Streaming String Transducers

Jérémy Ledent

ENS de Lyon
Département d'Informatique
L3 – Soutenance de stage

2 septembre 2013

Introduction

Les langages réguliers peuvent être caractérisés par :

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)
- L'algèbre : monoïdes syntaxiques

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)
- L'algèbre : monoïdes syntaxiques
- Et d'autres...

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)
- L'algèbre : monoïdes syntaxiques
- Et d'autres...

Transduction

- Relation $\subseteq \Sigma^* \times \Gamma^*$
- Cas déterministe : fonction partielle $\Sigma^* \to \Gamma^*$

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)
- L'algèbre : monoïdes syntaxiques
- Et d'autres...

Transduction

- Relation $\subseteq \Sigma^* \times \Gamma^*$
- Cas déterministe : fonction partielle $\Sigma^* \to \Gamma^*$

Exemple : $f(w) = w.w^R$

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)
- L'algèbre : monoïdes syntaxiques
- Et d'autres...

Transduction

- Relation $\subseteq \Sigma^* \times \Gamma^*$
- Cas déterministe : fonction partielle $\Sigma^* \to \Gamma^*$

Exemple : $f(w) = w.w^R$

- Peu d'égalités entre les classes de transductions

Introduction

Les langages réguliers peuvent être caractérisés par :

- Les automates finis et variantes (non-déterministes, two-way, ...)
- La logique : logique monadique du second ordre (MSO)
- L'algèbre : monoïdes syntaxiques
- Et d'autres...

Transduction

- Relation $\subseteq \Sigma^* \times \Gamma^*$
- Cas déterministe : fonction partielle $\Sigma^* \to \Gamma^*$

Exemple : $f(w) = w.w^R$

- Peu d'égalités entre les classes de transductions
- Transductions rationnelles : $2\text{DGSM} = \text{DMSOS}$ (Engelfriet & Hoogeboom)

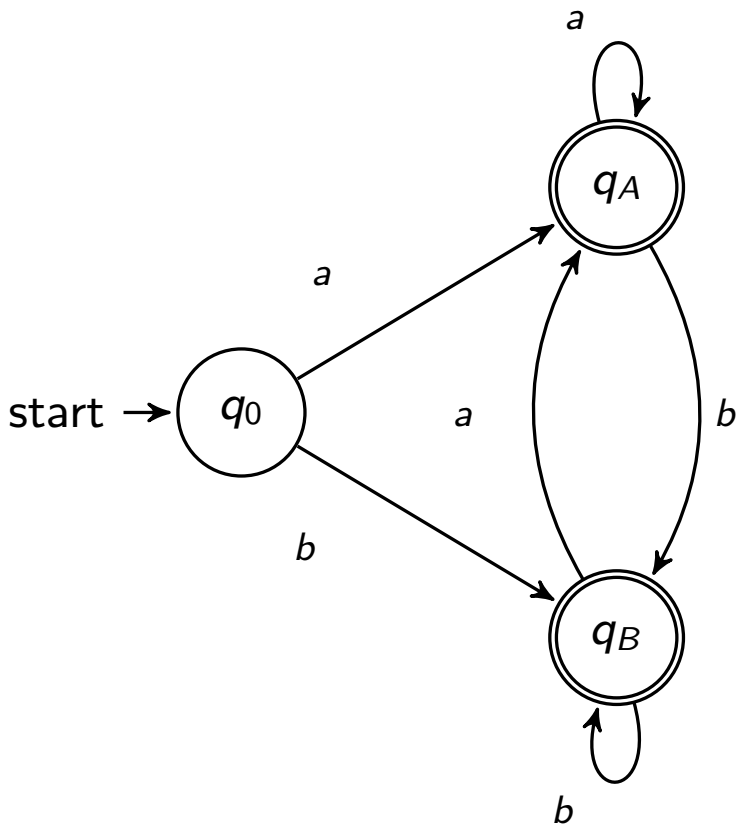
Plan

Plan

Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

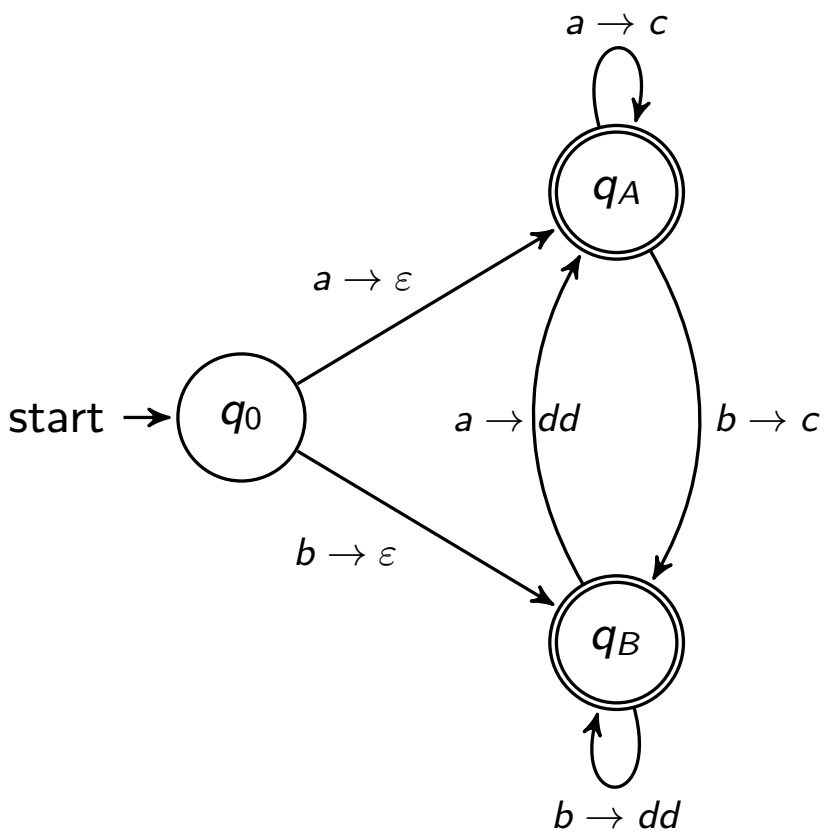
- Similaire à un automate fini



Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

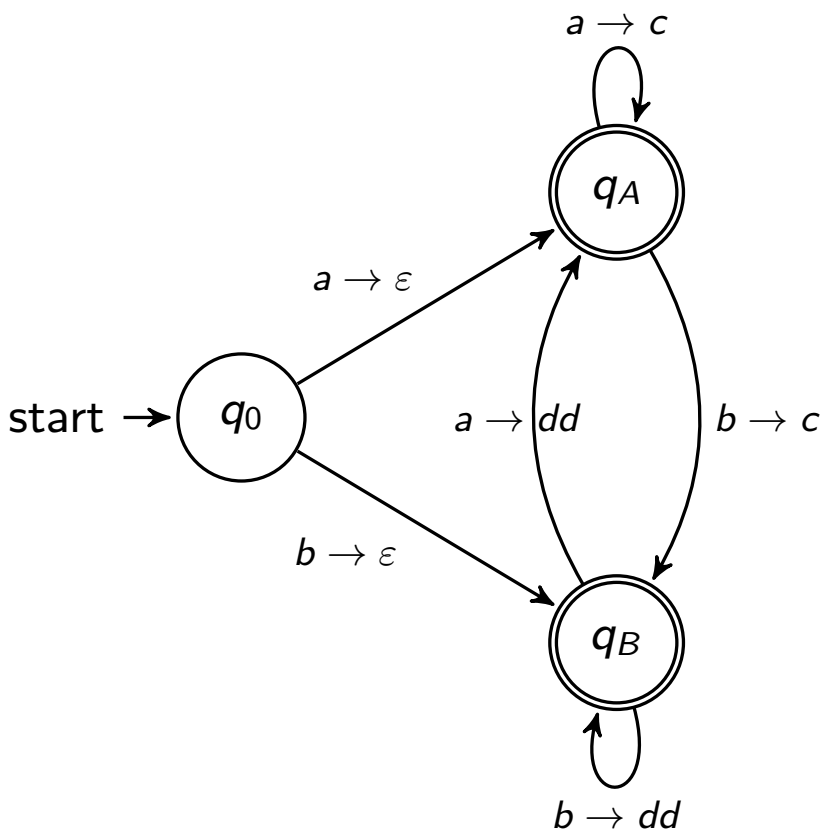
- Similaire à un automate fini
- Calcule un output



Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

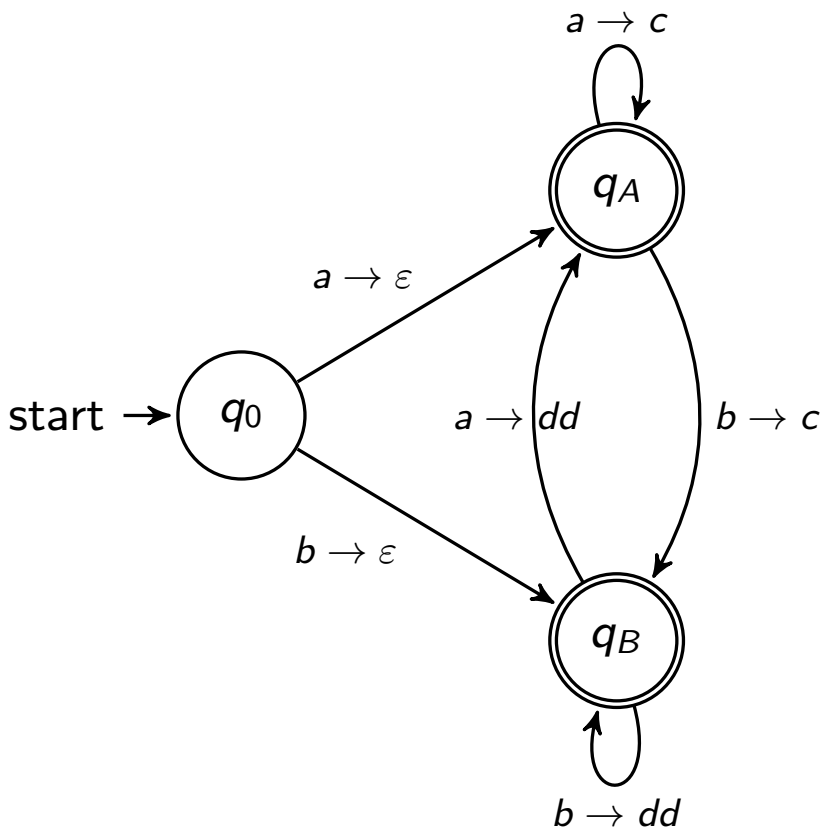
- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$



Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

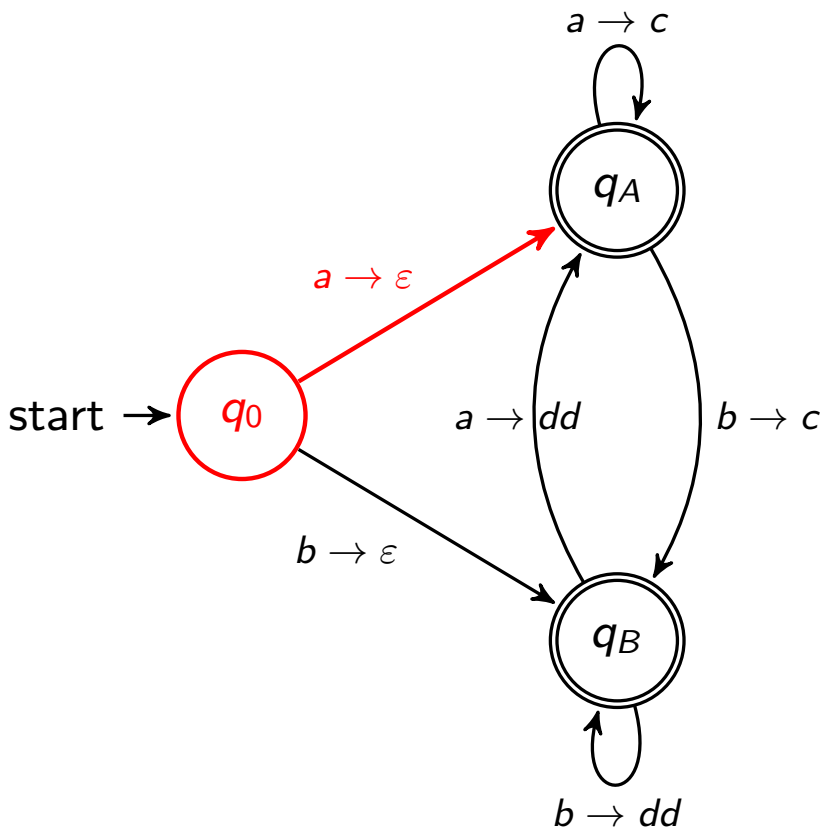


Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

Exemple : $aabab \rightarrow$

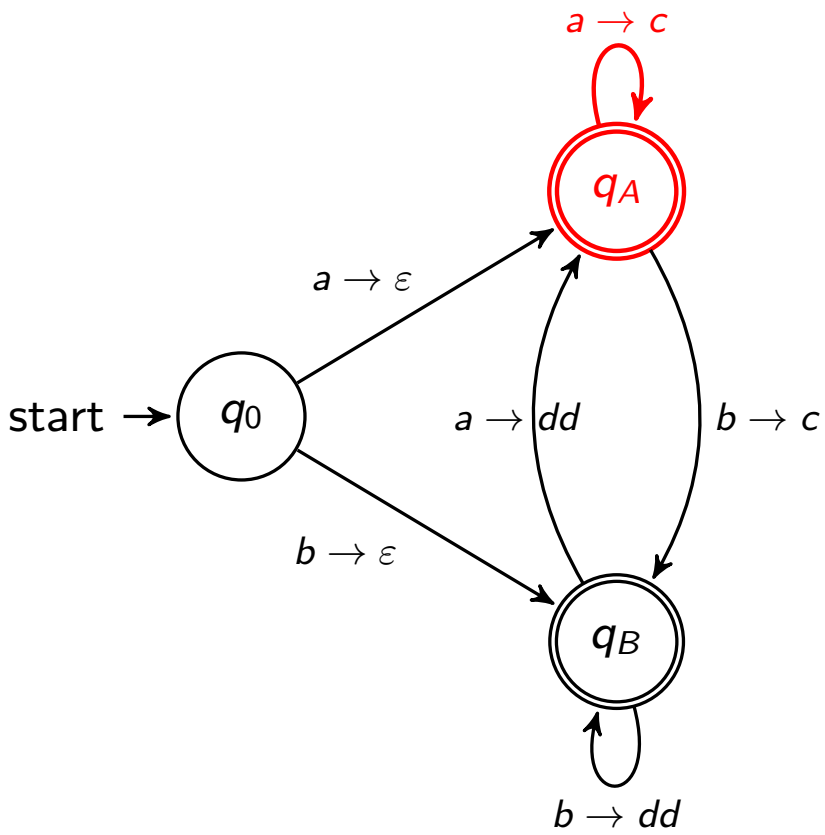


Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

Exemple : $aabab \to c$

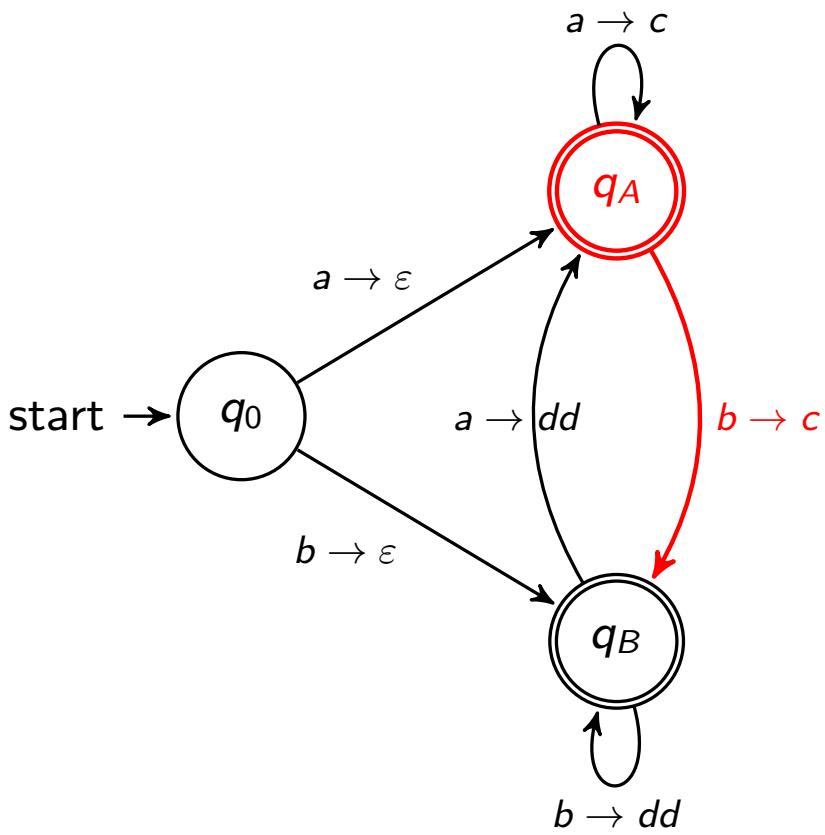


Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

Exemple : $aabab \rightarrow cc$

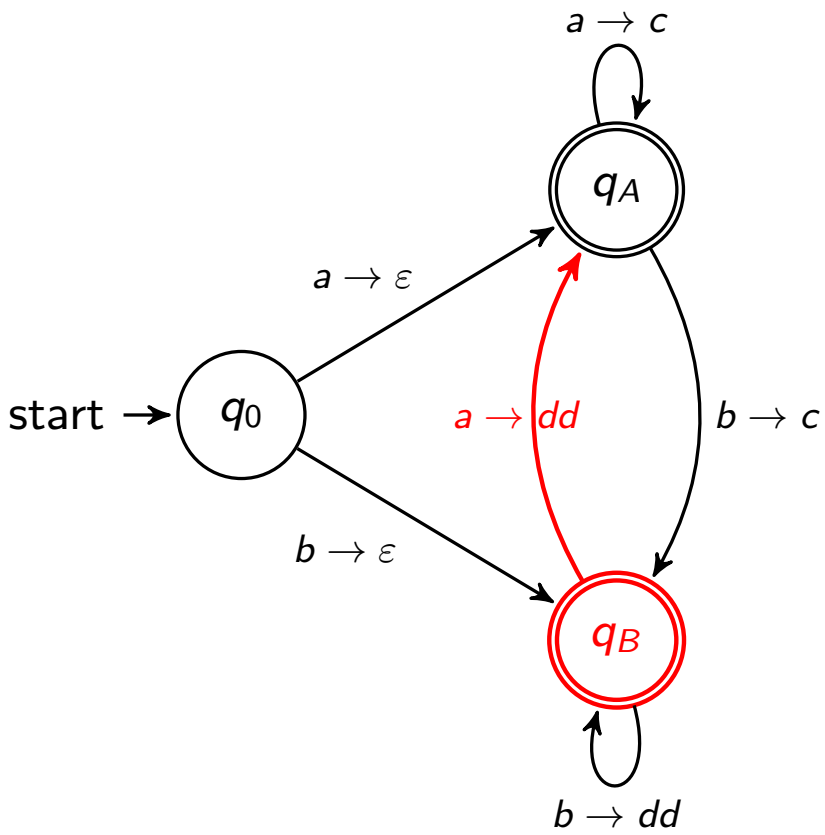


Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

Exemple : $aabab \rightarrow ccdd$

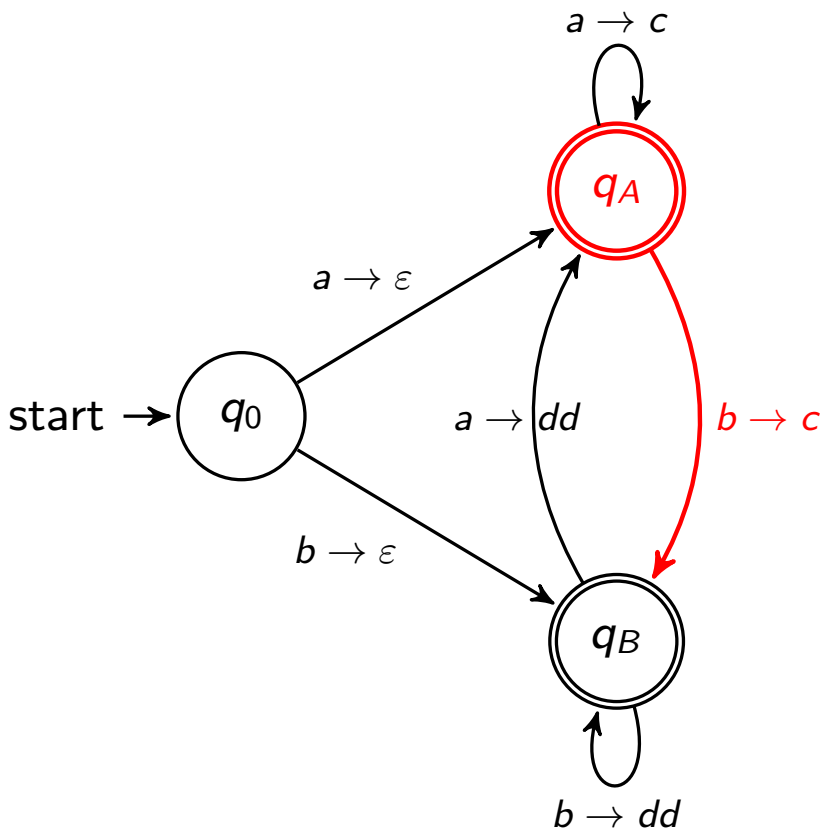


Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

