Principe de transposition et tours d'Artin-Schreier

L. De Feo

LIX, École Polytechnique

20 octobre 2008

Principe de transposition

L. De Feo

LIX, École Polytechnique

20 octobre 2008

Le principe de Tellegen

"De tout *algorithme linéaire* qui calcule une application linéaire on peut déduire an autre *algorithme linéaire* qui calcule la transposée de l'application ayant *à peu près* les mêmes complexités en espace et temps."

Le principe de Tellegen

"De tout *algorithme linéaire* qui calcule une application linéaire on peut déduire an autre *algorithme linéaire* qui calcule la transposée de l'application ayant *à peu près* les mêmes complexités en espace et temps."

Qu'est-ce qu'il y a de tellement spécial dans la transposition ?

Théorie des catégories, Généralités

- Catégorie $\mathscr{C}$
- Objets $\mathrm{ob}(\mathscr{C})$

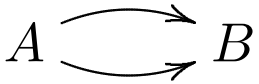
A B

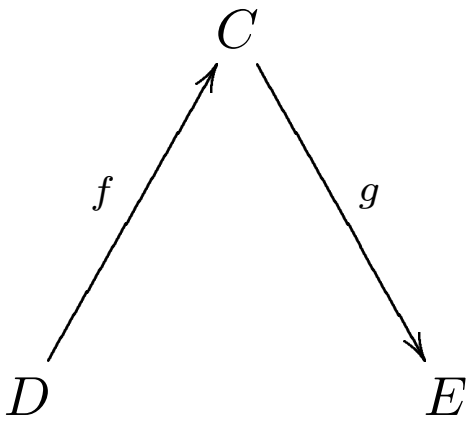
C

D E

Théorie des catégories, Généralités

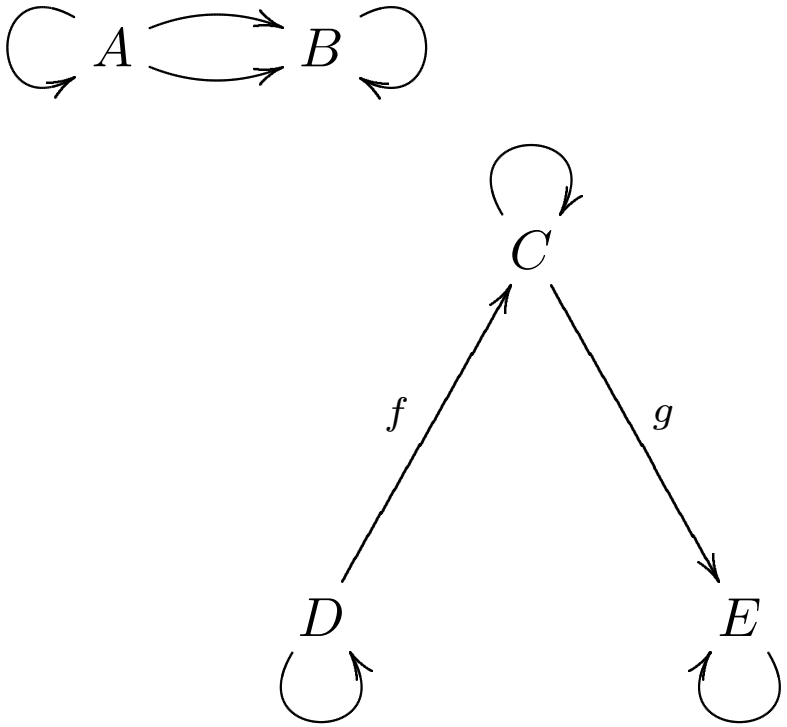
- Catégorie $\mathscr{C}$
- Objets $\mathrm{ob}(\mathscr{C})$
- Flèches $\hom(\mathscr{C})$, $\mathrm{Hom}(A, B)$





