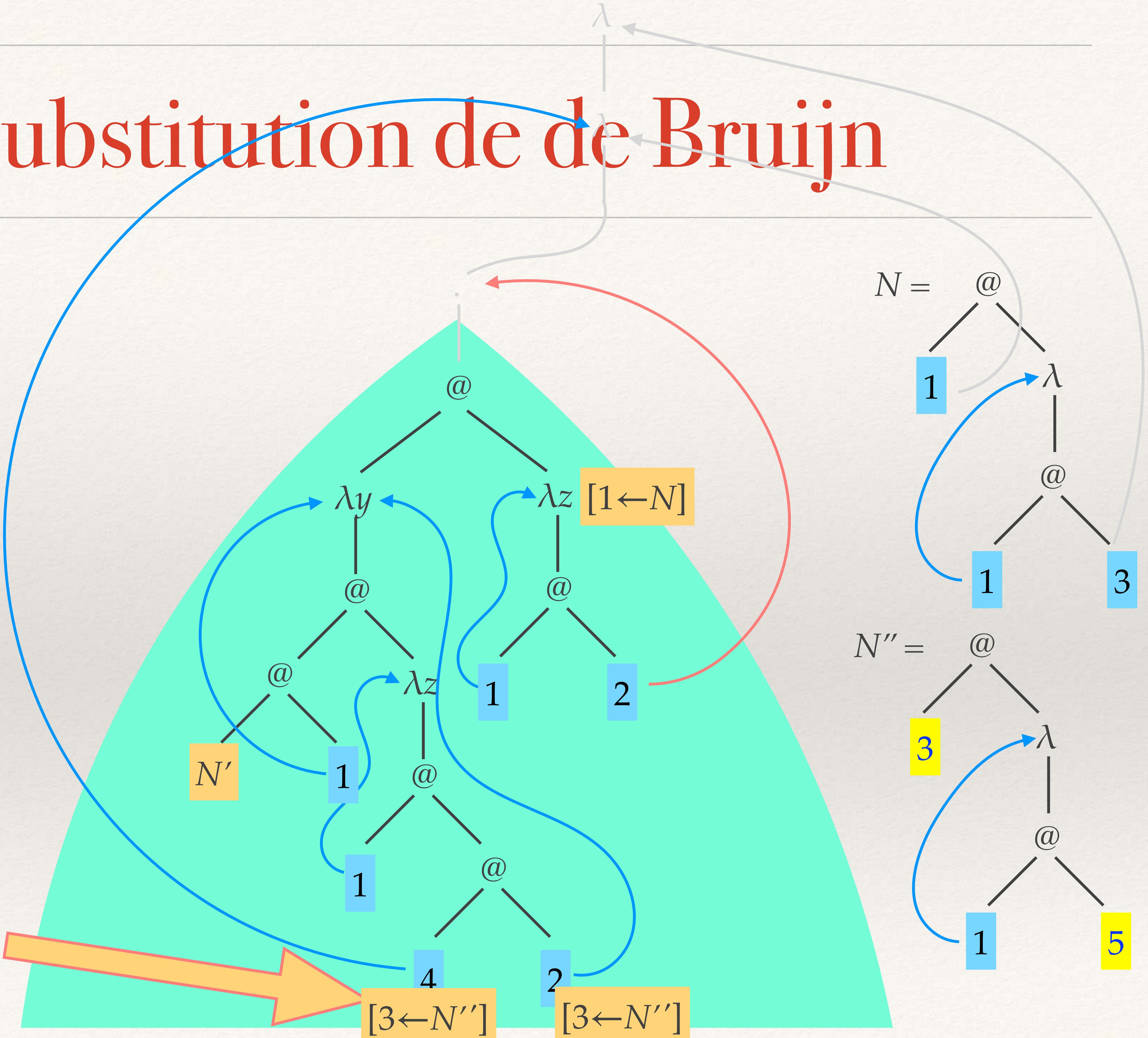
Substitution de de Bruijn