Exemple : $aabab \rightarrow ccddc$

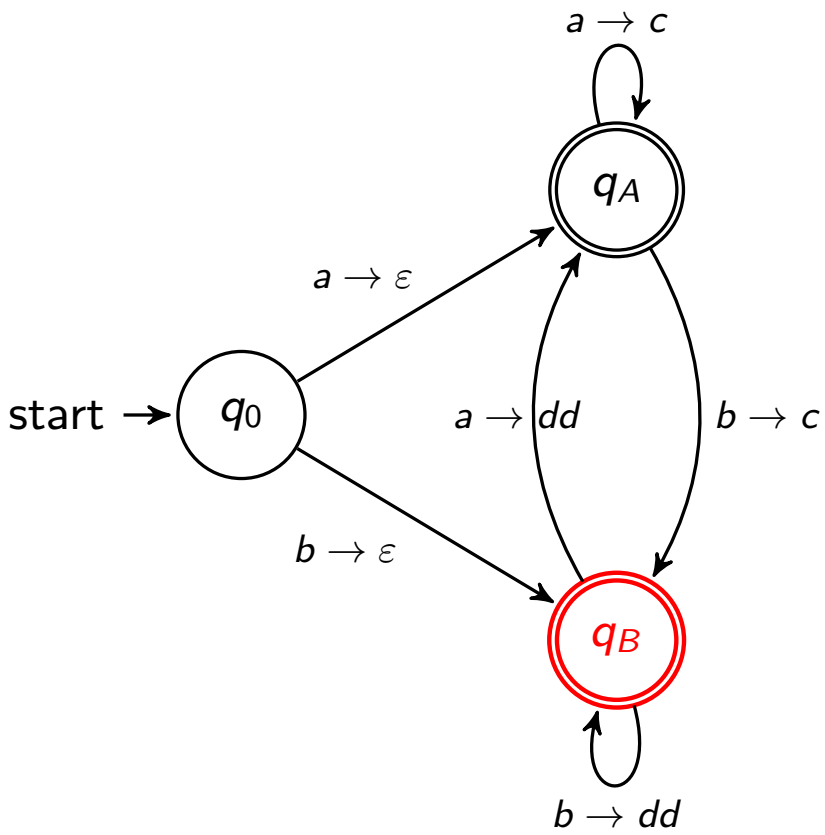


Generalised Sequential Machine (DGSM)

Un DGSM, c'est :

- Similaire à un automate fini
- Calcule un output
- Alphabet d'entrée $\Sigma = \{a, b\}$
- Alphabet de sortie $\Gamma = \{c, d\}$
- Pas d'ε-transition !

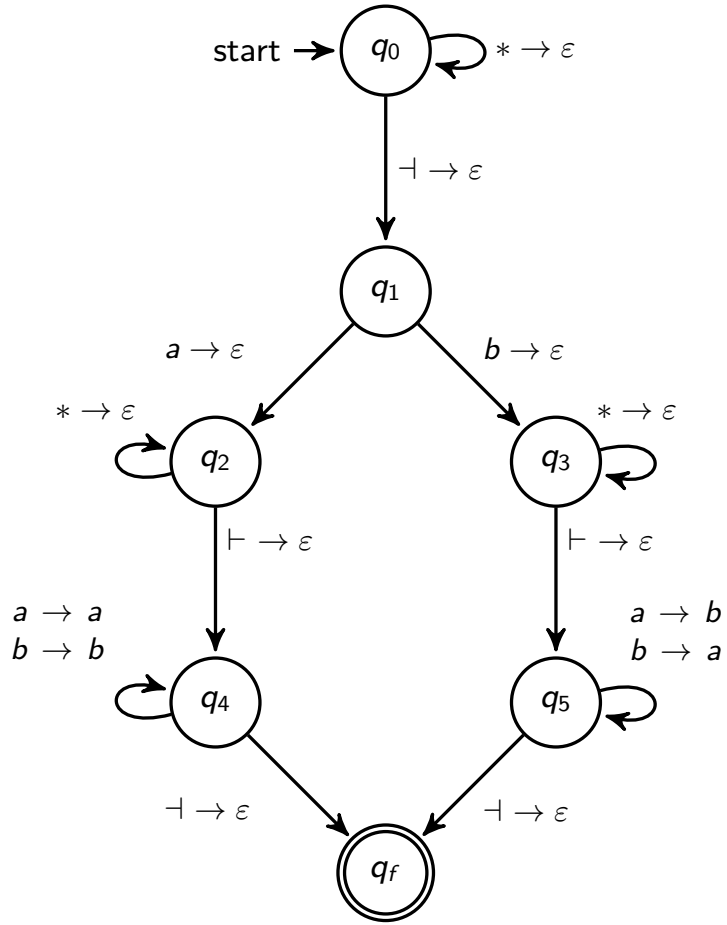
Exemple : $aabab \rightarrow ccddc$



Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

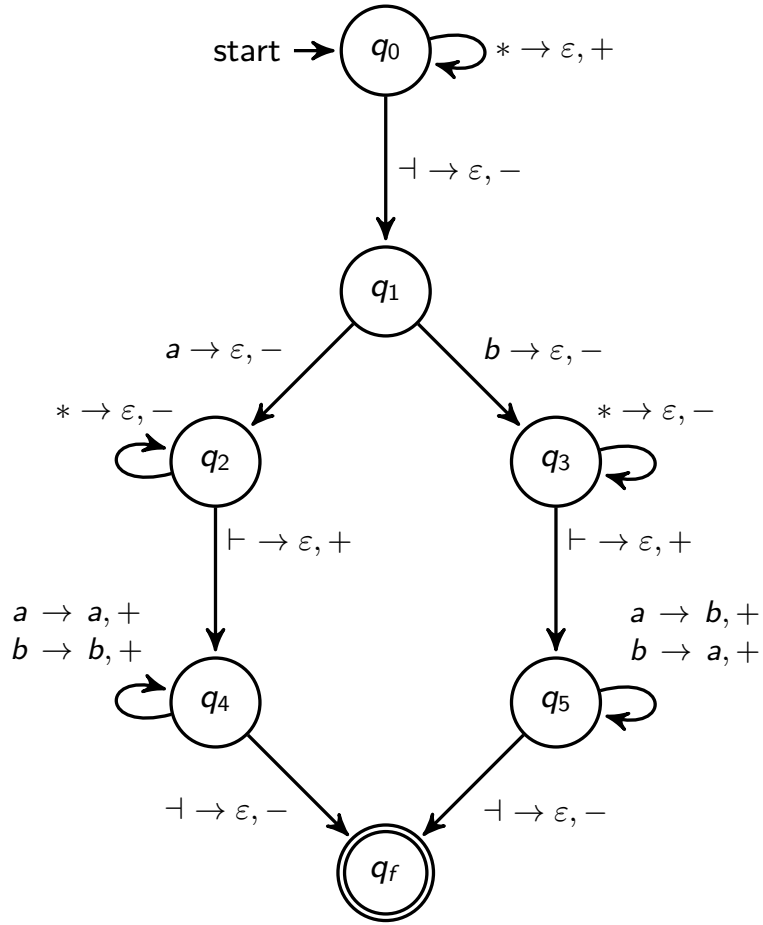
- Similaire à un DGSM



Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

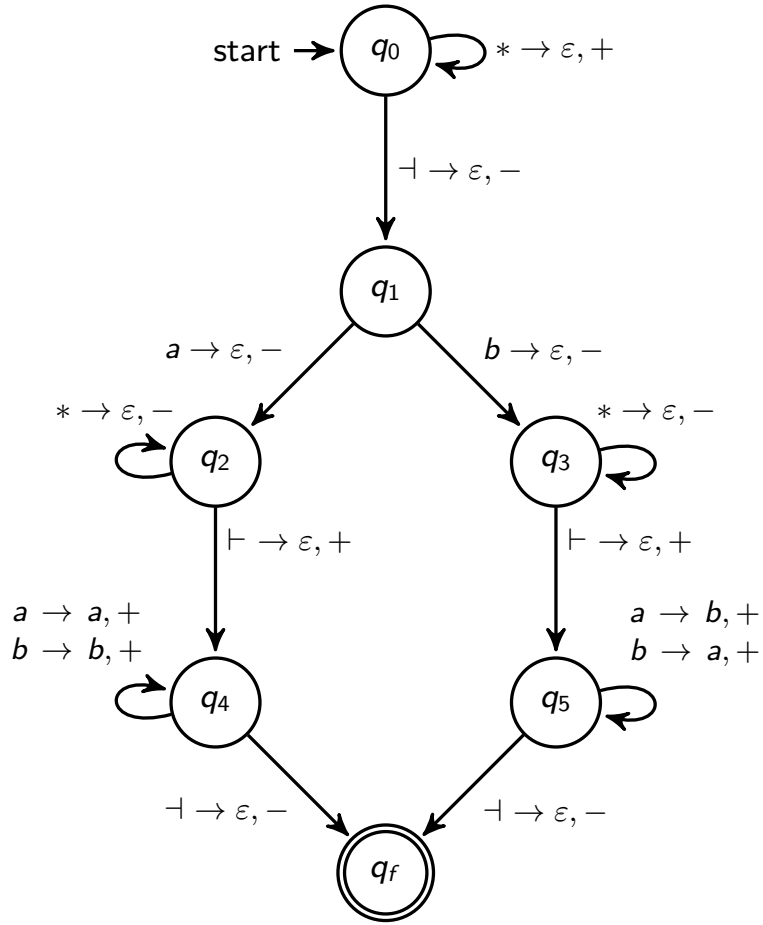
- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)



Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

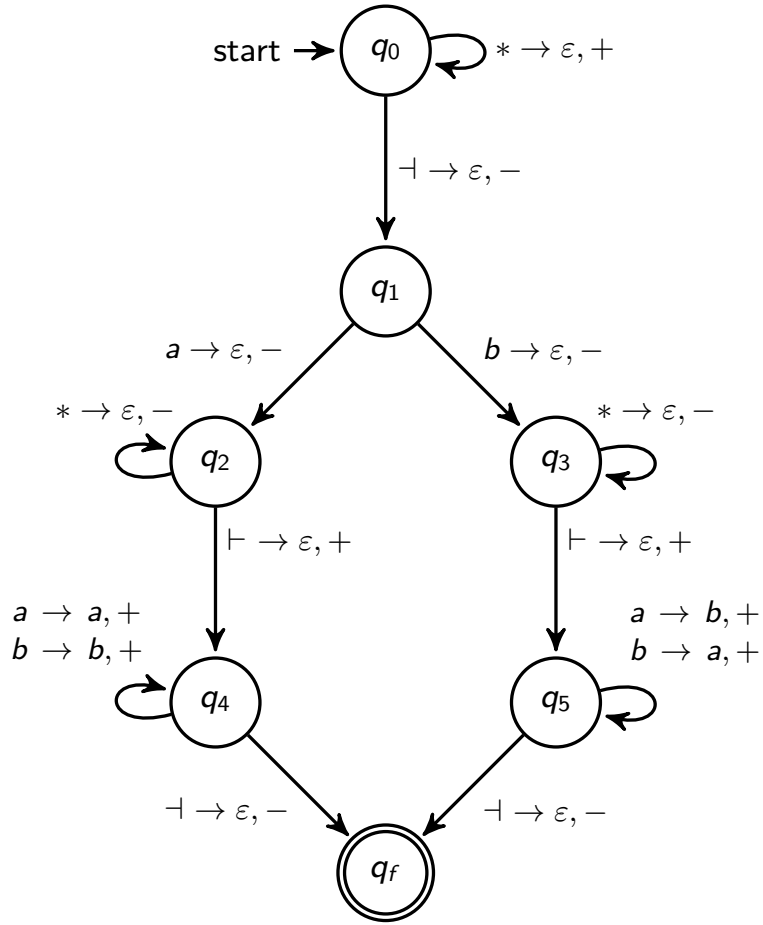
- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$



Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet



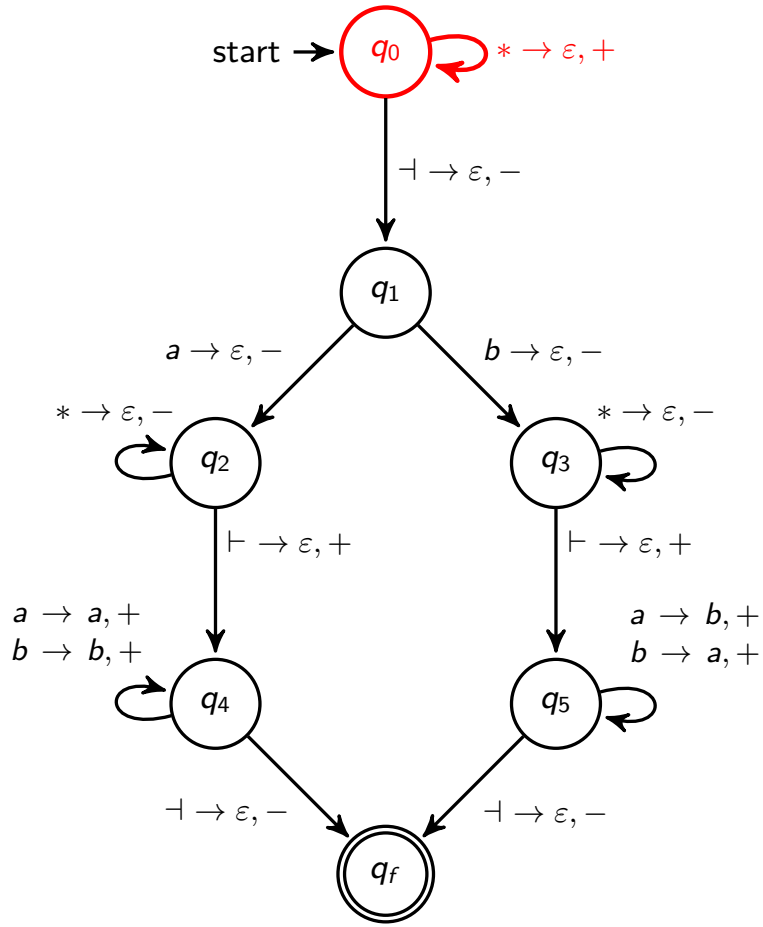
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



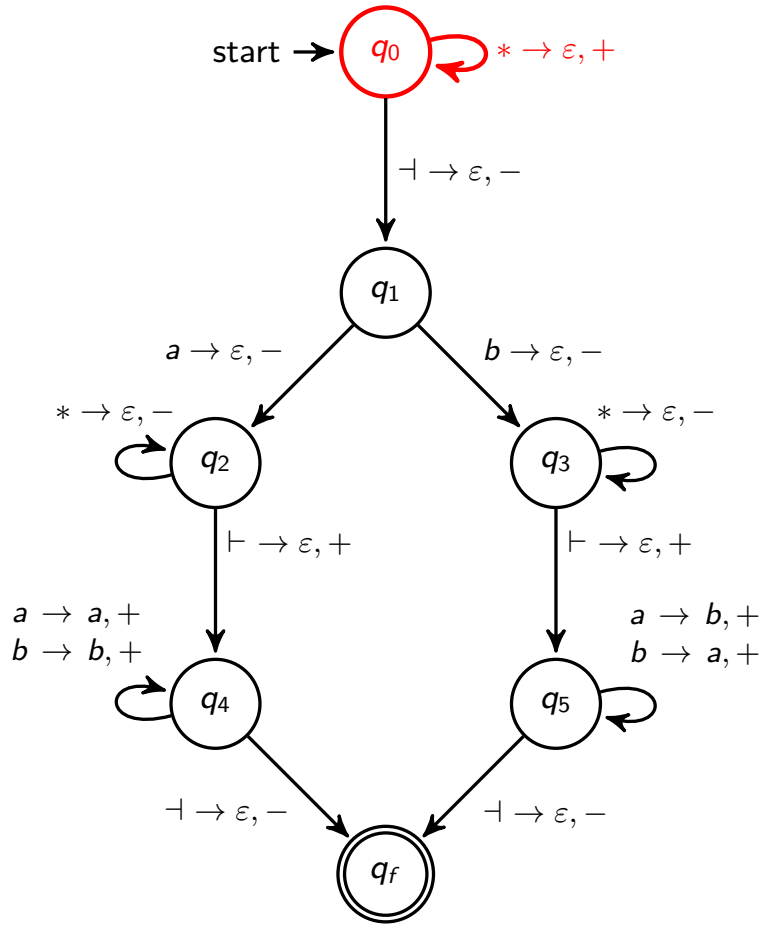
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbabab \dashv \quad \rightarrow$



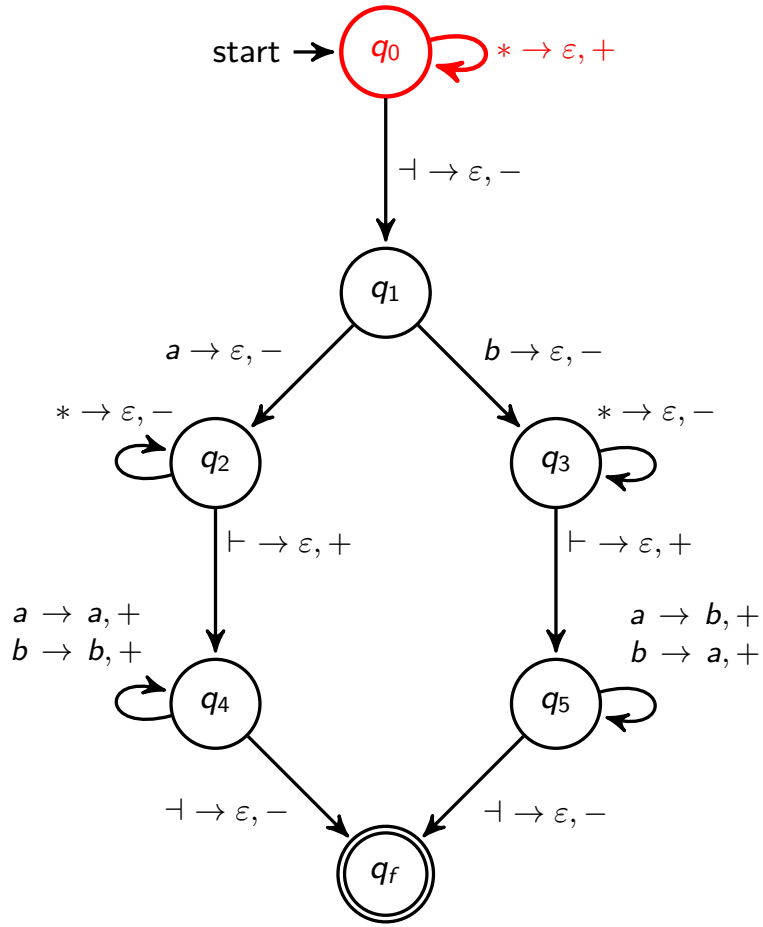
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



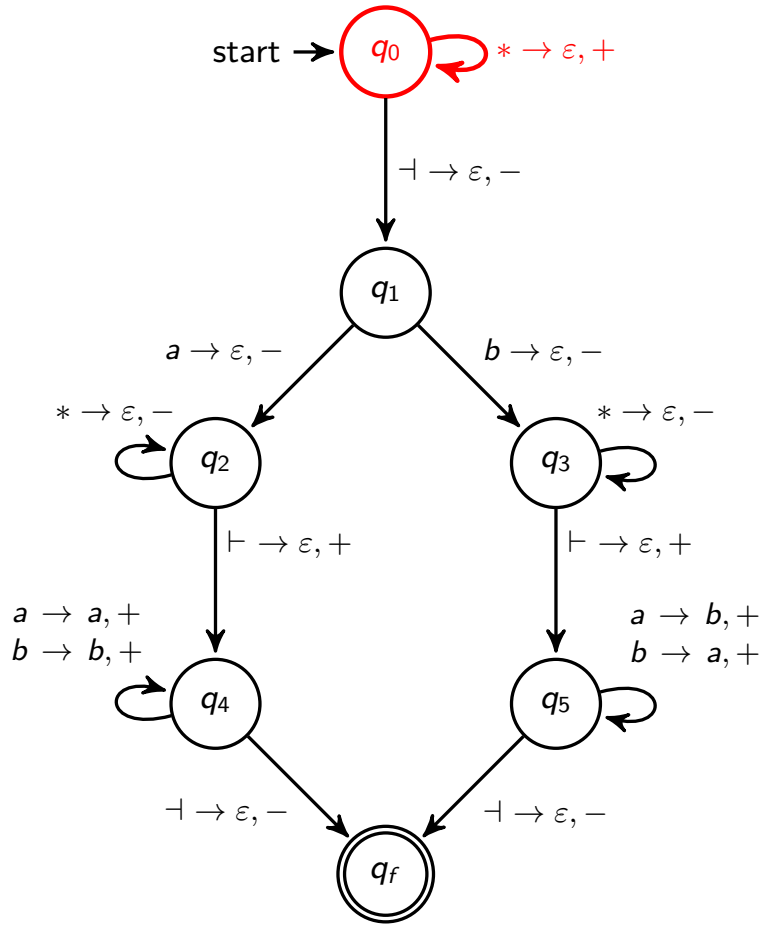
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



