workshop algorithmique, combinatoire du texte et applications en bio-informatique

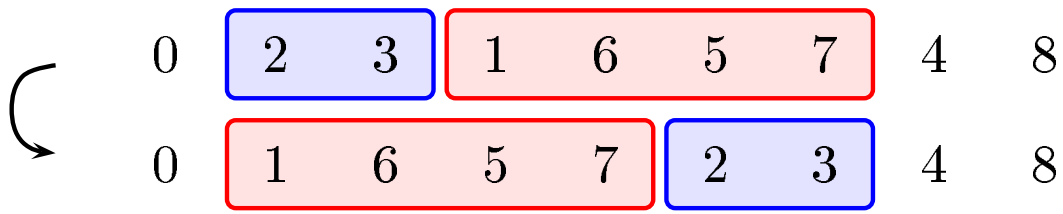
Le tri par transpositions est difficile

Laurent Bulteau, Guillaume Fertin, Irena Rusu
LINA, UMR 6241, Université de Nantes

Lundi 10 Janvier 2011

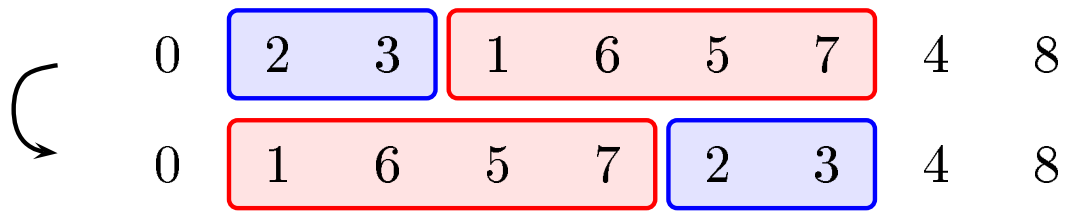
Plan

- Transposition : opération qui échange deux blocs adjacents d'une séquence (de gènes, de marqueurs, etc.)



- Distance de transposition : nombre de transpositions nécessaires pour trier une séquence

- Transposition : opération qui échange deux blocs adjacents d'une séquence (de gènes, de marqueurs, etc.)



- Distance de transposition : nombre de transpositions nécessaires pour trier une séquence

Problème de tri par transpositions :

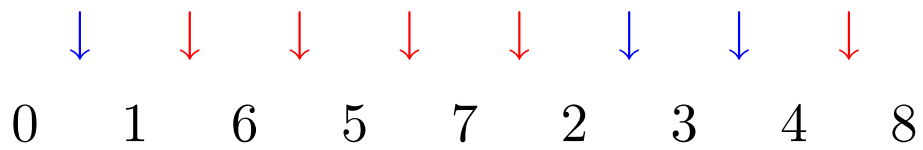
Étant donnée une permutation, calculer sa distance de transposition.

- Adjacence : paire $\pi_i \cdot \pi_{i+1}$ 'bien triée'

$$\pi_{i+1} = \pi_i + 1$$

- Breakpoint : paire $\pi_i \cdot \pi_{i+1}$ n'étant pas une adjacence

$$\pi_{i+1} \neq \pi_i + 1$$



- Distance de breakpoints :

$$d_b(\pi) = \text{nombre de breakpoints de } \pi.$$

Problème de tri par transpositions :

- Défini par Bafna et Pevzner [1998]
- Mailleur ratio d'approximation :

$$1.375$$

(Elias et Hartman [2005])

- Diamètre : inconnu,

$$\text{entre } \left\lfloor \frac{n+1}{2} \right\rfloor \text{ et } \frac{2n}{3}$$

(Bafna et Pevzner [1998], Eriksson et al. [2001])

Problème de tri par transpositions :

- Défini par Bafna et Pevzner [1998]
- Mailleur ratio d'approximation :

$$1.375$$

(Elias et Hartman [2005])

- Diamètre : inconnu,

$$\text{entre } \left\lfloor \frac{n+1}{2} \right\rfloor \text{ et } \frac{2n}{3}$$

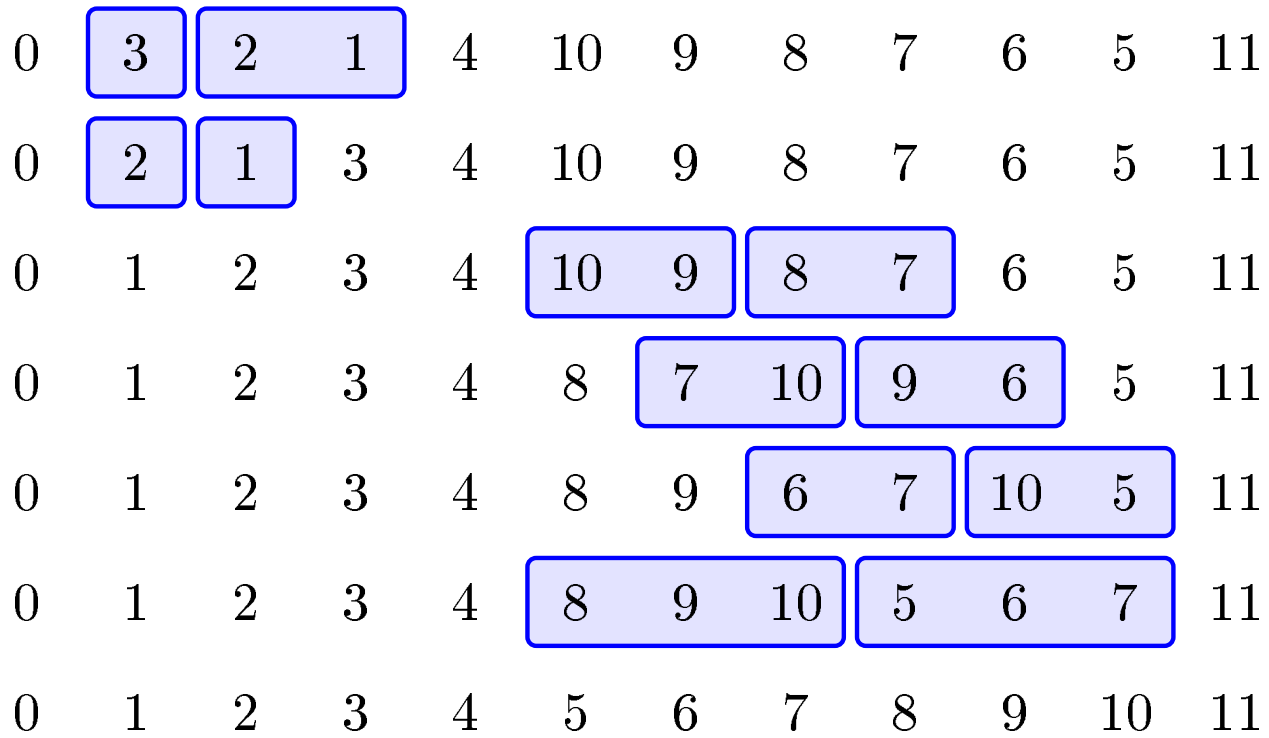
(Bafna et Pevzner [1998], Eriksson et al. [2001])

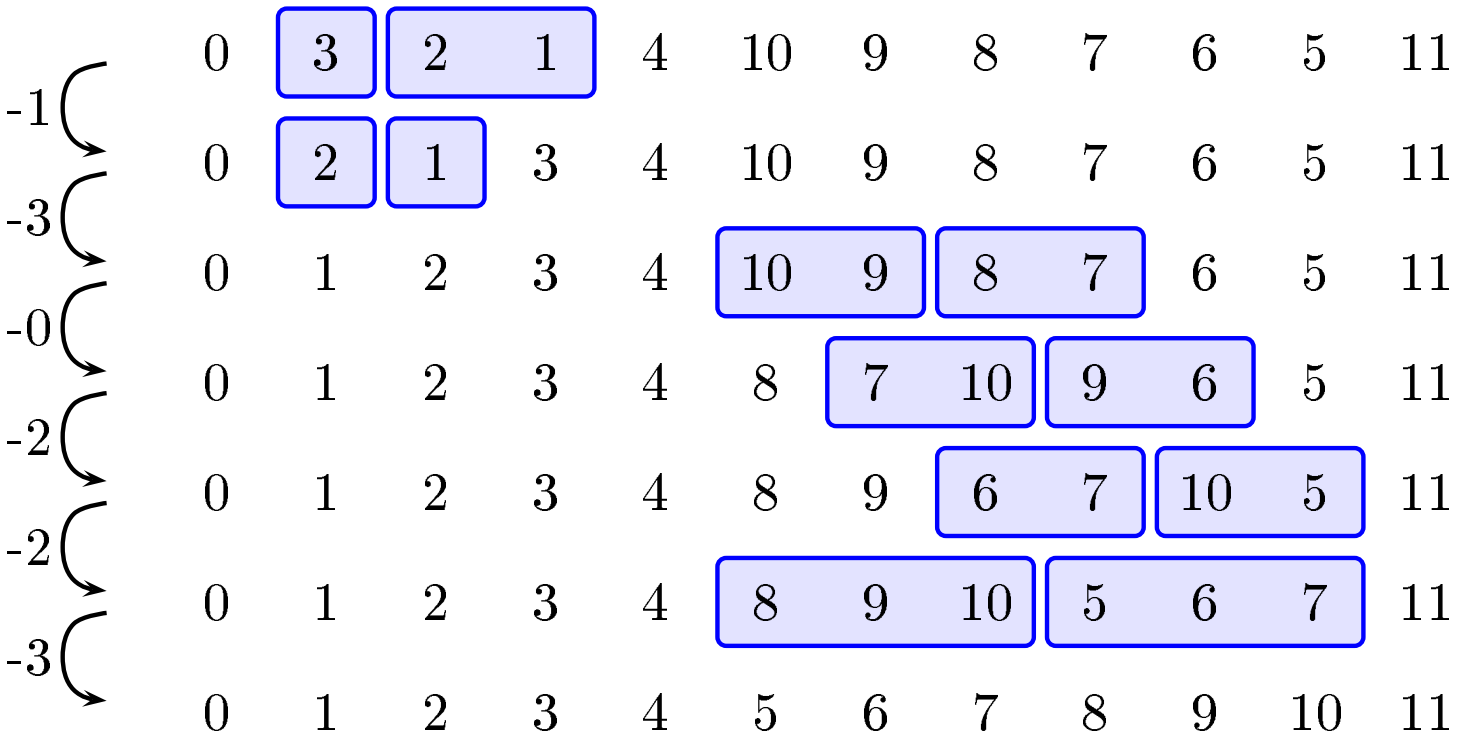
- Complexité : inconnue.