Substitution de de Bruijn

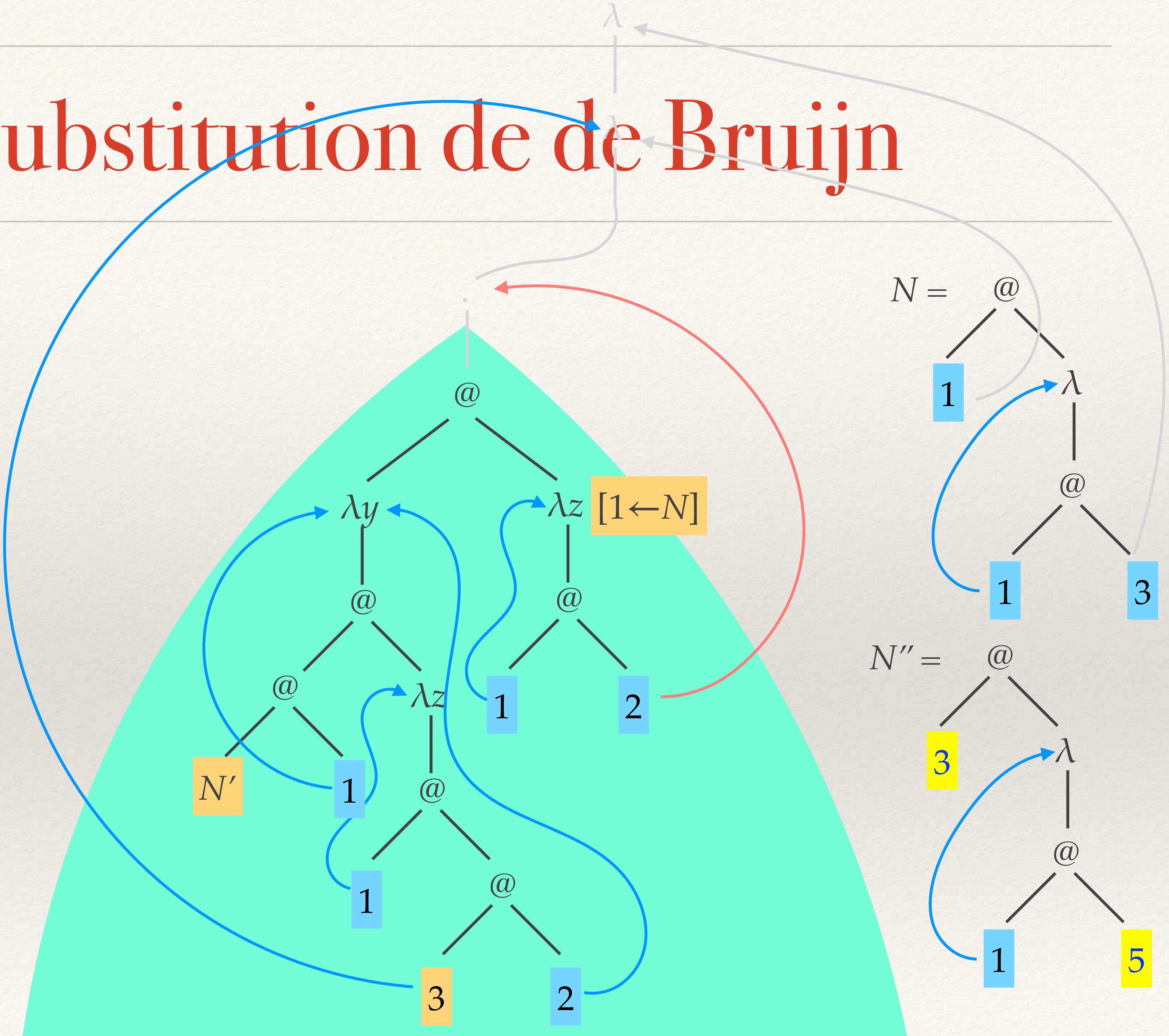


Substitution