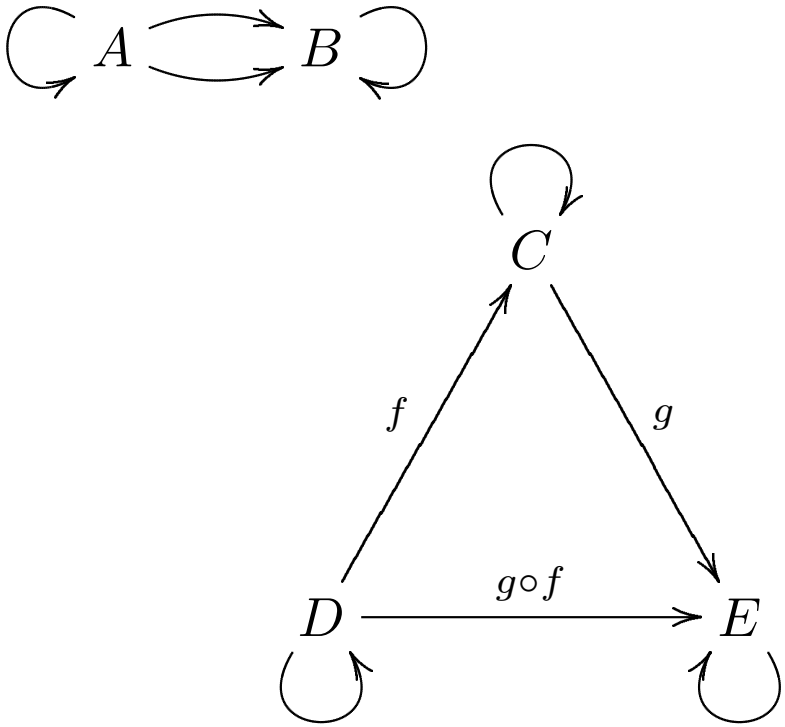
Théorie des catégories, Généralités

- Catégorie $\mathscr{C}$
- Objets $\mathrm{ob}(\mathscr{C})$
- Flèches $\hom(\mathscr{C})$, $\mathrm{Hom}(A, B)$
- Identités id_A



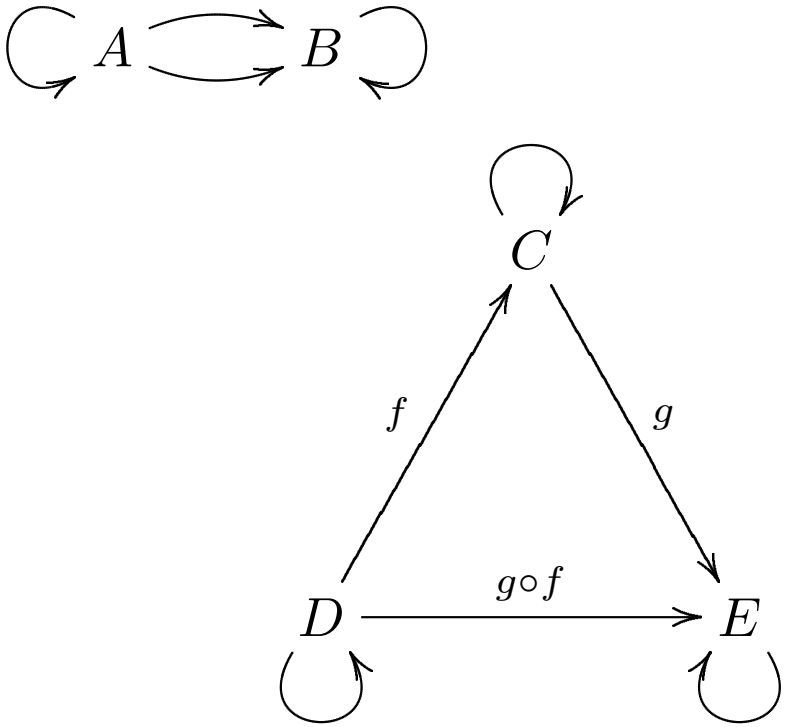
Théorie des catégories, Généralités

- Catégorie $\mathscr{C}$
- Objets $\mathrm{ob}(\mathscr{C})$
- Flèches $\hom(\mathscr{C})$, $\mathrm{Hom}(A, B)$
- Identités id_A
- Composition $g \circ f$