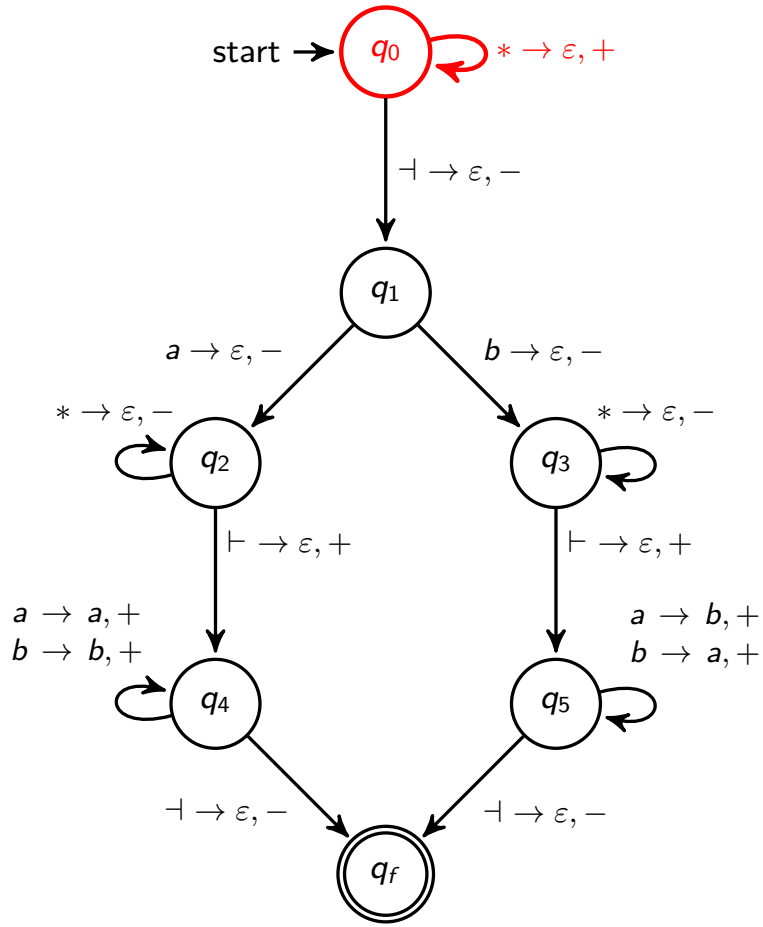
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



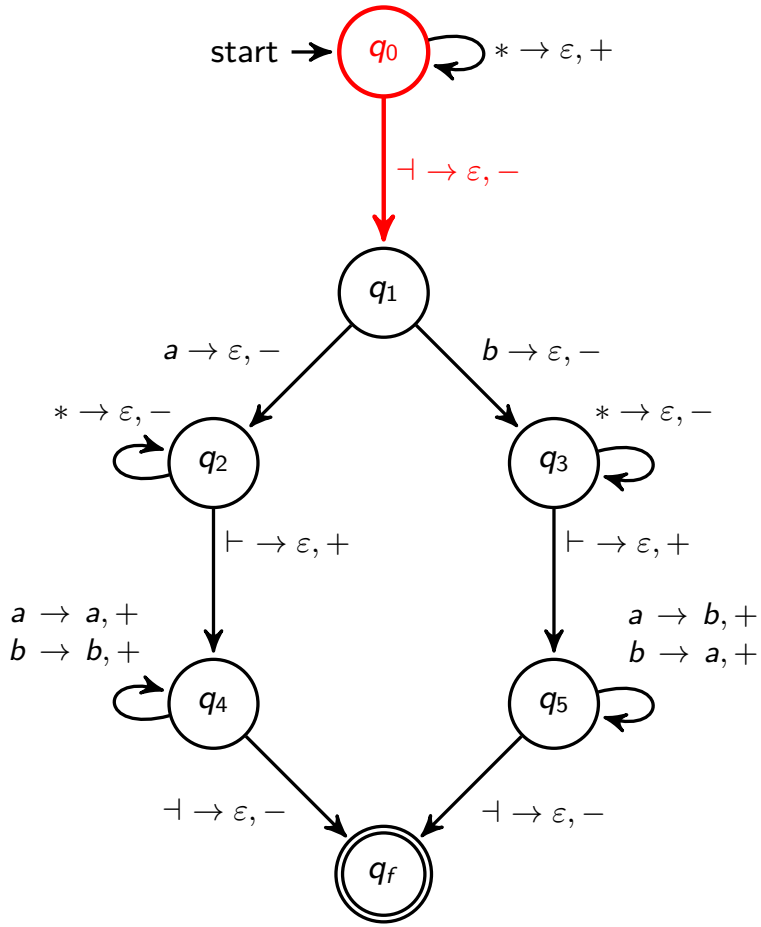
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$\vdash abbab \dashv \quad \rightarrow$$



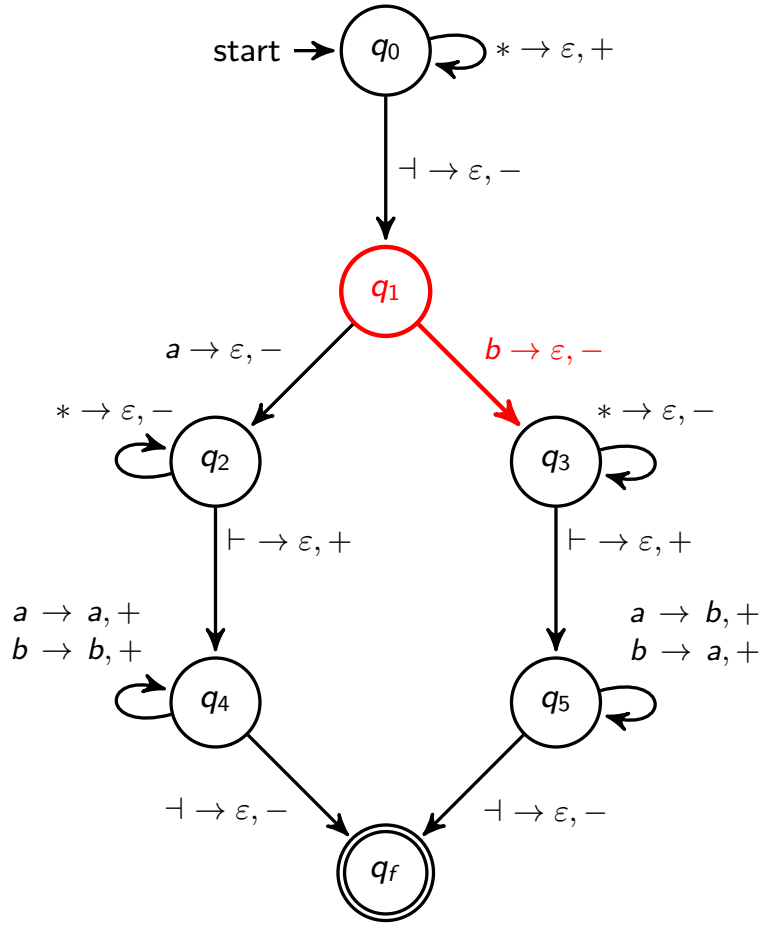
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



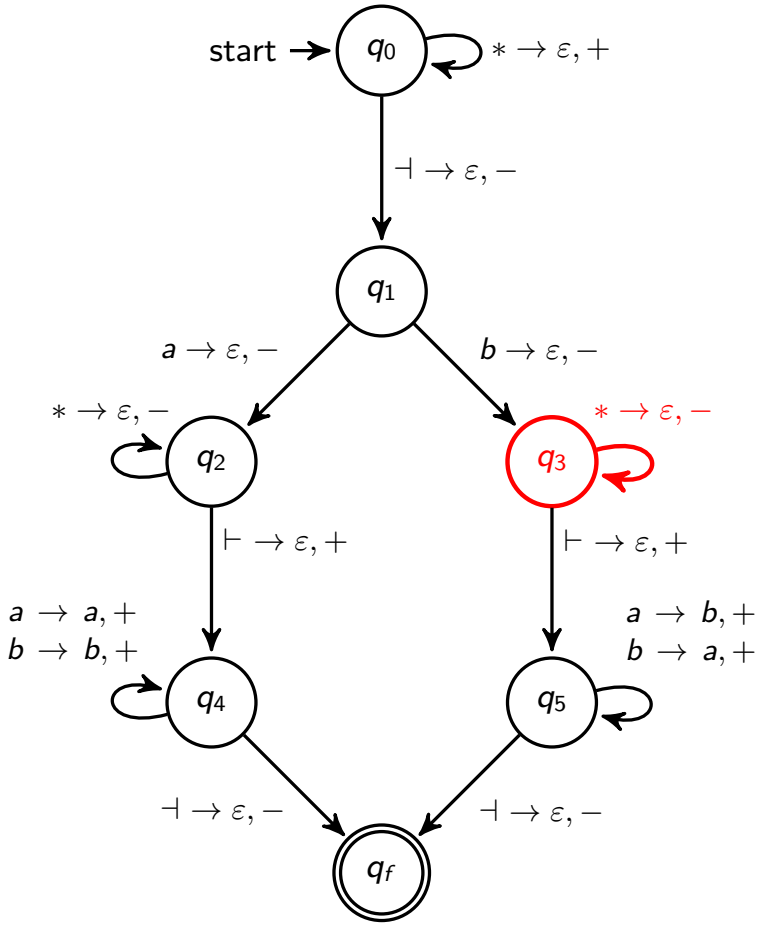
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