de de Bruijn

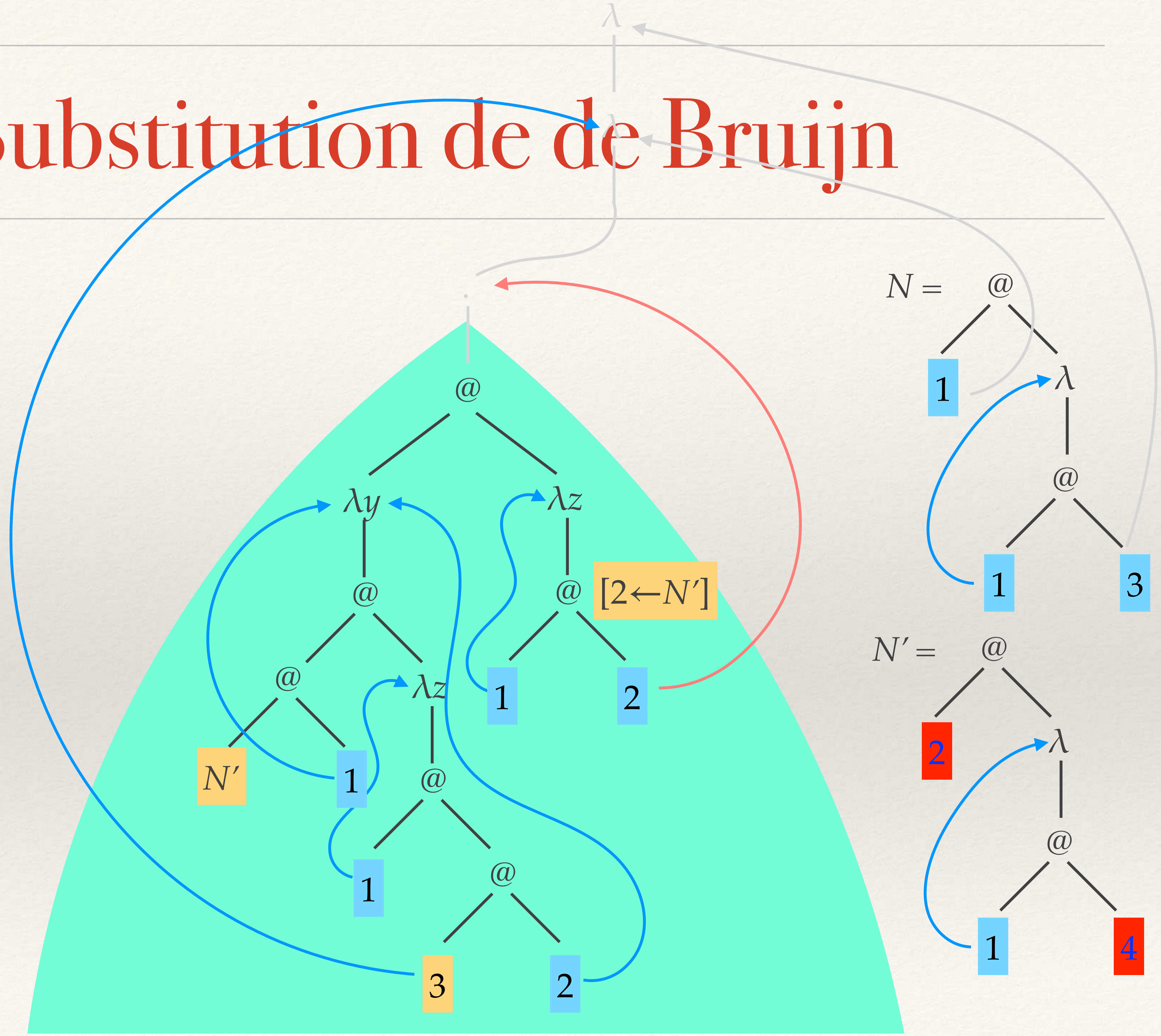


Que vaut
4 [3←N″]?