$$0 \quad 3 \quad 2 \quad 1 \quad 4 \quad 10 \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 11$$

$$0 \quad \boxed{3} \quad \boxed{2} \quad \boxed{1} \quad 4 \quad 10 \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 11$$

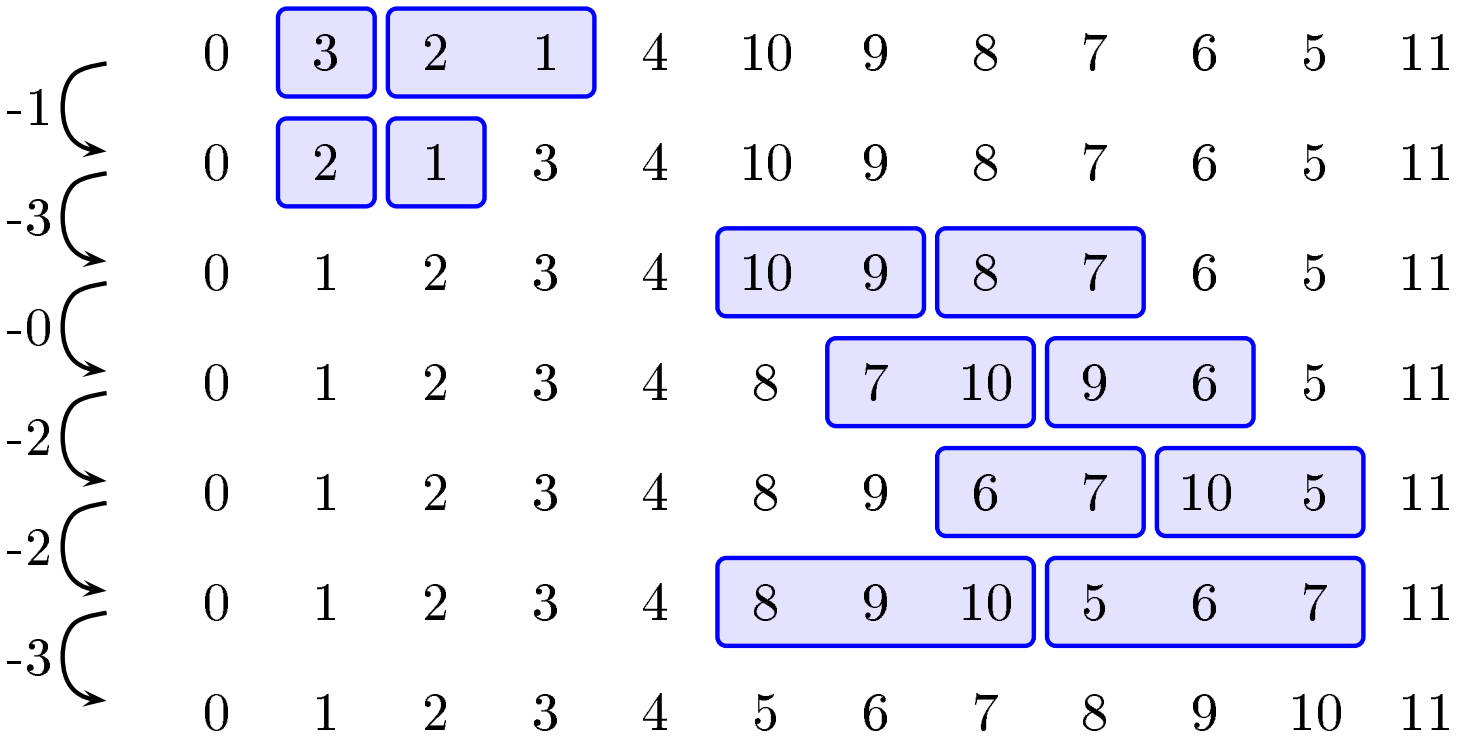
$$0 \quad 2 \quad 1 \quad 3 \quad 4 \quad 10 \quad 9 \quad 8 \quad 7 \quad 6 \quad 5 \quad 11$$


0 3 2 1 4 10 9 8 7 6 5 11
0 2 1 3 4 10 9 8 7 6 5 11
0 1 2 3 4 10 9 8 7 6 5 11
0 1 2 3 4 8 7 10 9 6 5 11
0 1 2 3 4 8 9 6 7 10 5 11
0 1 2 3 4 8 9 10 5 6 7 11
0 1 2 3 4 5 6 7 8 9 10 11



Nombre de breakpoints :

Diminue de 0 à 3 à chaque transposition 'optimale'.

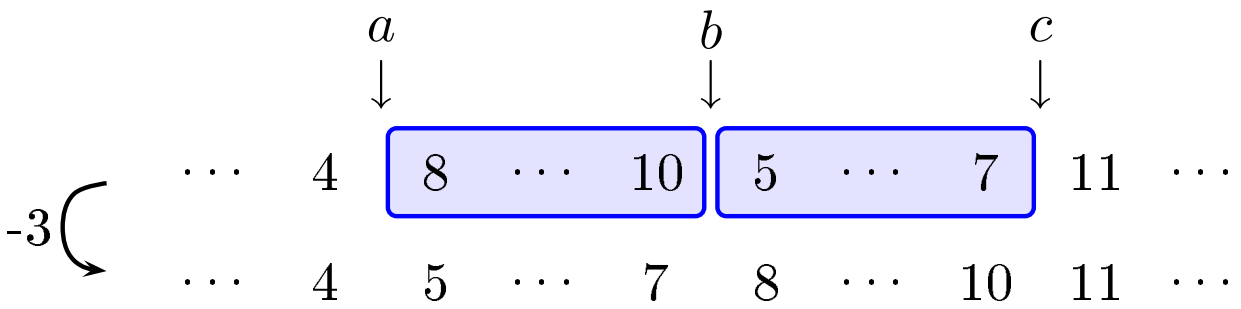


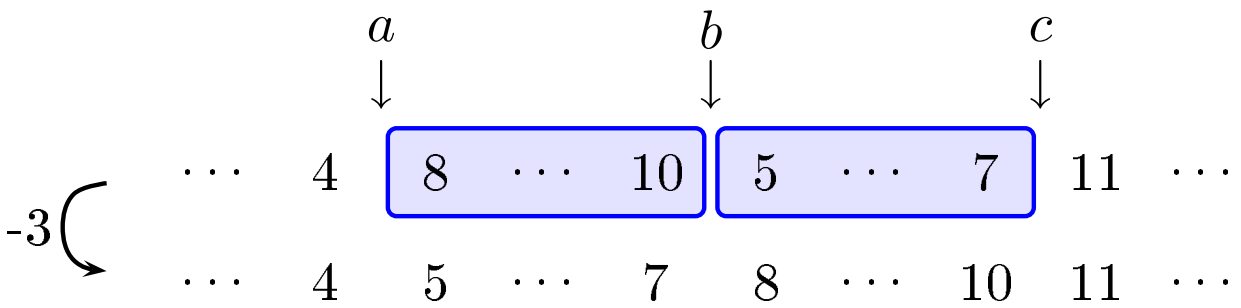
Nombre de breakpoints :

Diminue de 0 à 3 à chaque transposition 'optimale'.

Question :

Existe-t-il une séquence de transpositions retirant chacune 3 breakpoints ?


a
b
c
-3
· · · 4 8 · · · 10 5 · · · 7 11 · · ·
· · · 4 5 · · · 7 8 · · · 10 11 · · ·



Trois breakpoints particuliers : a, b, c

$$\exists \alpha, \beta, \gamma \qquad \begin{aligned} a &= 4 \cdot 8 &&= \alpha \cdot (\gamma+1) \\ b &= 10 \cdot 5 &&= \beta \cdot (\alpha+1) \\ c &= 7 \cdot 11 &&= \gamma \cdot (\beta+1) \end{aligned}$$

Il existe une transposition retirant 3 breakpoints si :

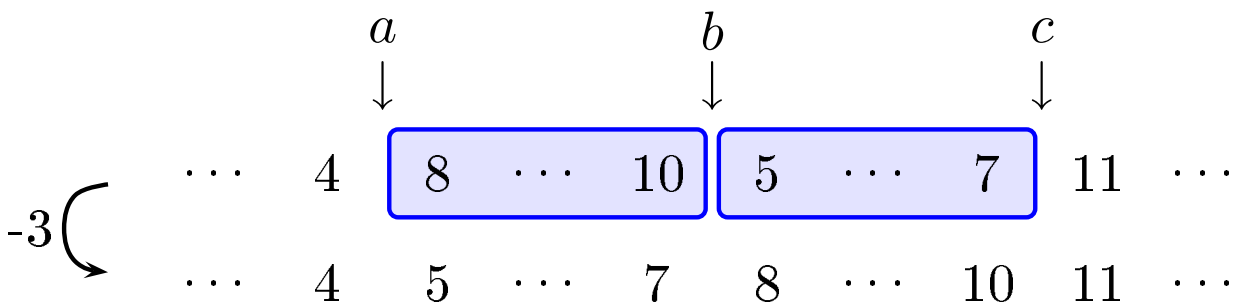
$$a \prec b \prec c$$

($\prec$: ordre d'apparition dans la permutation)
$\Rightarrow$ On note $\tau[a, b, c]$ la transposition correspondante.