Théorie des catégories, Généralités

- Catégorie $\mathscr{C}$
- Objets $\mathrm{ob}(\mathscr{C})$
- Flèches $\hom(\mathscr{C})$, $\mathrm{Hom}(A, B)$
- Identités id_A
- Composition $g \circ f$

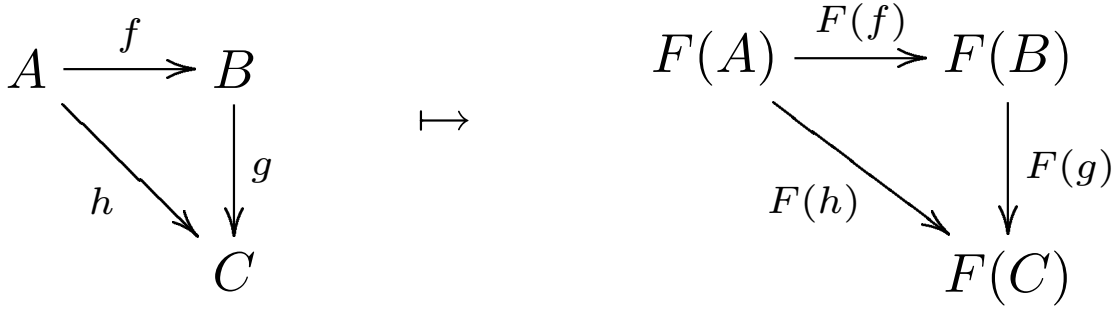


Exemple : $\mathbf{FMod}_R$

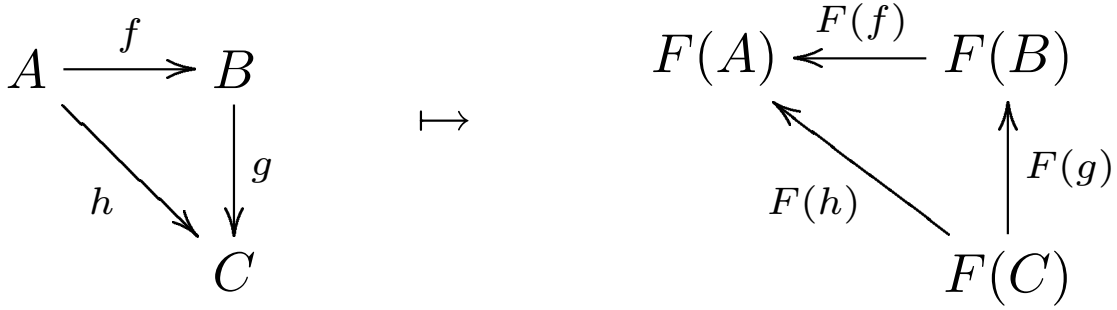
- $\mathrm{ob}(\mathscr{C}) = R^n$ les R-modules libres,
- $\hom(\mathscr{C}) =$ les applications linéaires.

Théorie des catégories, Foncteurs

Foncteur covariant
$F : \mathscr{C} \to \mathscr{D}$



Foncteur contravariant
$F : \mathscr{C} \to \mathscr{D}$



Équivalence, dualité

- Équivalence si $F : \mathscr{C} \to \mathscr{D}$ et $G : \mathscr{D} \to \mathscr{C}$ covariants
- Dualité si $F : \mathscr{C} \to \mathscr{D}$ et $G : \mathscr{D} \to \mathscr{C}$ contravariants

et $F \circ G \simeq \mathrm{Id}_{\mathscr{D}}$ et $G \circ F \simeq \mathrm{Id}_{\mathscr{C}}$.

Le principe de Tellegen

"De tout *algorithme linéaire* qui calcule une application linéaire on peut déduire an autre *algorithme linéaire* qui calcule la transposée de l'application ayant *à peu près* les mêmes complexités en espace et temps."