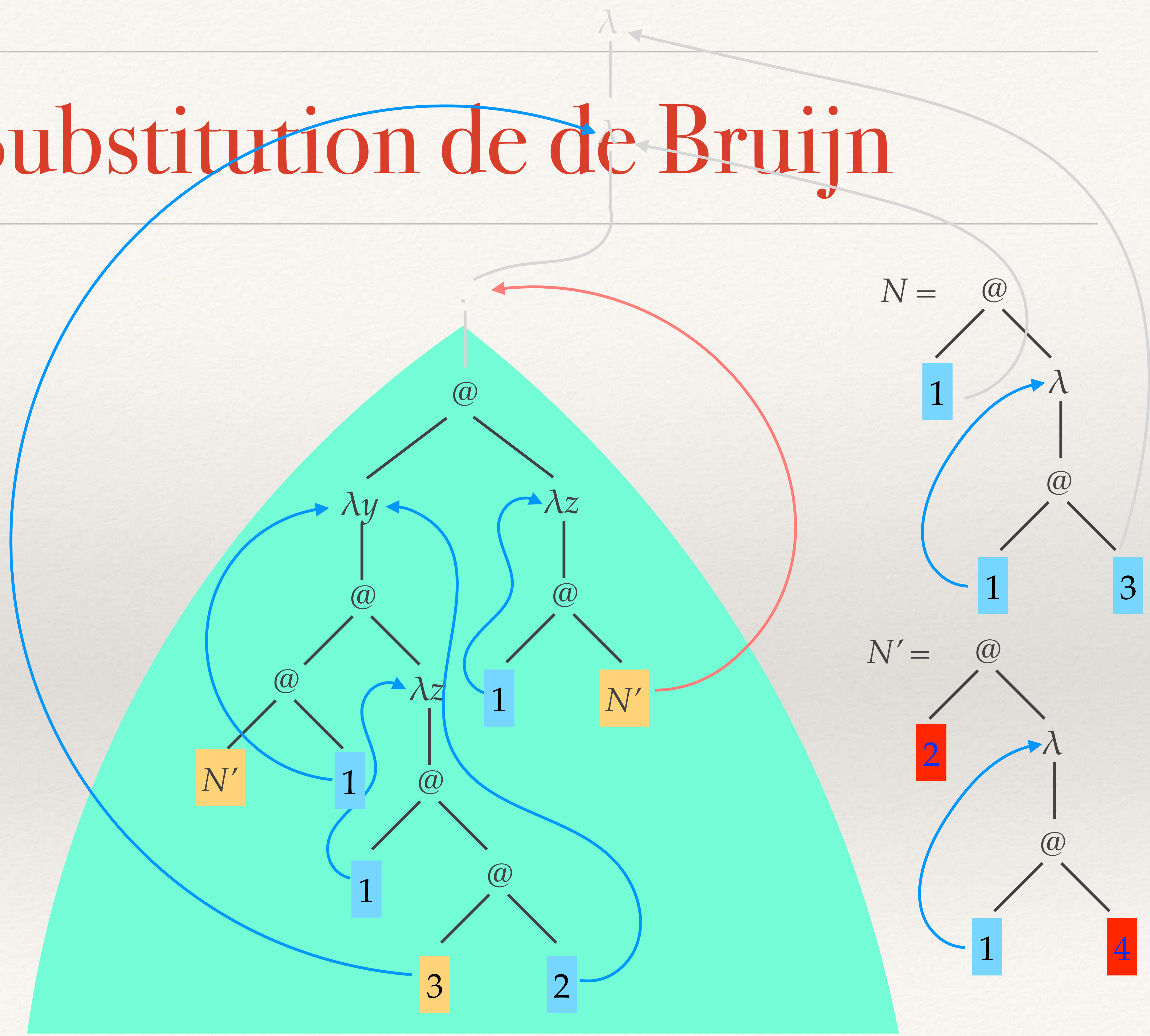
Substitution de de Bruijn



Substitution de de Bruijn



Substitution de de Bruijn



Substitution de de Bruijn, formellement

$$
\begin{array}{rcl}
(MN)[n \leftarrow P] & \hat{=} & M[n \leftarrow P]N[n \leftarrow P] \\
(\lambda M)[n \leftarrow P] & \hat{=} & \lambda(M[n+1 \leftarrow P]) \\
x[n \leftarrow P] & \hat{=} & x \\
m[n \leftarrow P] & \hat{=} & \begin{cases} m & \text{si } m < n \\ U_0^n(P) & \text{si } m = n \\ m-1 & \text{si } m > n \end{cases}
\end{array}
$$

$$
\begin{array}{rcl}
U_k^n(MN) & \hat{=} & U_k^n(M)U_k^n(N) \\
U_k^n(\lambda M) & \hat{=} & \lambda(U_{k+1}^n(M)) \\
U_k^n(x) & \hat{=} & x \\
U_k^n(p) & \hat{=} & \begin{cases} p+n-1 & \text{si } p > k \\ p & \text{sinon} \end{cases}
\end{array}
$$

Incrémente
tous les indices $> k$
(« libres »)
de n–1

- β-réduction: $(\lambda M)N \rightarrow M[1 \leftarrow N]$

Traductions de et vers de Bruijn

- Paramétrées par une liste ℓ=[x_1; …; x_n] de variables   [suffisamment longue]
  (idée: on va remplacer la variable x_i par l'indice de de Bruijn i)

<u>$\lambda \to$ de Bruijn</u>

- $x^{db}(\ell) = i$ si $x = x_i$ (avec i minimal)
  … et x si pas dans ℓ
  $(uv)^{db}(\ell) = u^{db}(\ell)\,(v^{db}(\ell))$
  $(\lambda x\,.\,u)^{db}(\ell) = \lambda(u^{db}(x::\ell))$
- **Prop.** si $u \to u'$
  alors $u^{db}(\ell) \to u'^{db}(\ell)$