$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

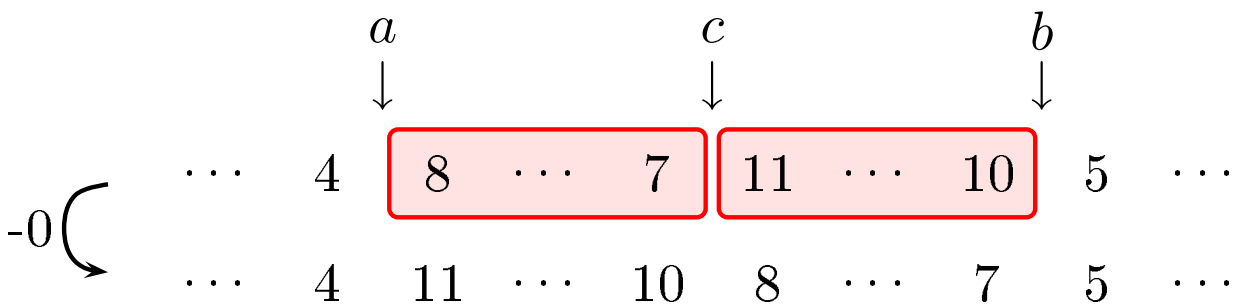
$a \prec b \prec c$ OUI -3 breakpoints



$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

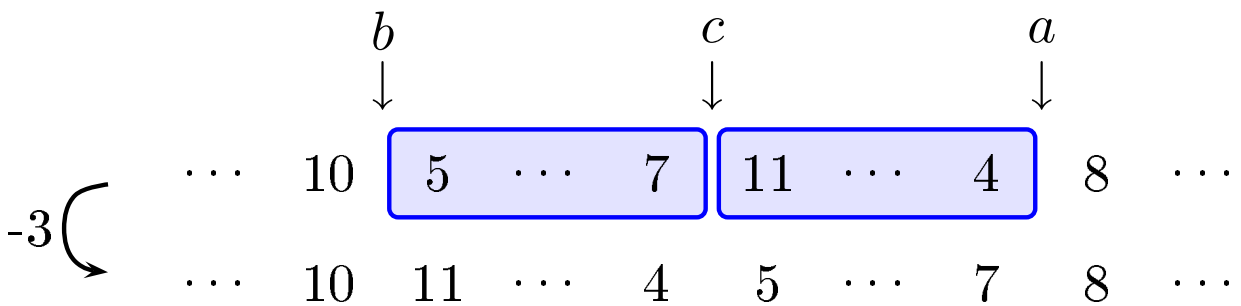
$a \prec b \prec c$	OUI	-3 breakpoints
$a \prec c \prec b$	NON	-0 breakpoints



$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

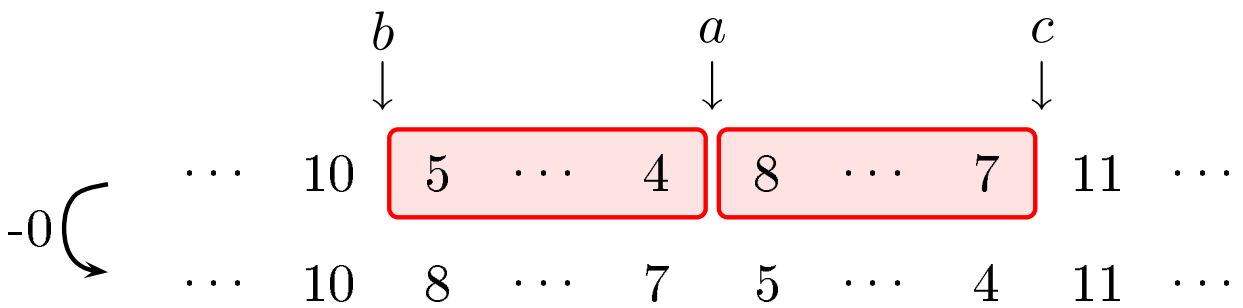
$a \prec b \prec c$	OUI	-3 breakpoints
$a \prec c \prec b$	NON	-0 breakpoints
$b \prec c \prec a$	OUI	-3 breakpoints



$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

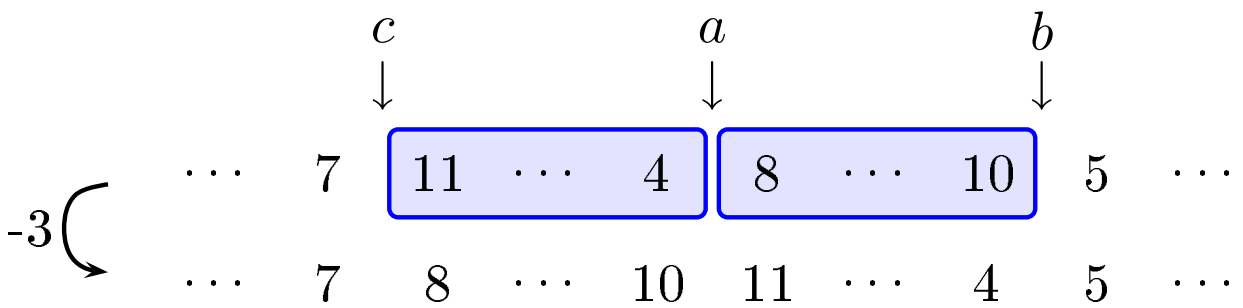
$a \prec b \prec c$	OUI	-3 breakpoints
$a \prec c \prec b$	NON	-0 breakpoints
$b \prec c \prec a$	OUI	-3 breakpoints
$b \prec a \prec c$	NON	-0 breakpoints



$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

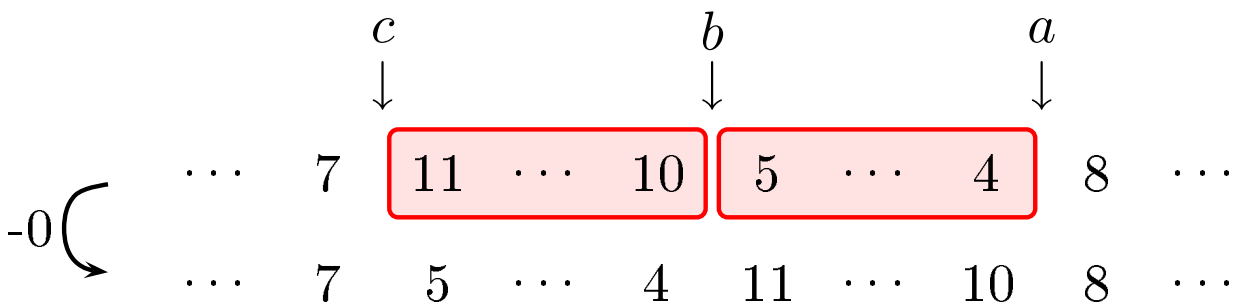
$a \prec b \prec c$	OUI	-3 breakpoints
$a \prec c \prec b$	NON	-0 breakpoints
$b \prec c \prec a$	OUI	-3 breakpoints
$b \prec a \prec c$	NON	-0 breakpoints
$c \prec a \prec b$	OUI	-3 breakpoints



$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

$a \prec b \prec c$	OUI	-3 breakpoints
$a \prec c \prec b$	NON	-0 breakpoints
$b \prec c \prec a$	OUI	-3 breakpoints
$b \prec a \prec c$	NON	-0 breakpoints
$c \prec a \prec b$	OUI	-3 breakpoints
$c \prec b \prec a$	NON	-0 breakpoints



$a = 4 \cdot 8$; $b = 10 \cdot 5$; $c = 7 \cdot 11$

Dans quels cas la transposition $\tau[a, b, c]$ retire-t-elle 3 breakpoints ?

$a \prec b \prec c$	OUI	-3 breakpoints
$a \prec c \prec b$	NON	-0 breakpoints
$b \prec c \prec a$	OUI	-3 breakpoints
$b \prec a \prec c$	NON	-0 breakpoints
$c \prec a \prec b$	OUI	-3 breakpoints
$c \prec b \prec a$	NON	-0 breakpoints

Le triplet (a, b, c) est
bien ordonné si $(a \prec b \prec c)$, $(b \prec c \prec a)$ ou $(c \prec a \prec b)$,
mal ordonné si $(a \prec c \prec b)$, $(b \prec a \prec c)$ ou $(c \prec b \prec a)$.

$3 \times$ *Distance de transposition* $=$ *Nombre de breakpoints*

si et seulement si :

- π a $3k$ breakpoints, pouvant être partitionnés en k triplets (a_i, b_i, c_i)
  Avec $a_i = \alpha_i \cdot (\gamma_i{+}1)$, $b_i = \beta_i \cdot (\alpha_i{+}1)$ *et* $c_i = \gamma_i \cdot (\beta_i{+}1)$.
  On note $\tau_i = \tau[a_i, b_i, c_i]$
- Il existe un ordre sur les triplets : $(i_1, i_2, \ldots, i_k)$ tel que $\forall j$, $(a_{i_j}, b_{i_j}, c_{i_j})$ soit bien ordonné dans la permutation

$$\pi \circ \tau_{i_1} \circ \tau_{i_2} \circ \ldots \circ \tau_{i_{j-1}}$$

$3 \times$ *Distance de transposition* $=$ *Nombre de breakpoints*

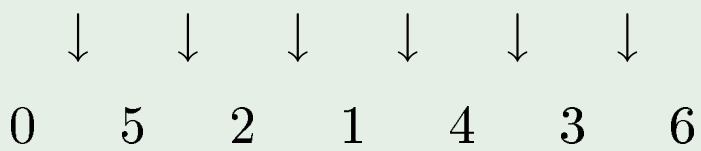
si et seulement si :

- π a $3k$ breakpoints, pouvant être partitionnés en k triplets (a_i, b_i, c_i)
  Avec $a_i = \alpha_i \cdot (\gamma_i{+}1)$, $b_i = \beta_i \cdot (\alpha_i{+}1)$ *et* $c_i = \gamma_i \cdot (\beta_i{+}1)$.
  On note $\tau_i = \tau[a_i, b_i, c_i]$
- Il existe un ordre sur les triplets : $(i_1, i_2, \ldots, i_k)$ tel que $\forall j$, $(a_{i_j}, b_{i_j}, c_{i_j})$ soit bien ordonné dans la permutation

$$\pi \circ \tau_{i_1} \circ \tau_{i_2} \circ \ldots \circ \tau_{i_{j-1}}$$

$\Rightarrow$ Alors $\pi \circ \tau_{i_1} \circ \tau_{i_2} \circ \ldots \circ \tau_{i_k}$ est l'identité (on a retiré tous les breakpoints) : la distance de transposition de π est au plus, donc exactement, k.