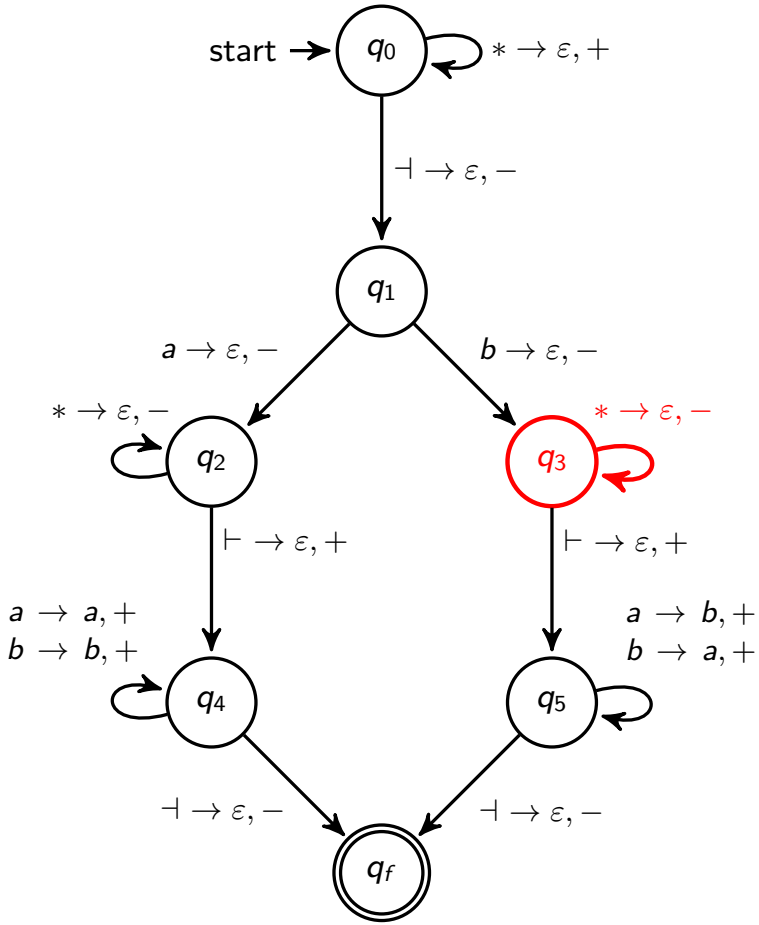
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



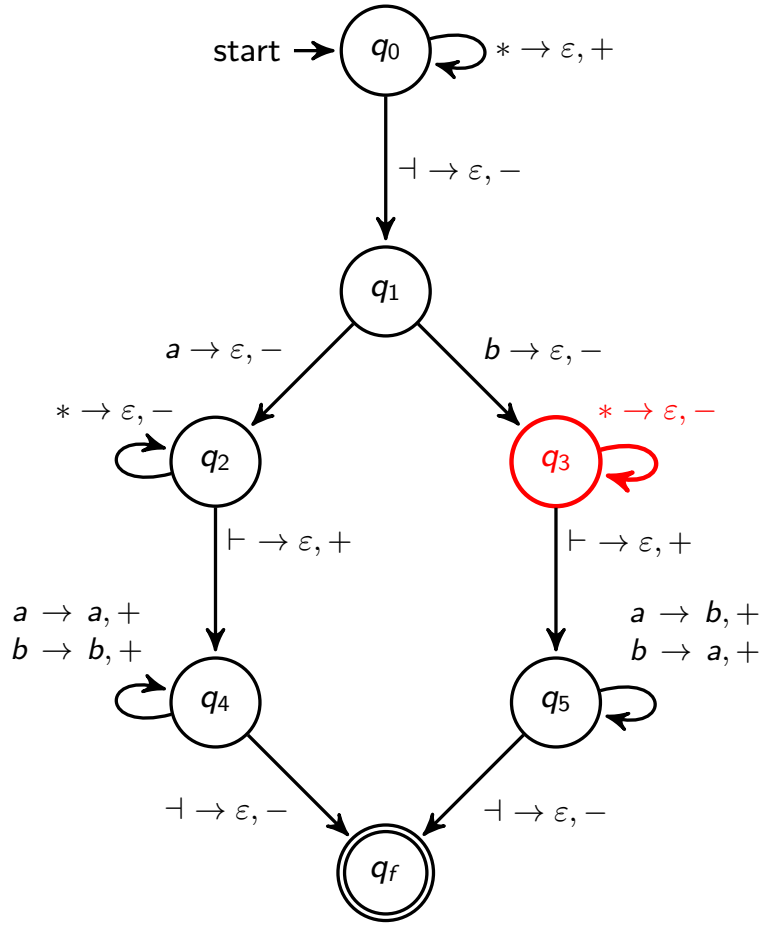
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbabab \dashv \quad \rightarrow$



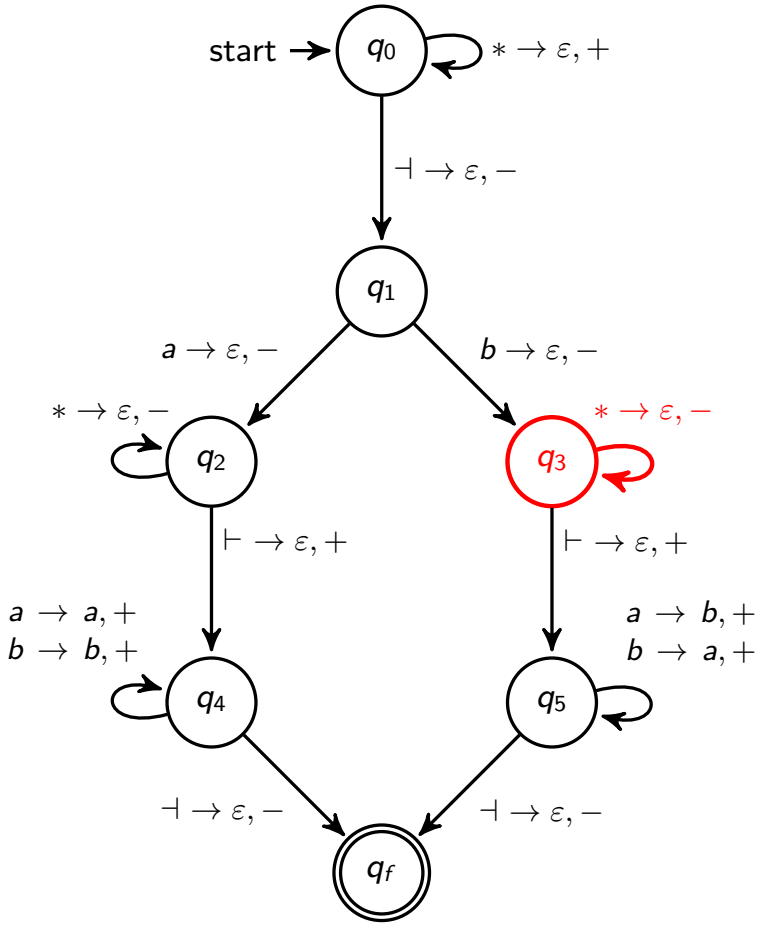
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



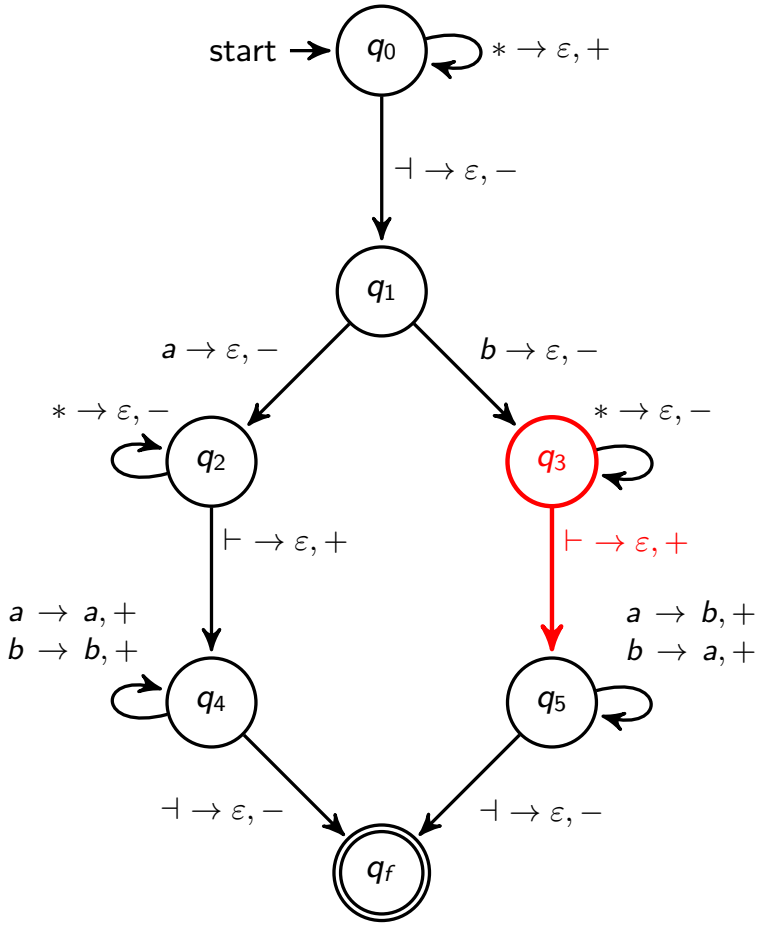
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow$



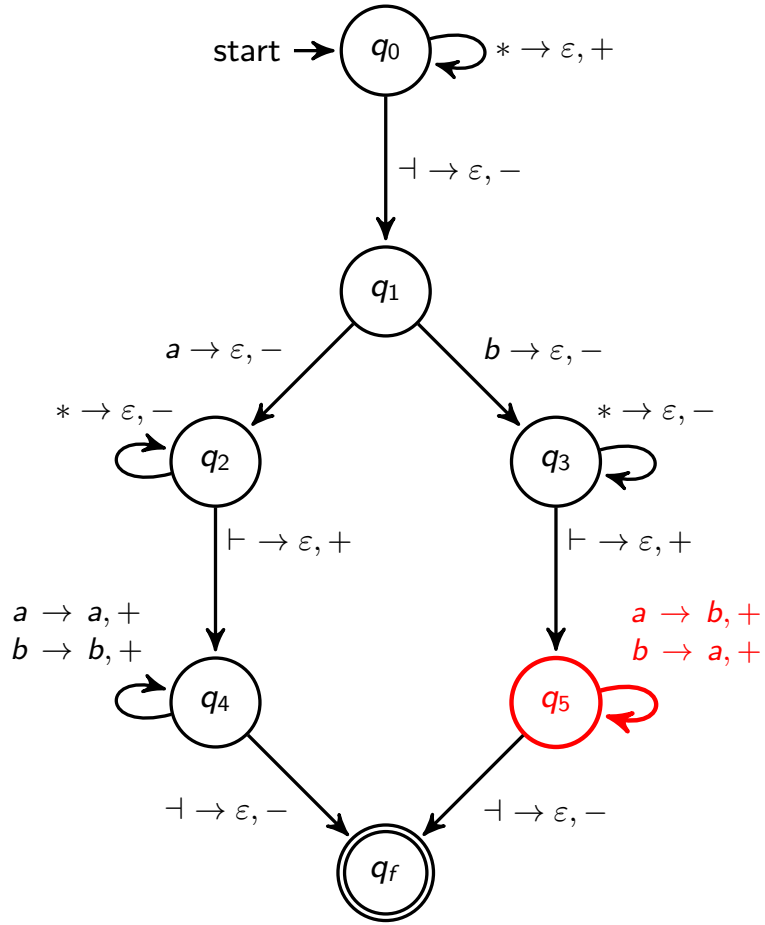
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbab \dashv \quad \rightarrow b$