<u>de Bruijn $\to \lambda$</u>

- $x^{o}(\ell) = x$
  $i^{o}(\ell) = x_i$
  $(MN)^{o}(\ell) = M^{o}(\ell)\,(N^{o}(\ell))$
  $(\lambda M)^{o}(\ell) = \lambda x\,.\,(M^{o}(x::\ell))$
- **Prop.** si $M \to M'$
  alors $M^{o}(\ell) \to M'^{o}(\ell)$

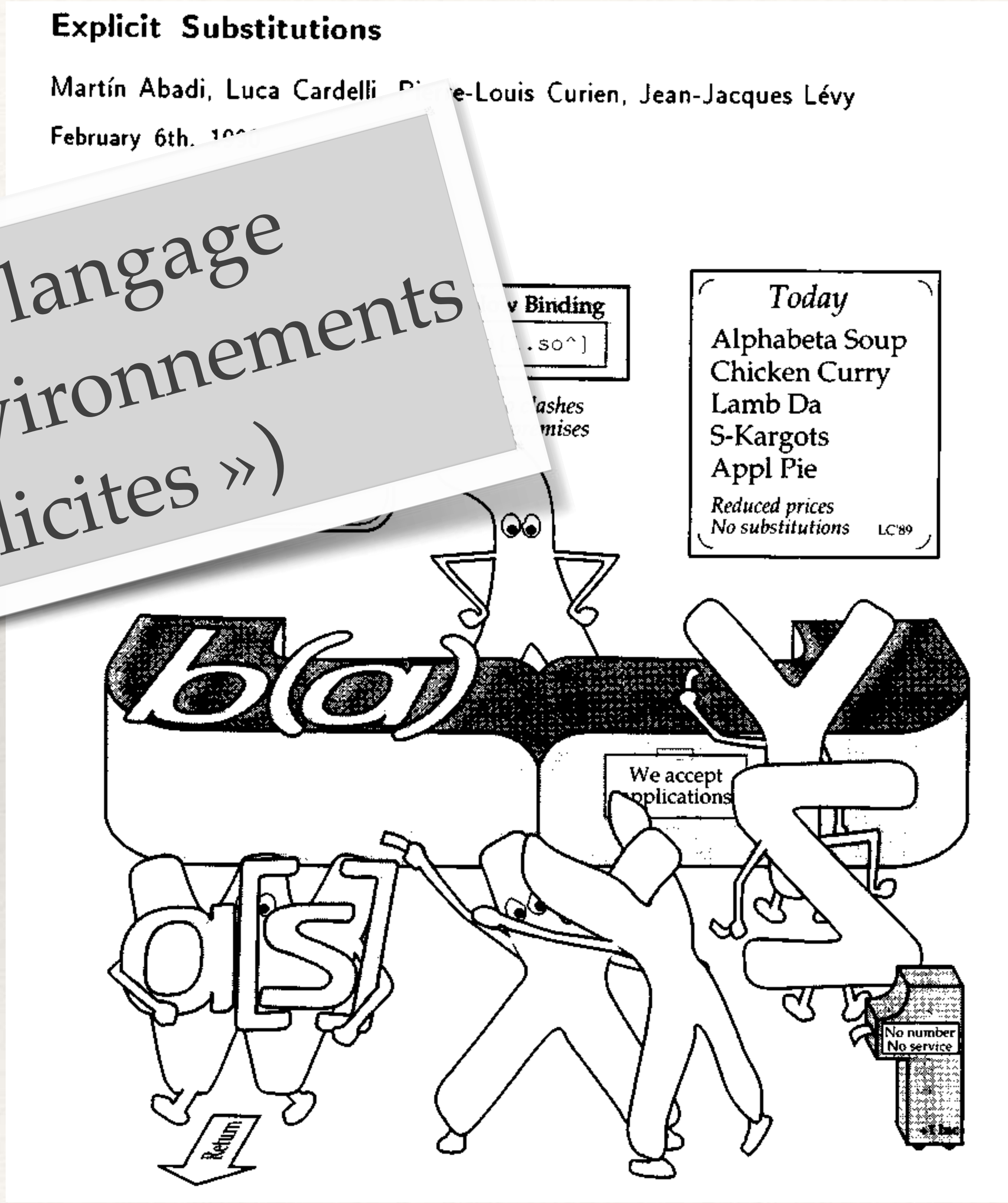
- … et les deux traductions sont inverses l'une de l'autre.

Les calculs à substitutions explicites

- … à commencer par le **λσ-calcul**
- gère les **en**…
  à l…
- résc…
  d'*α*-…nage…uivant une ancienne ide…
  N. G. de Bruijn

A faire: on va **enrichir** le langage
avec des constructions d'environnements
(= « substitutions explicites »)

Fait



Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= ?$

Il manque 2, 3, …?
Non, ils seront définissables.