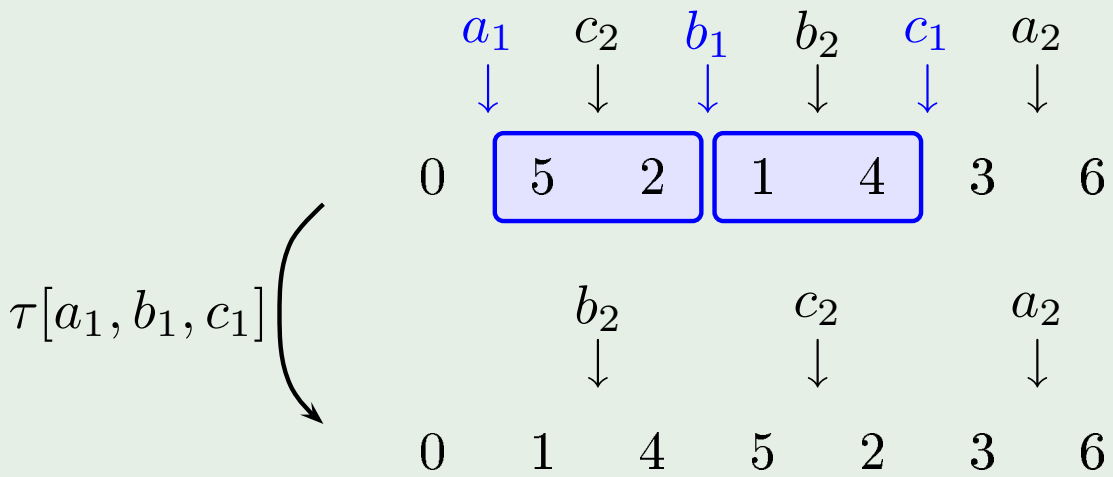
Exemple : $\pi = (0\ 5\ 2\ 1\ 4\ 3\ 6)$



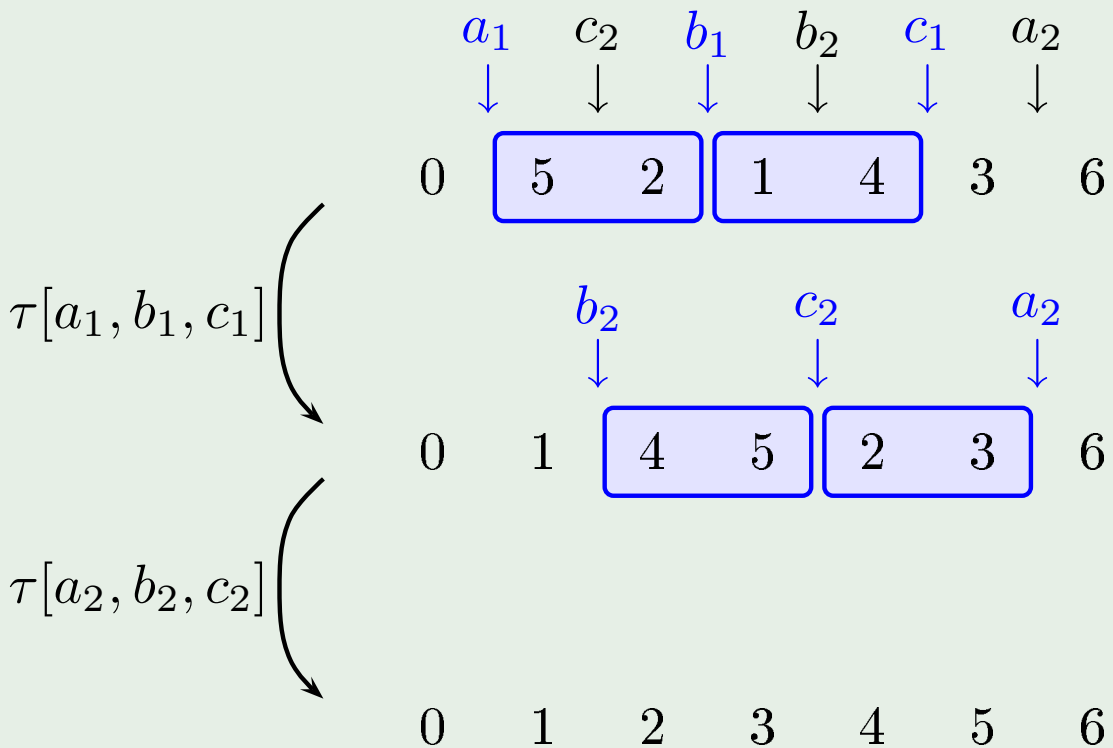
Exemple : $\pi = (0\ 5\ 2\ 1\ 4\ 3\ 6)$

$$\begin{array}{cccccccc} & a_1 & c_2 & b_1 & b_2 & c_1 & a_2 & \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ 0 & 5 & 2 & 1 & 4 & 3 & 6 & \end{array}$$

Exemple : $\pi = (0\ 5\ 2\ 1\ 4\ 3\ 6)$



Exemple : $\pi = (0\ 5\ 2\ 1\ 4\ 3\ 6)$



Étant donnée une permutation π

- Partitionner les breakpoints en triplets est rapide
- Déterminer s'il existe un bon ordre sur ces triplets est difficile : *aussi difficile que de déterminer si une formule booléenne est satisifiable.*

Étant donnée une permutation π

- Partitionner les breakpoints en triplets est rapide
- Déterminer s'il existe un bon ordre sur ces triplets est difficile : *aussi difficile que de déterminer si une formule booléenne est satisifiable.*

A partir d'une formule booléenne ϕ, on construit une permutation π, donnée sous forme d'une séquence de breakpoints partitionnés en triplets.

Variables : $A = [(a, b, c), (x, y, z)]$, utilise deux triplets.

- État de base :

$$y \prec x \prec b \quad \text{et} \quad a \prec z \prec c$$

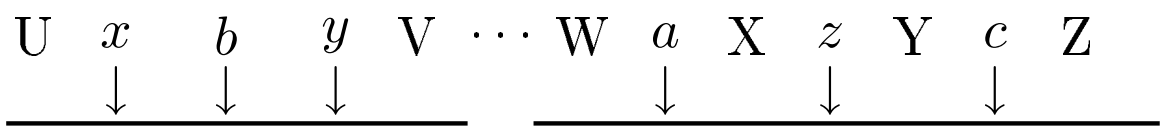
Avec $b \prec a$ ou $c \prec y$, donc (x, y, z) et (a, b, c) sont mal ordonnés.

- État activable :

$$x \prec b \prec y \quad \text{et} \quad a \prec z \prec c$$

Avec un seul breakpoint entre x et y : b.

- Activation :



Variables : $A = [(a, b, c), (x, y, z)]$, utilise deux triplets.

- État de base :

$$y \prec x \prec b \quad \text{et} \quad a \prec z \prec c$$

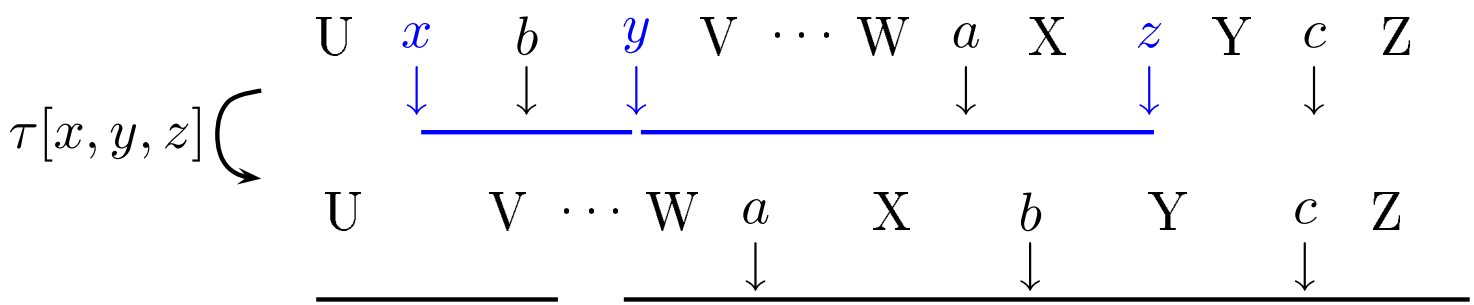
Avec $b \prec a$ ou $c \prec y$, donc (x, y, z) et (a, b, c) sont mal ordonnés.

- État activable :

$$x \prec b \prec y \quad \text{et} \quad a \prec z \prec c$$

Avec un seul breakpoint entre x et y : b.

- Activation :



Variables : $A = [(a, b, c), (x, y, z)]$, utilise deux triplets.

- État de base :

$$y \prec x \prec b \quad \text{et} \quad a \prec z \prec c$$

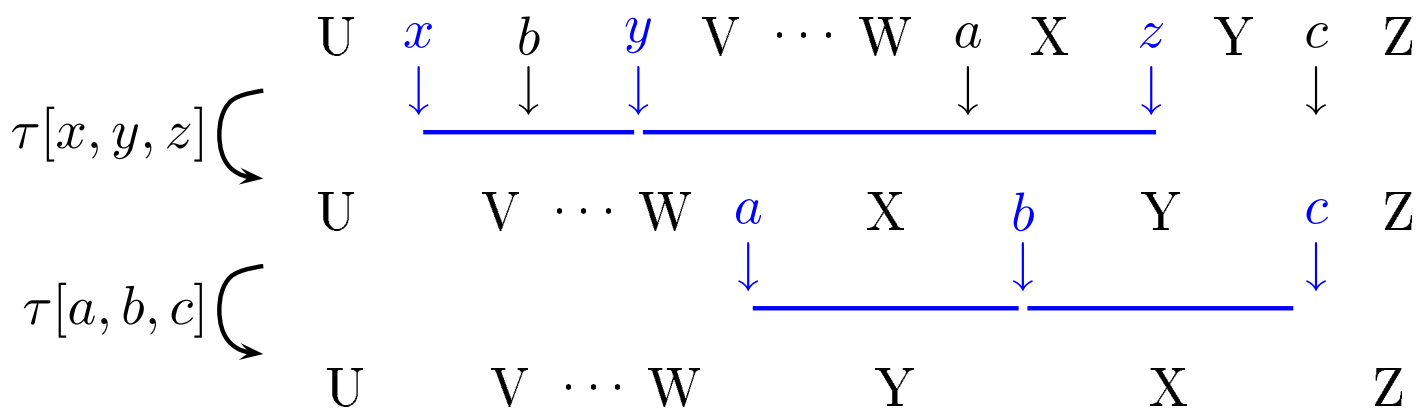
Avec $b \prec a$ ou $c \prec y$, donc (x, y, z) et (a, b, c) sont mal ordonnés.