Un exemple

```
for i = 1 to n-2 do
  a[i+1] = a[i] + a[i+1]
  a[i] = 0
end for
```

$$\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{pmatrix}$$

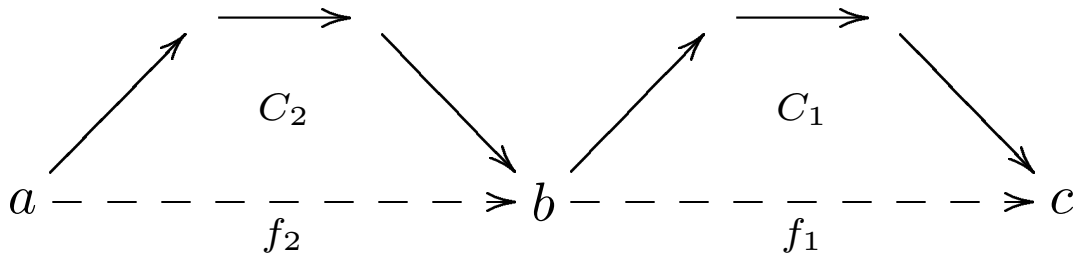
Calculs

Langage, taille

- Ensemble d'*instructions* $\mathscr{L} \subset \hom(\mathscr{C})$,
- Fonction de taille $\|.\| : \mathrm{ob}(\mathscr{C}) \to \mathbb{N}$.

Calcul

Suite $C_1 : b \to c$ d'instructions.



Coûts en temps et espace

- $\mathrm{t}(C) =$ longueur du calcul,
- $\mathrm{s}(C) = \max_{o \in C} \|o\|$.

Le cas de $\mathbf{FMod}_R$

A. Bostan, G. Lecerf, E. Schost :

$$+_1 : R^2 \to R^2 \qquad (p,q) \mapsto (p+q,q)$$

$$+_2 : R^2 \to R^2 \qquad (p,q) \mapsto (p,p+q)$$

$$*_a : R \to R \qquad p \mapsto ap$$

$$\pi : R \to 0 \qquad p \mapsto 0$$

$$\iota : 0 \to R \qquad 0 \mapsto 0$$

$$\mathscr{L} = \left\{ \mathrm{Id}_n \times \mathrm{op} \times \mathrm{Id}_m \middle| n, m \in \mathbb{N}, \mathrm{op} \in \{+_1, +_2, *_a, \pi, \iota \mid a \in R\} \right\}.$$

$$\|R^n\| = n$$

Notre exemple

```
for i = 1 to n-2 do
  a[i+1] = a[i] + a[i+1]
  a[i] = 0
end for
```

$$\mathrm{Id}_i \times +_2 \times \mathrm{Id}_{n-2-i}$$
$$\mathrm{Id}_i \times *_0 \times \mathrm{Id}_{n-1-i}$$

$$R^n \xrightarrow{+_2 \times \mathrm{Id}_{n-2}} R^n \xrightarrow{0 \times \mathrm{Id}_{n-1}} R^n \cdots\cdots\cdots\cdots\cdots\cdots\cdots R^n$$

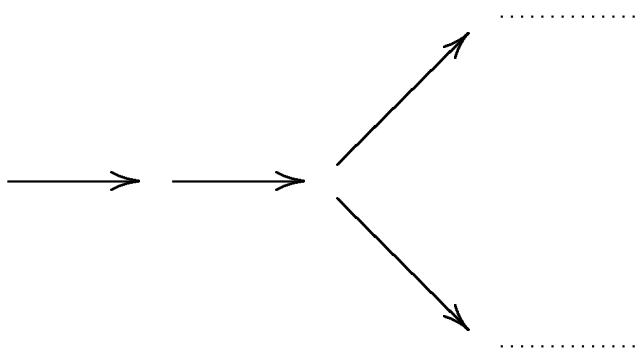
Notre exemple

```
for i = 1 to n-2 do
  a[i+1] = a[i] + a[i+1]
  a[i] = 0
end for
```

$$\mathrm{Id}_i \times +_2 \times \mathrm{Id}_{n-2-i}$$
$$\mathrm{Id}_i \times *_0 \times \mathrm{Id}_{n-1-i}$$