Nouvelle construction
interne au langage:
dans l'environnement S »

Par exemple,
$1(1[2 \cdot \uparrow])[\mathrm{id}]$
sera un terme de notre langage,
pas une notation pour 12
(qui sera sa forme normale)

On va définir les
« environnements »
= **substitutions explicites**
= **piles**
petit à petit

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= ?$
- $(\beta)\ (\lambda M)N \longrightarrow M[N \cdot \mathrm{id}]$

Imaginons qu'une pile S soit une notation pour une liste infinie $[1 \leftarrow N_1, 2 \leftarrow N_2, \ldots]$

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid ?$
- (β) $(\lambda M)N \longrightarrow M[N \cdot \mathrm{id}]$

Ça, c'est la pile représentant la liste infinie $[1 \leftarrow 1, 2 \leftarrow 2, \ldots]$

Imaginons qu'une pile S soit une notation pour une liste infinie $[1 \leftarrow N_1, 2 \leftarrow N_2, \ldots]$

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid ?$
- $(\beta)\ (\lambda M)N \longrightarrow M[N \cdot \mathrm{id}]$

Ça, c'est la pile représentant la liste infinie $[1{\leftarrow}1, 2{\leftarrow}2, \ldots]$

Imaginons qu'une pile S soit une notation pour une liste infinie $[1{\leftarrow}N_1, 2{\leftarrow}N_2, \ldots]$

Si S représente $[1{\leftarrow}N_1, 2{\leftarrow}N_2, \ldots]$, alors $N \cdot S$ représente $[1{\leftarrow}N, 2{\leftarrow}N_1, 3{\leftarrow}N_2, \ldots]$

…donc $N \cdot \mathrm{id}$ représente $[1{\leftarrow}N, 2{\leftarrow}1, 3{\leftarrow}2, \ldots]$

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid ?$

$\cdot\ \mathrm{id}]$

Il manque 2, 3, …?
Non, ils seront définissables.

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid ?$

$\cdot$ id

Il manque 2, 3, …?
Non, ils seront définissables.

Le **shift**: représente
$[1 \leftarrow 2, 2 \leftarrow 3, \ldots]$

- On pose: $2 = 1[\uparrow]$

(Introduire le shift ↑
plutôt que 2 directement
résoudra aussi
un autre problème
que nous verrons plus tard)

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$

$\cdot$ id

Il manque 2, 3, …?
Non, ils seront définissables.

On aurait pu aussi poser
$3 = 1[\uparrow][\uparrow]$,
$4 = 1[\uparrow][\uparrow][\uparrow]$, etc.
C'est un choix; en $\lambda\sigma$,
ces derniers termes ne seraient pas normaux.

- On pose: $2 = 1[\uparrow]$
  $3 = 1[\uparrow \circ \uparrow]$
  $4 = 1[\uparrow \circ (\uparrow \circ \uparrow)]$
  etc.

Composition.
Par exemple, $\uparrow \circ \uparrow$ représente
$[1 \leftarrow 3, 2 \leftarrow 4, \ldots]$

et $\uparrow \circ (\uparrow \circ \uparrow)$ représente
$[1 \leftarrow 4, 2 \leftarrow 5, \ldots]$

Le λσ-calcul, progressivement

Il ne reste plus qu'à écrire des règles permettant de faire descendre la substitution explicite dans M

- $M, N, \ldots ::= x \mid 1 \mid MN \mid$ …
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid$ …
- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- .
  .
  .
  .
  .
  .
  .

(à remplir)

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$
- (β) $(\lambda M)N \longrightarrow M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \longrightarrow N$
  .
  .
  .
  .
  .
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$
- $(\beta)\ (\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$
  . $(MN)[S] \to M[S](N[S])$
  .
  .
  .
  .
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid$

Ça, c'est difficile à trouver.
Faisons-le plus tard

- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$
  . $(MN)[S] \to M[S](N[S])$
  . $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$
  .
  .
  .
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$
- $(\beta)\ (\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$
  . $(MN)[S] \to M[S](N[S])$
  . $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$
  . $M[S][S'] \to M[S \circ S']$
  .
  .
  .

La composition $\circ$,
c'est comme [].
Il faut aussi des règles pour
faire descendre S' dans S!