- État activable :

$$x \prec b \prec y \quad \text{et} \quad a \prec z \prec c$$

Avec un seul breakpoint entre x et y : b.

- Activation :



Variables : $A = [(a, b, c), (x, y, z)]$, utilise deux triplets.

- État de base :

$$y \prec x \prec b \quad \text{et} \quad a \prec z \prec c$$

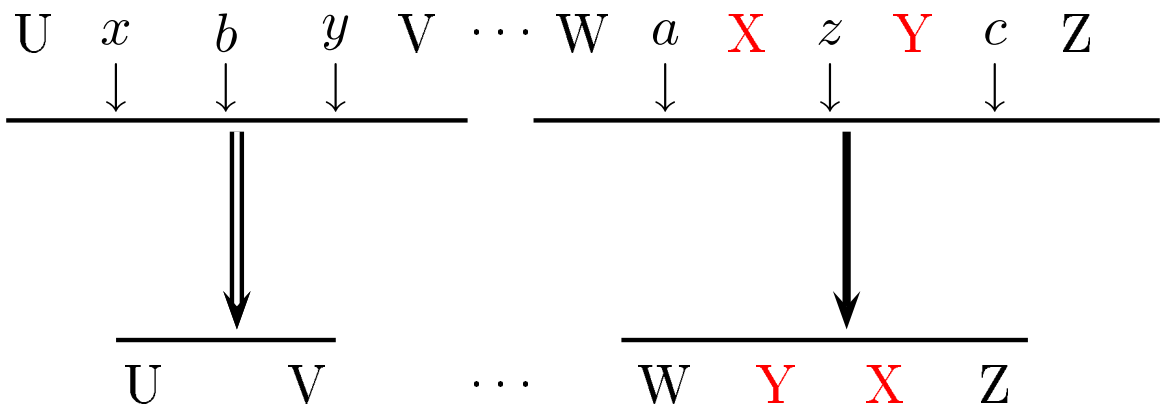
Avec $b \prec a$ ou $c \prec y$, donc (x, y, z) et (a, b, c) sont mal ordonnés.

- État activable :

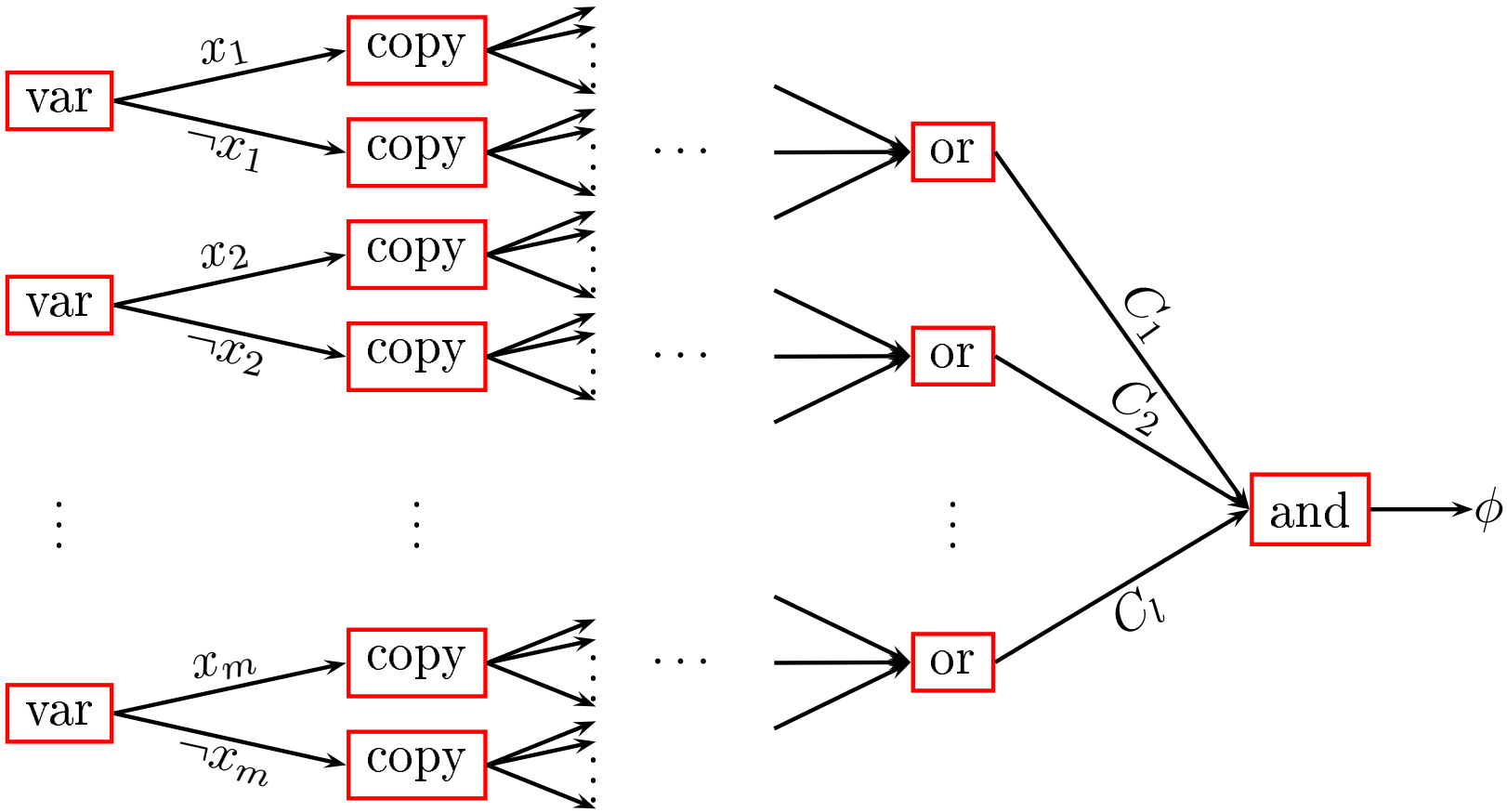
$$x \prec b \prec y \quad \text{et} \quad a \prec z \prec c$$

Avec un seul breakpoint entre x et y : b.

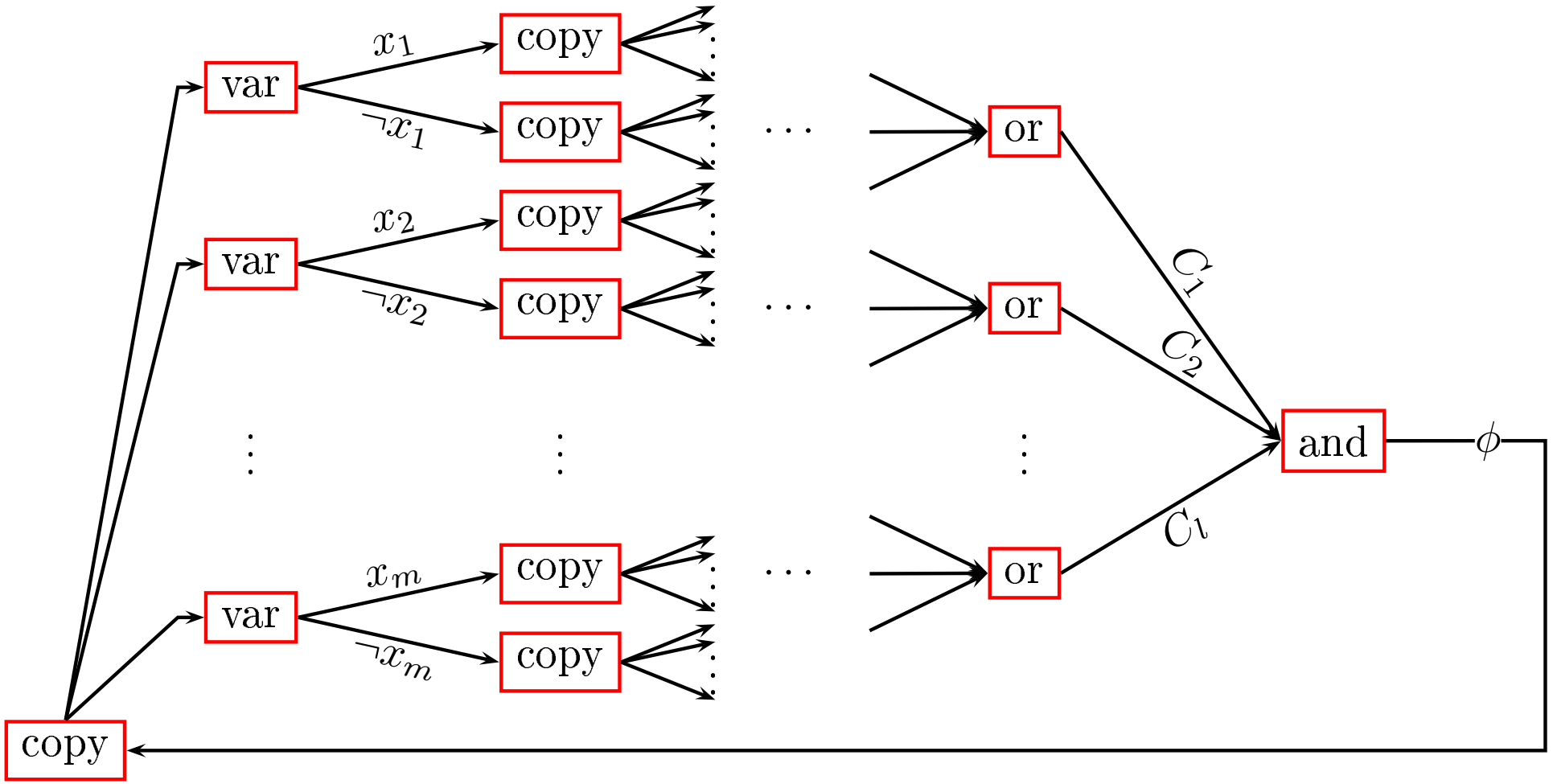
- Activation :



Entrée : formule $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_l$, chaque clause utilise 3 litéraux parmis $\{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_m, \neg x_m\}$.