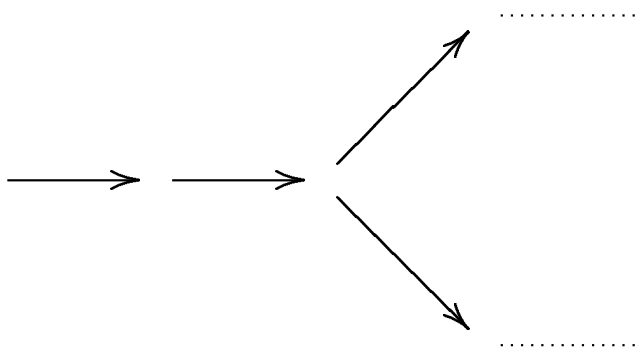
$$R^n \xrightarrow{+_2 \times \mathrm{Id}_{n-2}} R^n \xrightarrow{0 \times \mathrm{Id}_{n-1}} R^n \cdots\cdots\cdots\cdots\cdots\cdots\cdots R^n$$

Branchements



```
if a = (0,...,0) then
   ...
else
   ...
endif
```

Branchements



```
if n = 0 then
   ...
else
   ...
endif
```

Algorithme

Espace des paramètres

Par un ensemble récursivement énumérable

Par exemple, $\mathrm{Par} = \mathbb{N}$

Algorithme

Une fonction $A : \mathrm{Par} \to \mathscr{C}_{\to}$ ($\mathscr{C}_{\to}$ = les calculs)

Algorithme

Espace des paramètres

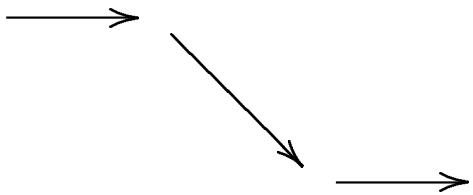
Par un ensemble récursivement énumérable

Par exemple, $\mathrm{Par} = \mathbb{N}$

Algorithme

Une fonction $A : \mathrm{Par} \to \mathscr{C}_{\to}$ ($\mathscr{C}_{\to}$ = les calculs)

1



Algorithme

Espace des paramètres

Par un ensemble récursivement énumérable

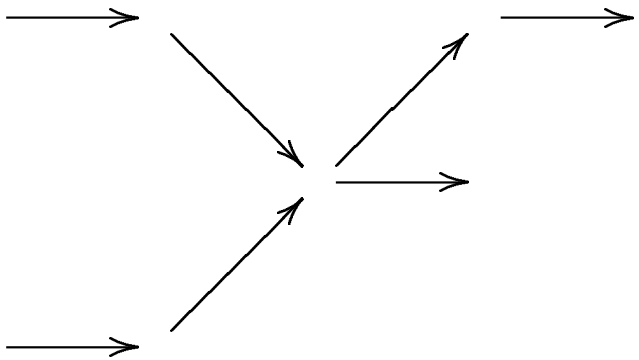
Par exemple, $\mathrm{Par} = \mathbb{N}$

Algorithme

Une fonction $A : \mathrm{Par} \to \mathscr{C}_{\to}$ ($\mathscr{C}_{\to}$ = les calculs)

1

2



Algorithme

Espace des paramètres

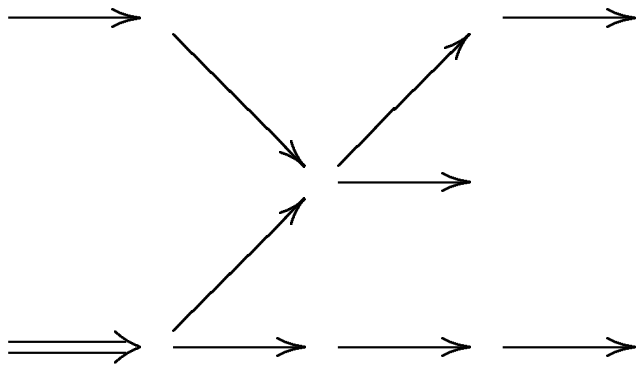
Par un ensemble récursivement énumérable