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$

Vous remarquerez la ressemblance avec les règles de projection des couples…

- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$
- . $1[N \cdot S] \to N$ $\qquad$ $\uparrow \circ (N \cdot S) \to S$
  . $(MN)[S] \to M[S](N[S])$
  . $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$
  . $M[S][S'] \to M[S \circ S']$
  .
  .
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$
- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$ $\quad$ $\uparrow \circ (N \cdot S) \to S$
  . $(MN)[S] \to M[S](N[S])$ $\quad$ $(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
  . $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$
  . $M[S][S'] \to M[S \circ S']$
  .
  .
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$

Oui, ça ressemble à l'associativité (orientée)

- (β) $(\lambda M)N \longrightarrow$

Mais c'est aussi similaire à la règle
$M[S][S'] \longrightarrow M[S \circ S']$
si on identifie [] à ∘

- $.\ 1[N \cdot S] \longrightarrow N$ $) \longrightarrow S$
  $.\ (MN)[S] \longrightarrow$ $S' \longrightarrow N[S'] \cdot (S \circ S')$
  $.\ (\lambda M)[S] \longrightarrow \lambda(M[\quad ? \quad])$
  $.\ M[S][S'] \longrightarrow M[S \circ S']$ $\qquad (S \circ S') \circ S'' \longrightarrow S \circ (S' \circ S'')$
  .
  .
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$

- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- $. \; 1[N \cdot S] \to N$
  $. \; (MN)[S] \to M[S](N[S])$
  $. \; (\lambda M)[S] \to \lambda(M[\quad ? \quad])$
  $. \; M[S][S'] \to M[S \circ S']$
  .
  .
  .

$\uparrow \circ (N \cdot S) \to S$

$(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$

$(S \circ S') \circ S'' \to S \circ (S' \circ S'')$

$\mathrm{id} \circ S \to S$

La règle $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$

- Si S représente $[1 \leftarrow N_1, 2 \leftarrow N_2, \ldots]$
- on veut que, dans la pile inconnue:
- $1 \leftarrow 1$
- $2 \leftarrow N_1 \ldots$

La règle $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$

- Si S représente $[1 \leftarrow N_1, 2 \leftarrow N_2, \ldots]$
- on veut que, dans la pile inconnue:
- $1 \leftarrow 1$
- $2 \leftarrow N_1$ … euh, non, il faut incrémenter
  tous les indices de N_1 de 1
- C’est à ça que sert le shift $\uparrow$!

La règle $(\lambda M)[S] \to \lambda(M[\quad ? \quad])$

- Si S représente $[1 \leftarrow N_1, 2 \leftarrow N_2, \ldots]$
- on veut que, dans la pile inconnu
- $1 \leftarrow 1$
- $2 \leftarrow N_1[\uparrow]$: pour incrémenter tous les indices de N_1 de 1
- $3 \leftarrow N_2[\uparrow]$
- etc.
- La pile inconnue est donc: $1 \cdot (S \circ \uparrow)$

Tout ça se représente
par la pile
$S \circ \uparrow$

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$
- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$ $\qquad \uparrow \circ (N \cdot S) \to S$
  . $(MN)[S] \to M[S](N[S])$ $\qquad (N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
  . $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$
  . $M[S][S'] \to M[S \circ S']$ $\qquad (S \circ S') \circ S'' \to S \circ (S' \circ S'')$
  . $\qquad \mathrm{id} \circ S \to S$
  .
  .

Le $\lambda\sigma$-calcul progressivement

Sinon, $(\lambda 2)N \to 2[N \cdot \mathrm{id}]$

(rappel: $2=1[\uparrow]$)

$\to 1[\uparrow \circ (N \cdot \mathrm{id})]$

$\to 1[\mathrm{id}]$

… qui serait en forme normale

- M … $\lambda M \mid M[S]$
  S … $\mid S \circ S'$
- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$ $\qquad \uparrow \circ (N \cdot S) \to S$
  . $(MN)[S] \to M[S](N[S])$ $\qquad (N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
  . $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$
  . $M[S][S'] \to M[S \circ S']$ $\qquad (S \circ S') \circ S'' \to S \circ (S' \circ S'')$
  . $\qquad \mathrm{id} \circ S \to S$
  . $M[\mathrm{id}] \to M$
  .

Le λσ-calcul, progressivement

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$
- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

(σ)

- . $1[N \cdot S] \to N$ $\uparrow \circ (N \cdot S) \to S$
  . $(MN)[S] \to M[S](N[S])$ $(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
  . $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$
  . $M[S][S'] \to M[S \circ S']$ $(S \circ S') \circ S'' \to S \circ (S' \circ S'')$
  . $\mathrm{id} \circ S \to S$
  . $M[\mathrm{id}] \to M$ $S \circ \mathrm{id} \to S$
  .