Entrée : formule $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_l$, chaque clause utilise 3 litéraux parmis $\{x_1, \neg x_1, x_2, \neg x_2, \ldots, x_m, \neg x_m\}$.



La brique **and**

Variables d'entrée : A_1
A_2
Variable de sortie : A

$$A = \mathbf{and}(A_1, A_2)$$

La brique **and**

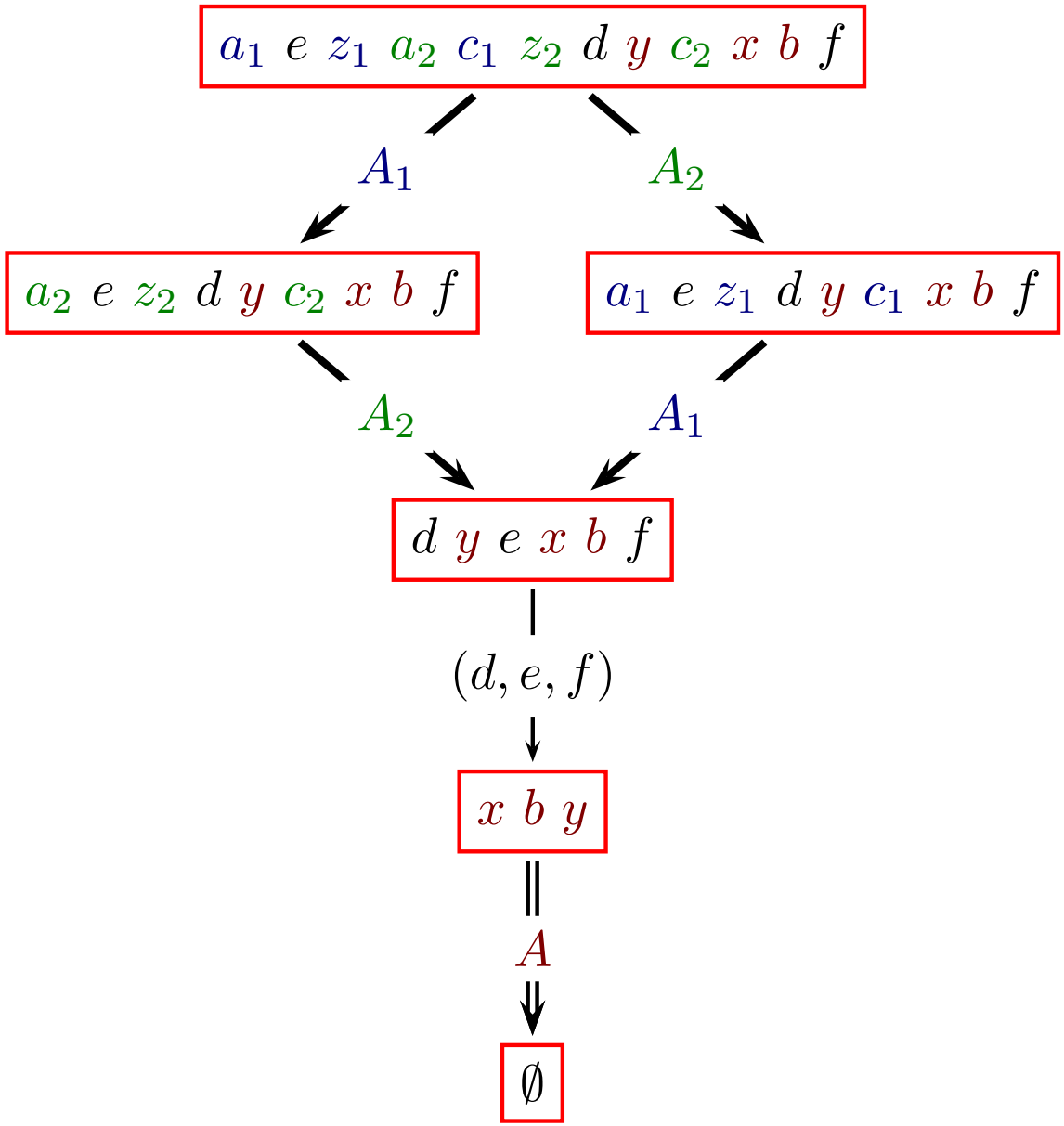
Variables d'entrée : $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$
$A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$

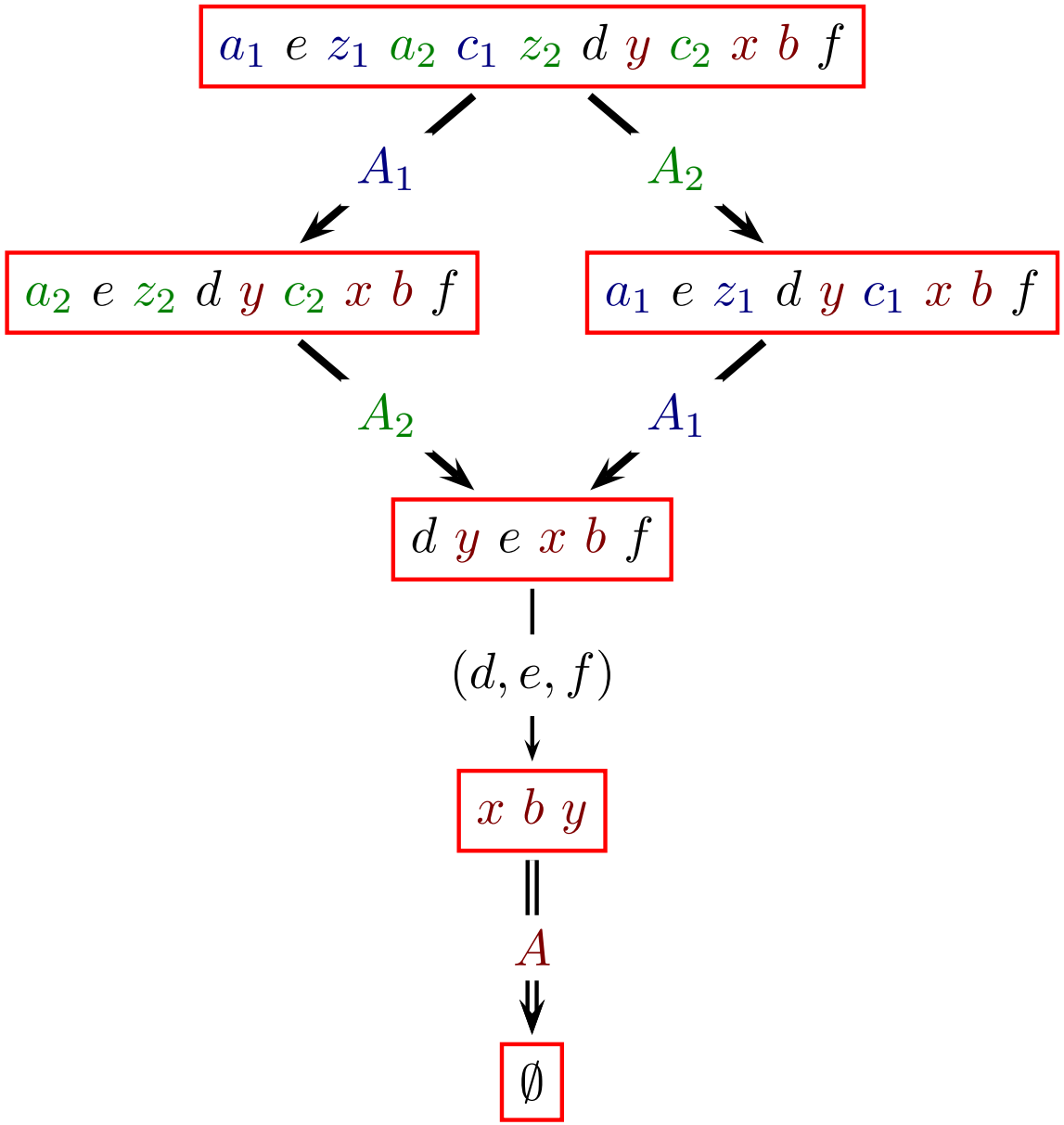
Variable de sortie : $A = [(a, b, c), (x, y, z)]$

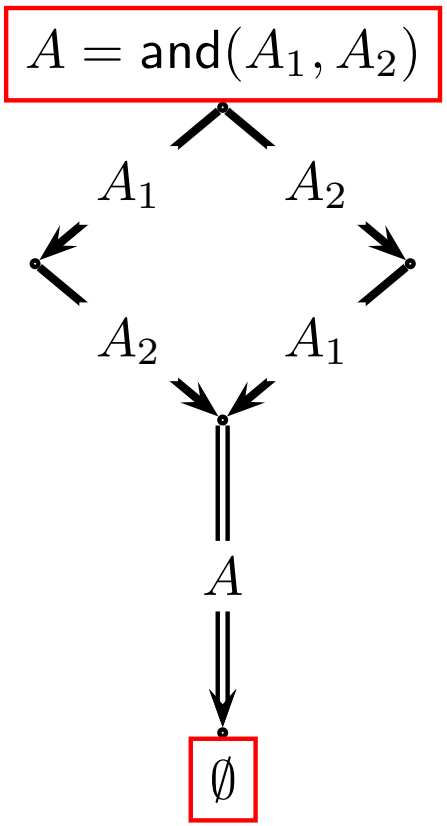
Triplet 'interne' : (d, e, f)

$$A = \mathbf{and}(A_1, A_2)$$

$a_1\ e\ z_1\ a_2\ c_1\ z_2\ d\ y\ c_2\ x\ b\ f$


$a_1\ e\ z_1\ a_2\ c_1\ z_2\ d\ y\ c_2\ x\ b\ f$
A_1
A_2
$a_2\ e\ z_2\ d\ y\ c_2\ x\ b\ f$
$a_1\ e\ z_1\ d\ y\ c_1\ x\ b\ f$
A_2
A_1
$d\ y\ e\ x\ b\ f$
(d, e, f)
$x\ b\ y$
A
$\emptyset$