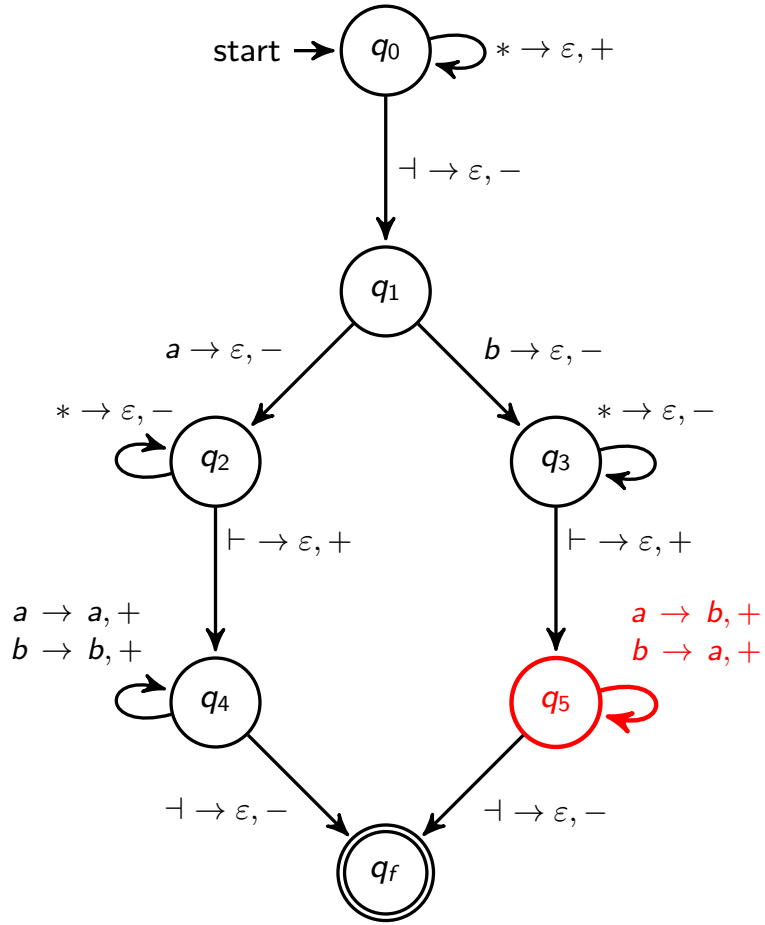
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$\vdash abbabab \dashv \quad \rightarrow ba$



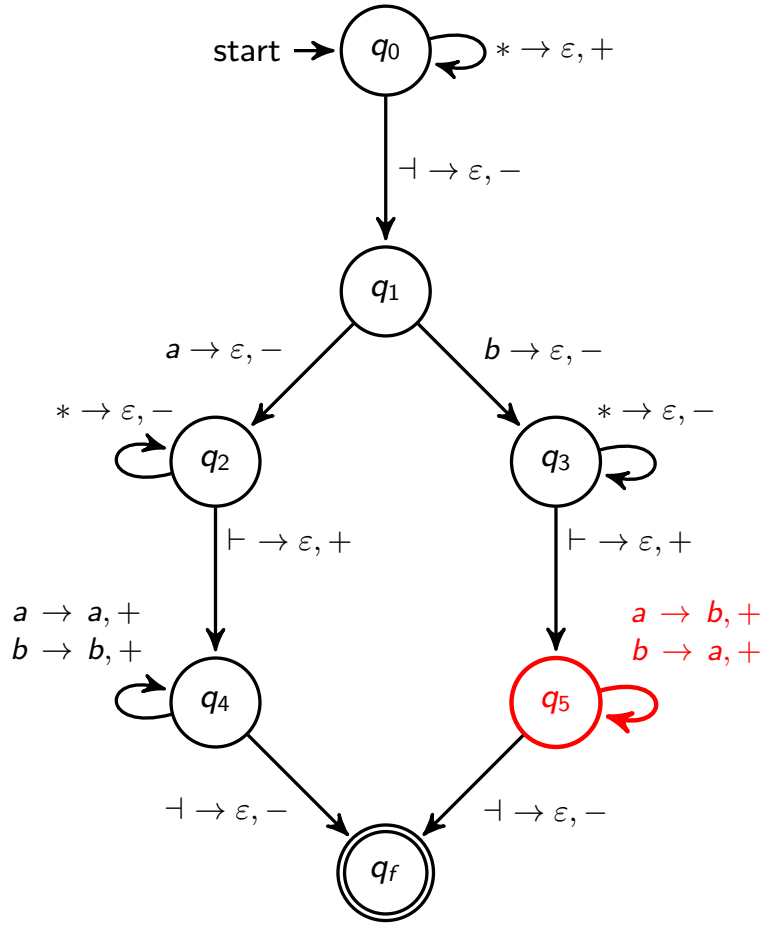
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$\vdash abbab\dashv \quad \rightarrow baa$$



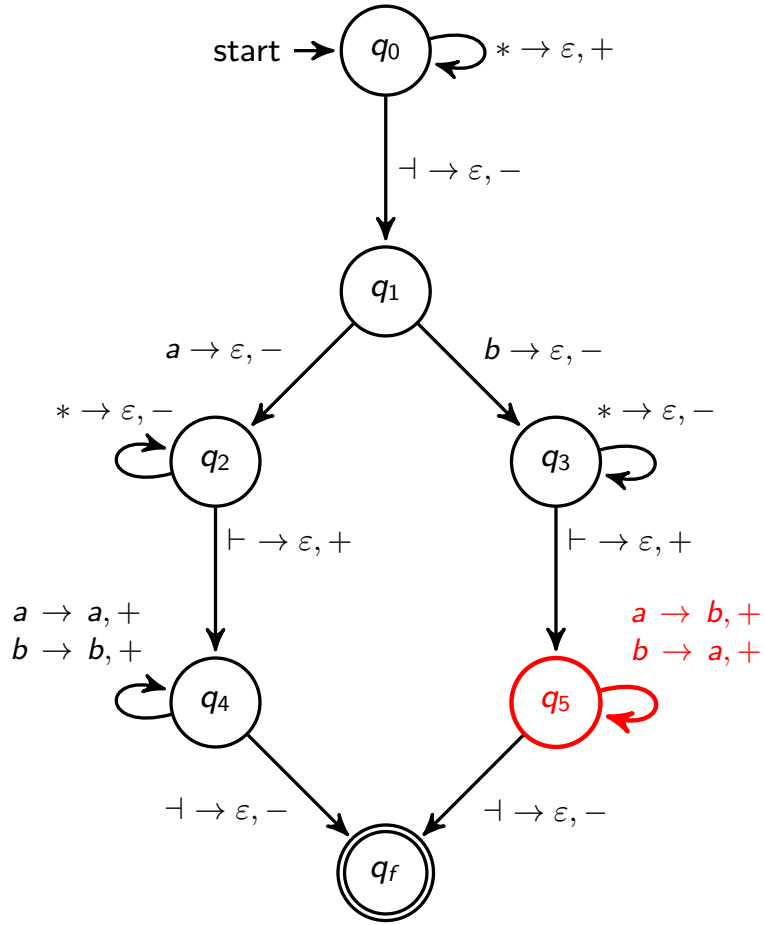
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$\vdash abbab\dashv \quad \rightarrow baaab$$



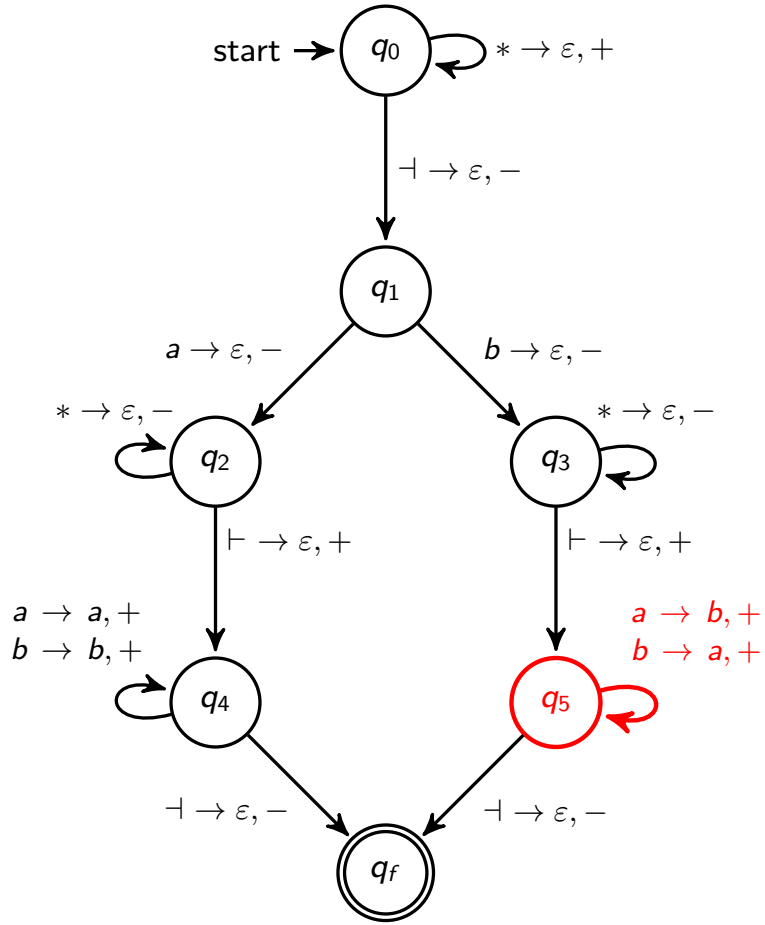
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$\vdash abbab \dashv \quad \rightarrow baaba$$