Problèmes de confluence 1

$$
\begin{tikzcd}
 & (\lambda M)[\mathrm{id}] \arrow[dl] \arrow[dr] & \\
\lambda(M[1 \cdot (\mathrm{id} \circ \uparrow)]) \arrow[d] & & \lambda M \\
\lambda(M[1 \cdot \uparrow]) & &
\end{tikzcd}
$$

$(\beta)\ (\lambda M)N \to M[N \cdot \mathrm{id}]$	(σ)
. $1[N \cdot S] \to N$	$\uparrow \circ (N \cdot S) \to S$
. $(MN)[S] \to M[S](N[S])$	$(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
. $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$	
. $M[S][S'] \to M[S \circ S']$	$(S \circ S') \circ S'' \to S \circ (S' \circ S'')$
.	$\mathrm{id} \circ S \to S$
. $M[\mathrm{id}] \to M$	$S \circ \mathrm{id} \to S$
.	

… et là, pas de réduit commun en général
(prendre $M{=}x$)

Problèmes de confluence 1

$$\begin{tikzcd}[column sep=small]
 & (\lambda M)[\mathrm{id}] \arrow[dl] \arrow[dr] & \\
\lambda(M[1 \cdot (\mathrm{id} \circ \uparrow)]) \arrow[d] & & \lambda M \\
\lambda(M[1 \cdot \uparrow]) & &
\end{tikzcd}$$

(β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$	(σ)
. $1[N \cdot S] \to N$	$\uparrow \circ (N \cdot S) \to S$
. $(MN)[S] \to M[S](N[S])$	$(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
. $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$	
. $M[S][S'] \to M[S \circ S']$	$(S \circ S') \circ S'' \to S \circ (S' \circ S'')$
.	$\mathrm{id} \circ S \to S$
. $M[\mathrm{id}] \to M$	$S \circ \mathrm{id} \to S$
.	

Il y a une procédure (Knuth-Bendix) qui répare ce genre de problèmes. Ici, elle rajouterait la règle:

$$\lambda(M[1 \cdot \uparrow]) \to \lambda M$$

Nous allons plutôt rajouter la règle $1 \cdot \uparrow \to \mathrm{id}$

Problèmes de confluence 2

En présence de $1 \cdot \uparrow \to \mathrm{id}$:

$$\begin{tikzcd}[column sep=small]
 & (1 \cdot \uparrow) \circ S \arrow[dl] \arrow[dr] & \\
1[S] \cdot (\uparrow \circ S) & & \mathrm{id} \circ S \arrow[d] \\
 & & S
\end{tikzcd}$$

(β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$	(σ)
. $1[N \cdot S] \to N$	$\uparrow \circ (N \cdot S) \to S$
. $(MN)[S] \to M[S](N[S])$	$(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
. $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$	
. $M[S][S'] \to M[S \circ S']$	$(S \circ S') \circ S'' \to S \circ (S' \circ S'')$
.	$\mathrm{id} \circ S \to S$
. $M[\mathrm{id}] \to M$	$S \circ \mathrm{id} \to S$
.	

On rajoute la règle: $1[S] \cdot (\uparrow \circ S) \to S$

Le λσ-calcul

- $M, N, \ldots ::= x \mid 1 \mid MN \mid \lambda M \mid M[S]$
  $S, S', \ldots ::= \mathrm{id} \mid N \cdot S \mid \uparrow \mid S \circ S'$

- (β) $(\lambda M)N \to M[N \cdot \mathrm{id}]$

- . $1[N \cdot S] \to N$ $\qquad$ $\uparrow \circ (N \cdot S) \to S$
  . $(MN)[S] \to M[S](N[S])$ $\qquad$ $(N \cdot S) \circ S' \to N[S'] \cdot (S \circ S')$
  . $(\lambda M)[S] \to \lambda(M[1 \cdot (S \circ \uparrow)])$
  . $M[S][S'] \to M[S \circ S']$ $\qquad$ $(S \circ S') \circ S'' \to S \circ (S' \circ S'')$
  . $\qquad$ $\mathrm{id} \circ S \to S$
  . $M[\mathrm{id}] \to M$ $\qquad$ $S \circ \mathrm{id} \to S$
  . $1 \cdot \uparrow \to \mathrm{id}$ $\qquad$ $1[S] \cdot (\uparrow \circ S) \to S$

(σ)

C'est fini!
On a trouvé σ.