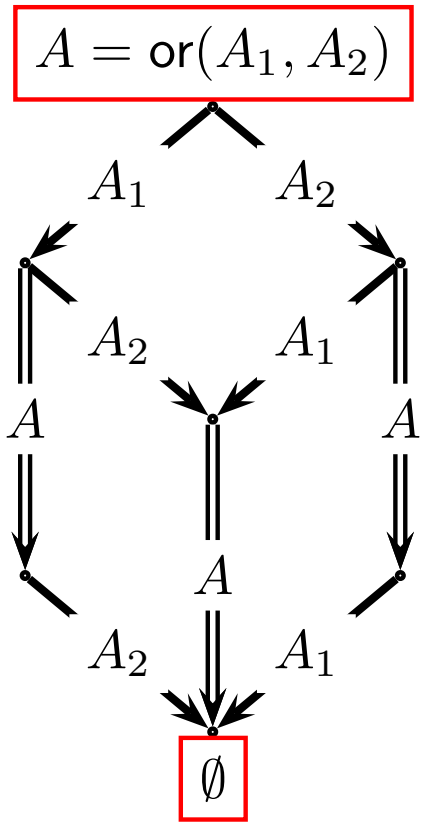

$a_1\ e\ z_1\ a_2\ c_1\ z_2\ d\ y\ c_2\ x\ b\ f$
A_1
A_2
$a_2\ e\ z_2\ d\ y\ c_2\ x\ b\ f$
$a_1\ e\ z_1\ d\ y\ c_1\ x\ b\ f$
A_2
A_1
$d\ y\ e\ x\ b\ f$
(d, e, f)
$x\ b\ y$
A
$\emptyset$


$A = \mathsf{and}(A_1, A_2)$
A_1
A_2
A_2
A_1
A
$\emptyset$

La brique **or**

Variables d'entrée : A_1
A_2
Variable de sortie : A

$$A = \mathbf{or}(A_1, A_2)$$

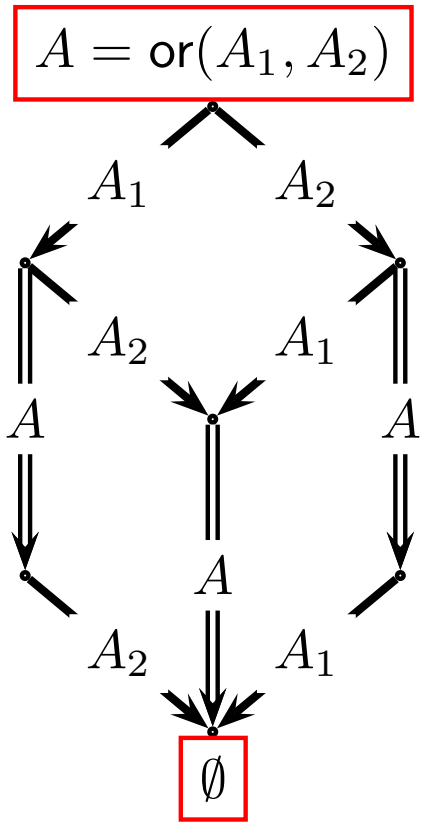


La brique **or**

Variables d'entrée : $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$
$A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$
Variable de sortie : $A = [(a, b, c), (x, y, z)]$
Triplets internes : $(a', b', c'), (d, e, f)$

$$A = \mathbf{or}(A_1, A_2)$$

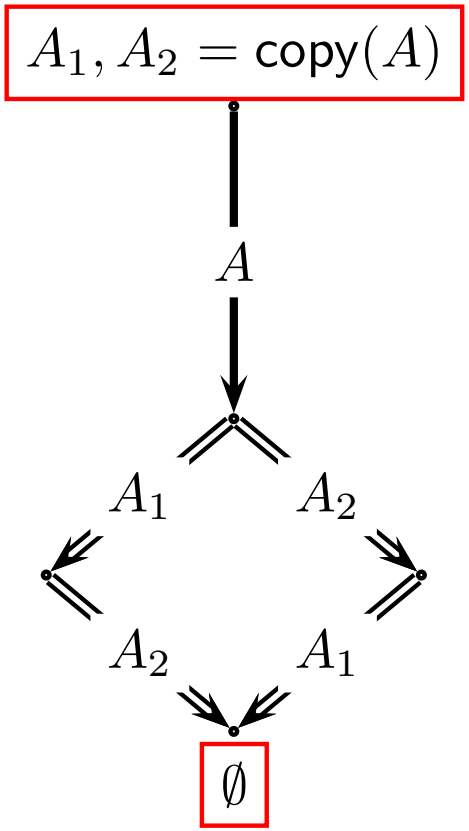
$a_1\ b'\ z_1\ a_2\ d\ y\ a'\ x\ b\ f\ z_2\ c_1\ e\ c'\ c_2$



La brique **copy**

Variable d'entrée : A
Variables de sortie : A_1
A_2

$$A_1, A_2 = \textbf{copy}(A)$$

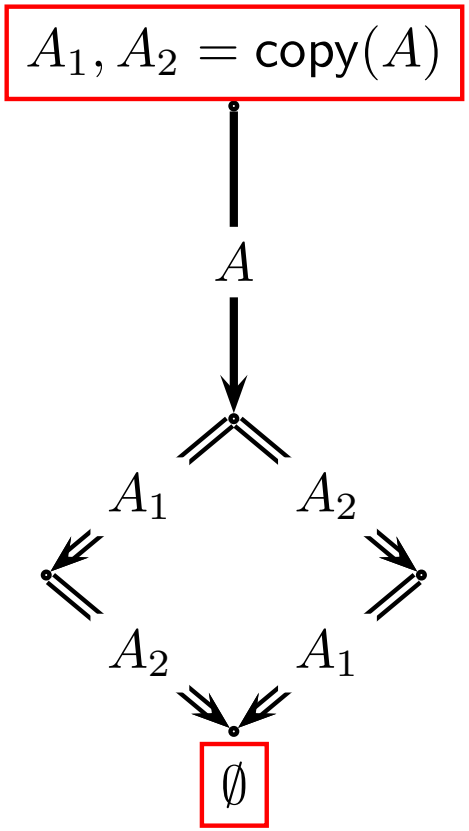


La brique **copy**

Variable d'entrée : $A = [(a, b, c), (x, y, z)]$
Variables de sortie : $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$
$A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$
Triplet interne : (d, e, f)

$$A_1, A_2 = \textbf{copy}(A)$$

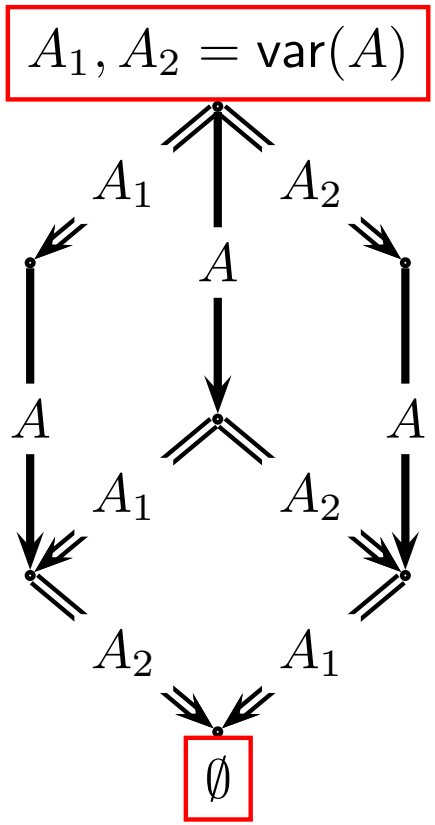
$a\ y_1\ e\ z\ d\ y_2\ x_1\ b_1\ c\ x_2\ b_2\ f$



La brique **var**

Variable d'entrée : A
Variables de sortie : A_1
A_2

$$A_1, A_2 = \mathbf{var}(A)$$



La brique **var**

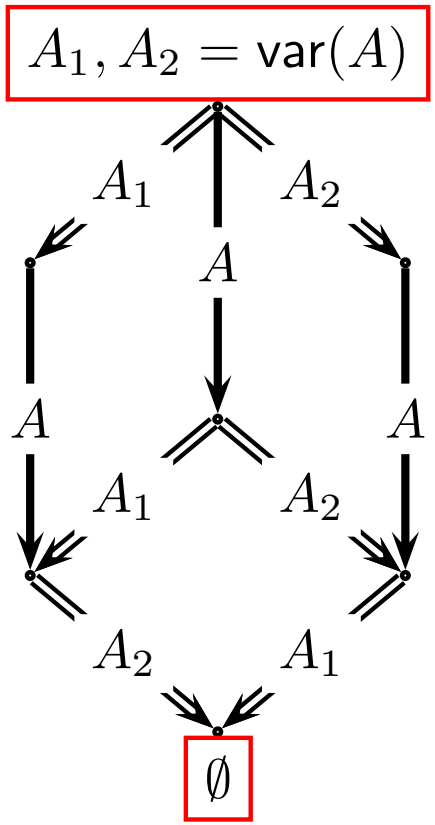
Variable d'entrée : $A = [(a, b, c), (x, y, z)]$