Par exemple, $\mathrm{Par} = \mathbb{N}$

Algorithme

Une fonction $A : \mathrm{Par} \to \mathscr{C}_{\to}$ ($\mathscr{C}_{\to}$ = les calculs)

1
2
3
$\vdots$



Complexité

Complexité en temps

$A : \mathrm{Par} \to \mathscr{C}_{\to}$ induit une fonction $\mathrm{t}_A : \mathrm{Par}\,\mathbb{N}$ donnée par

$$\mathrm{t}_A(x) = \mathrm{t}(A(x))$$

Complexité en espace

$A : \mathrm{Par} \to \mathscr{C}_{\to}$ induit une fonction $\mathrm{s}_A : \mathrm{Par}\,\mathbb{N}$ donnée par

$$\mathrm{s}_A(x) = \mathrm{s}(A(x))$$

Notre exemple

```
for i = 1 to n-2 do
  a[i+1] = a[i] + a[i+1]
  a[i] = 0
end for
```

$$R^n \xrightarrow{+_2 \times \mathrm{Id}_{n-2}} R^n \xrightarrow{0 \times \mathrm{Id}_{n-1}} R^n \cdots\cdots\cdots\cdots\cdots\cdots\cdots R^n$$

Notre exemple

```
a[1] = a[0] + a[1]
a[0] = 0
a[2] = a[1] + a[2]
a[1] = 0
...
a[n-1] = a[n-2] + a[n-1]
a[n-2] = 0
```

$$n \quad \mapsto \quad R^n \xrightarrow{+_2 \times \mathrm{Id}_{n-2}} R^n \xrightarrow{0 \times \mathrm{Id}_{n-1}} R^n \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots R^n$$

Le théorème de Tellegen

Foncteur de Tellegen

Un foncteur $F : \mathscr{C} \to \mathscr{D}$ est dit de Tellegen si $F(\mathscr{L}_{\mathscr{C}}) \subset \mathscr{L}_{\mathscr{D}}$.

Théorème de Tellegen

- $F : \mathscr{C} \to \mathscr{D}$ un foncteur de Tellegen
- Par un espace de paramètre
- $A : \mathrm{Par} \to \mathscr{C}_{\to}$ un algorithme

$F \circ A$, noté $F(A)$ est un algorithme de $\mathrm{Par} \to \mathscr{D}_{\to}$ avec

- $\mathrm{t}_{F(A)} = \mathrm{t}_a$,
- $\mathrm{s}_{F(A)} = B(\mathrm{s}_A)$ si $B : \mathbb{N} \to \mathbb{N}$ est une borne sur F.

Le cas $\mathbf{FMod}_R$

On connaît un foncteur (contravariant) $T : \mathbf{FMod}_R \to \mathbf{FMod}_R$ donné par la transposition des matrices.

Théorème de Tellegen pour l'algèbre linéaire

T est un foncteur de Tellegen pour le langage $\mathscr{L}$ donné auparavant.

$$T(+_1) = +_2 \qquad T(*_a) = *_a \qquad T(\pi) = \iota$$

Notre exemple

```
a[1] = a[0] + a[1]
a[0] = 0
a[2] = a[1] + a[2]
a[1] = 0
...
a[n-1] = a[n-2] + a[n-1]
a[n-2] = 0
```

```
a[n-2] = 0
a[n-2] = a[n-2] + a[n-1]
...
a[1] = 0
a[1] = a[1] + a[2]
a[0] = 0
a[0] = a[0] + a[1]
```

Notre exemple

```
a[1] = a[0] + a[1]
a[0] = 0
a[2] = a[1] + a[2]
a[1] = 0
...
a[n-1] = a[n-2] + a[n-1]
a[n-2] = 0
```

```
a[n-2] = 0
a[n-2] = a[n-2] + a[n-1]
...
a[1] = 0
a[1] = a[1] + a[2]
a[0] = 0
a[0] = a[0] + a[1]
```

```
for i = n-2 to 0 do
  a[i] = 0
  a[i] = a[i] + a[i+1]
end for
```

$$\begin{pmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{pmatrix}$$