Variables de sortie : $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$

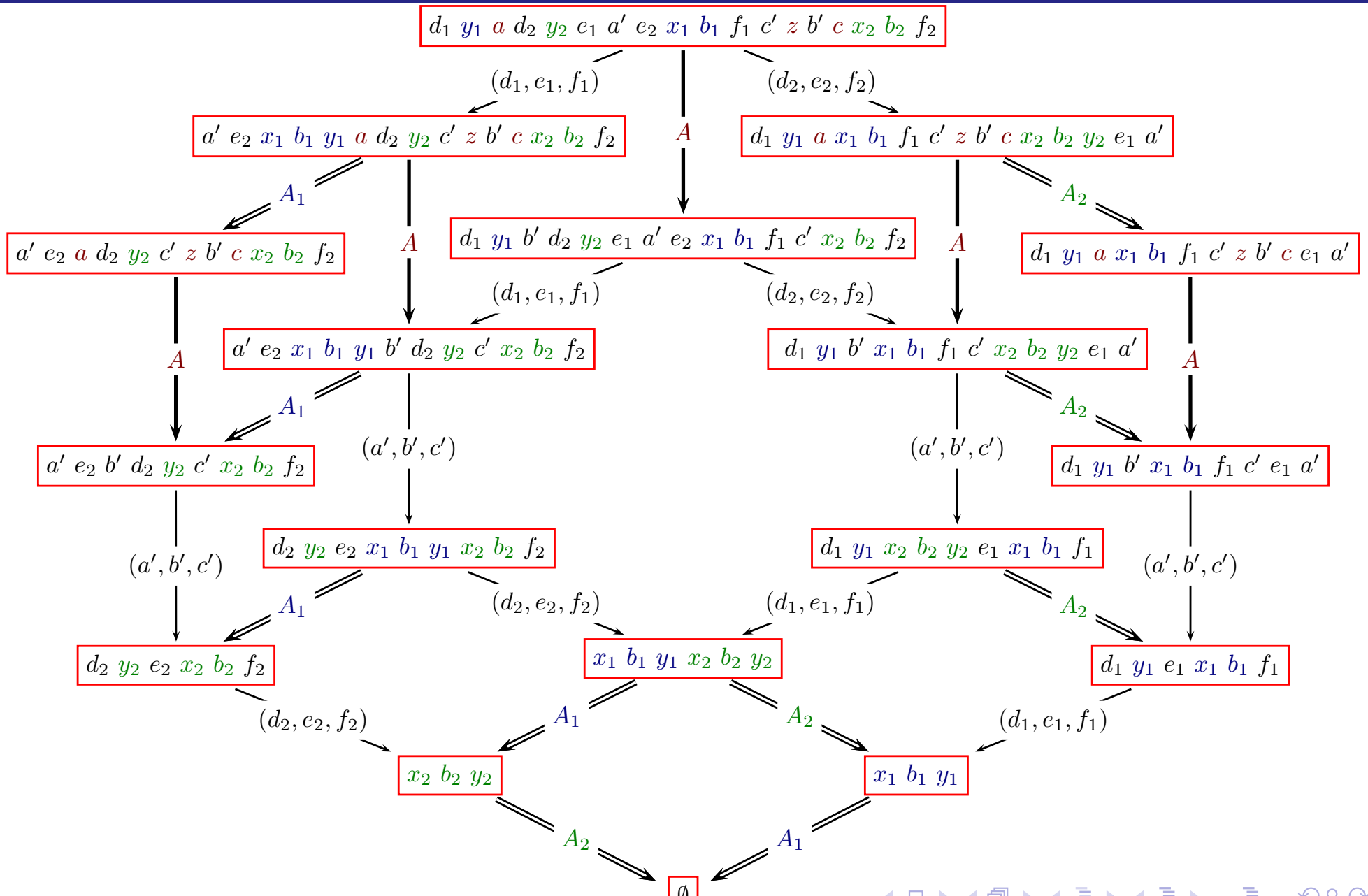
$A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$

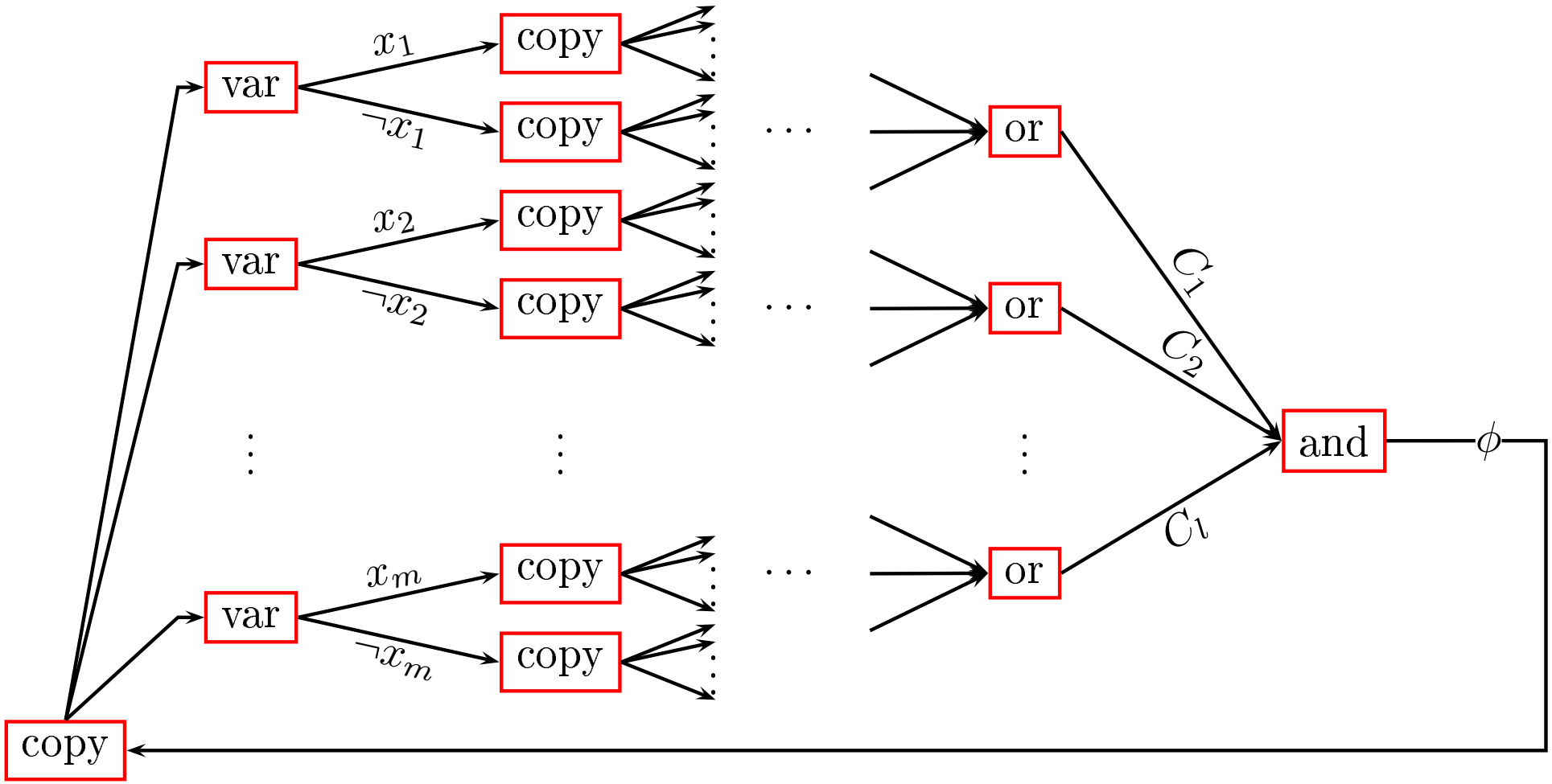
Triplets internes : (d_1, e_1, f_1),
(d_2, e_2, f_2),
(a', b', c').

$$A_1, A_2 = \mathbf{var}(A)$$



d_1 y_1 a d_2 y_2 e_1 a' e_2 x_1 b_1 f_1 c' z b' c x_2 b_2 f_2


$d_1\ y_1\ a\ d_2\ y_2\ e_1\ a'\ e_2\ x_1\ b_1\ f_1\ c'\ z\ b'\ c\ x_2\ b_2\ f_2$
(d_1, e_1, f_1)
A
(d_2, e_2, f_2)
$a'\ e_2\ x_1\ b_1\ y_1\ a\ d_2\ y_2\ c'\ z\ b'\ c\ x_2\ b_2\ f_2$
$d_1\ y_1\ a\ x_1\ b_1\ f_1\ c'\ z\ b'\ c\ x_2\ b_2\ y_2\ e_1\ a'$
A_1
A_2
$a'\ e_2\ a\ d_2\ y_2\ c'\ z\ b'\ c\ x_2\ b_2\ f_2$
$d_1\ y_1\ b'\ d_2\ y_2\ e_1\ a'\ e_2\ x_1\ b_1\ f_1\ c'\ x_2\ b_2\ f_2$
$d_1\ y_1\ a\ x_1\ b_1\ f_1\ c'\ z\ b'\ c\ e_1\ a'$
$a'\ e_2\ x_1\ b_1\ y_1\ b'\ d_2\ y_2\ c'\ x_2\ b_2\ f_2$
$d_1\ y_1\ b'\ x_1\ b_1\ f_1\ c'\ x_2\ b_2\ y_2\ e_1\ a'$
$a'\ e_2\ b'\ d_2\ y_2\ c'\ x_2\ b_2\ f_2$
(a', b', c')
$d_1\ y_1\ b'\ x_1\ b_1\ f_1\ c'\ e_1\ a'$
$d_2\ y_2\ e_2\ x_1\ b_1\ y_1\ x_2\ b_2\ f_2$
$d_1\ y_1\ x_2\ b_2\ y_2\ e_1\ x_1\ b_1\ f_1$
$d_2\ y_2\ e_2\ x_2\ b_2\ f_2$
$x_1\ b_1\ y_1\ x_2\ b_2\ y_2$
$d_1\ y_1\ e_1\ x_1\ b_1\ f_1$
$x_2\ b_2\ y_2$
$x_1\ b_1\ y_1$
$\emptyset$


copy
var
x_1
$\neg x_1$
copy
copy
var
x_2
$\neg x_2$
copy
copy
var
x_m
$\neg x_m$
copy
copy
or
or
or
C_1
C_2
C_l
and
ϕ

Conclusion

À partir d'une formule ϕ, on construit une permutation π_ϕ telle que :

- π_ϕ a $3k$ breakpoints ;
- π_ϕ a pour distance de transposition k ssi ϕ est satisfiable.

Conclusion

À partir d'une formule ϕ, on construit une permutation π_ϕ telle que :

- π_ϕ a $3k$ breakpoints ;
- π_ϕ a pour distance de transposition k ssi ϕ est satisfiable.

Donc :

- Le problème de tri par transpositions est NP-difficile
- Déterminer si la borne $dt(\pi) \geq db(\pi)/3$ est atteinte est NP-difficile

Questions toujours ouvertes

- Diamètre de S_n pour la distance de transposition ?
- Schéma d'approximation possible ?
- Algorithme paramétré possible ?