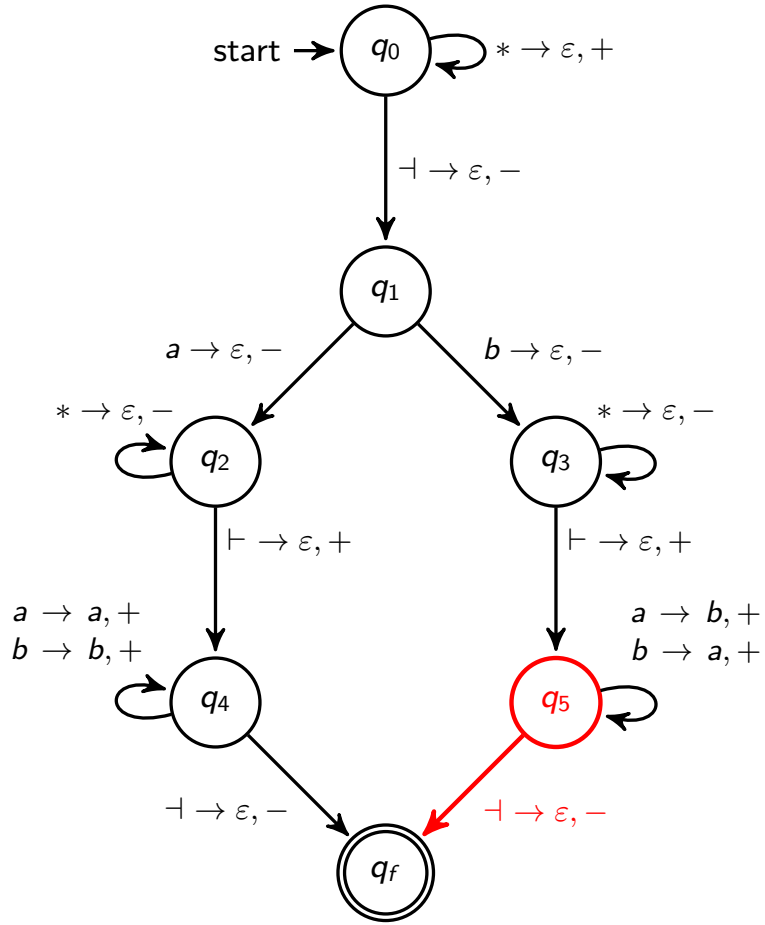
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$\vdash abbab\dashv \quad \rightarrow baaba$$



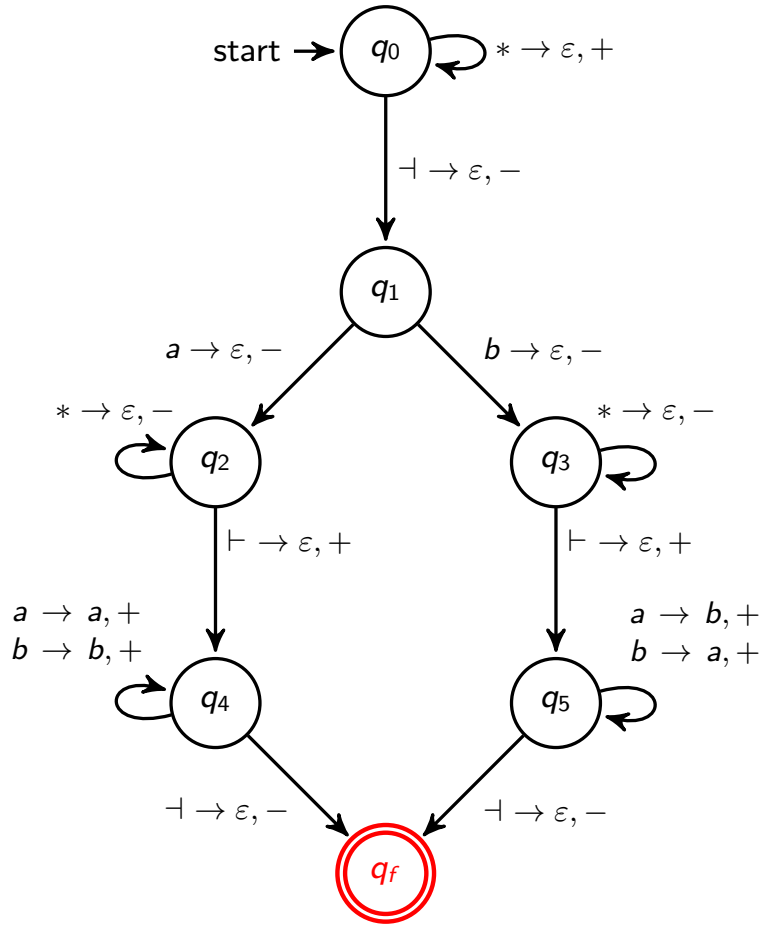
Two-way Generalised Sequential Machine (2DGSM)

Un 2DGSM, c'est :

- Similaire à un DGSM
- La tête de lecture peut bouger à droite (+) ou à gauche (-)
- Ruban d'entrée : $\vdash w \dashv$
- Si on sort du ruban, rejet

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

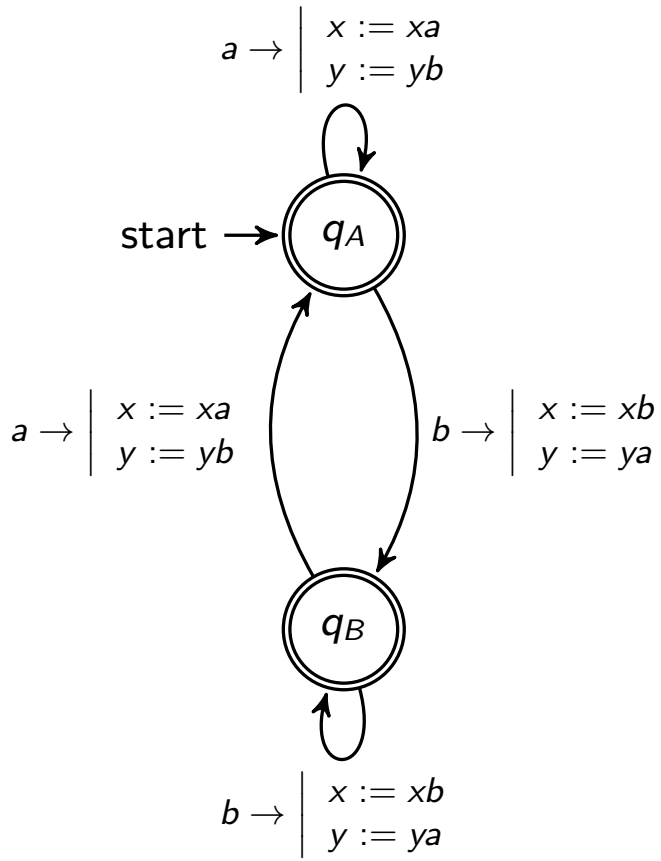
$$\vdash abbab \dashv \quad \rightarrow baaba$$



Streaming String Transducer (DSST)

Un DSST, c'est :

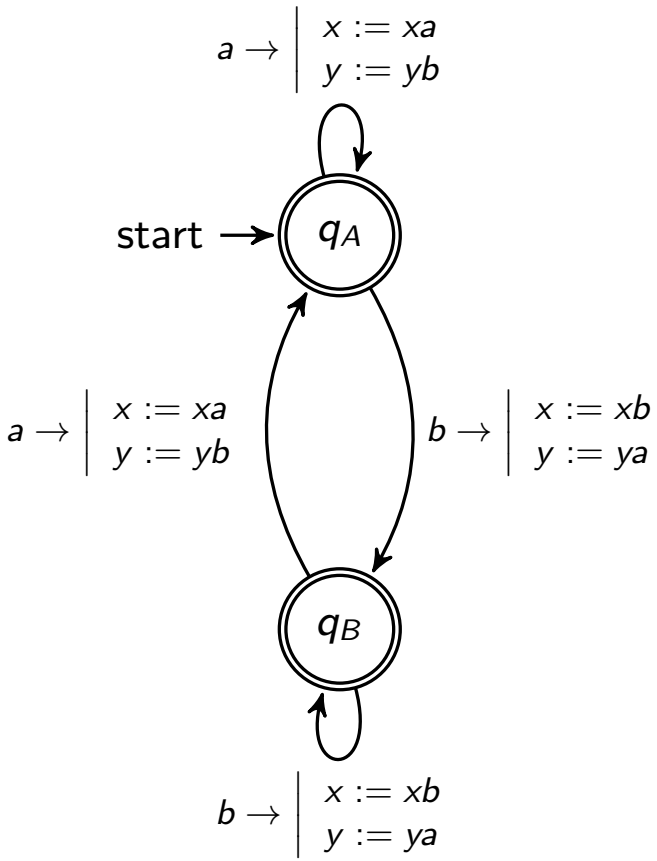
- Similaire à un automate fini



Streaming String Transducer (DSST)

Un DSST, c'est :

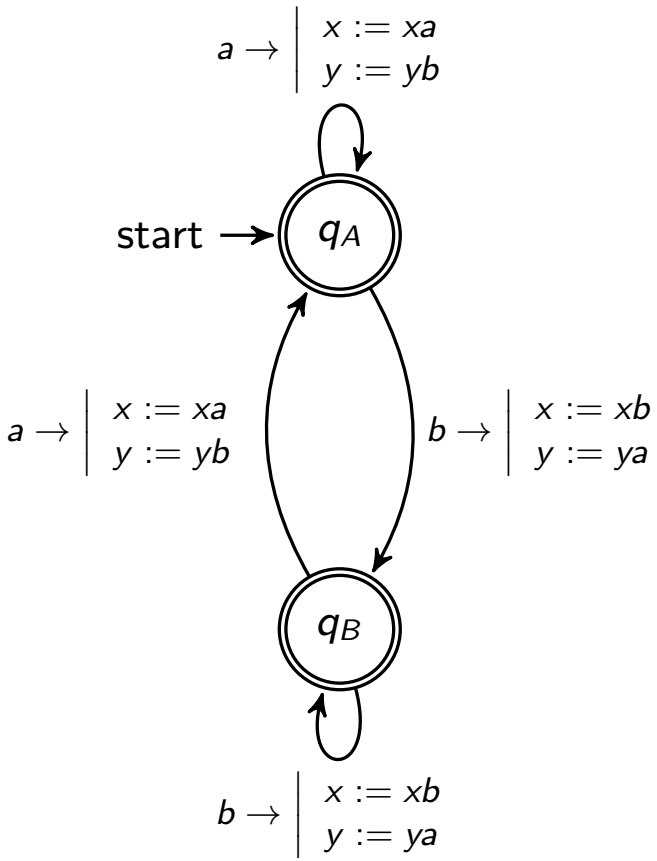
- Similaire à un automate fini
- Avec un ensemble fini X de variables



Streaming String Transducer (DSST)

Un DSST, c'est :

- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition

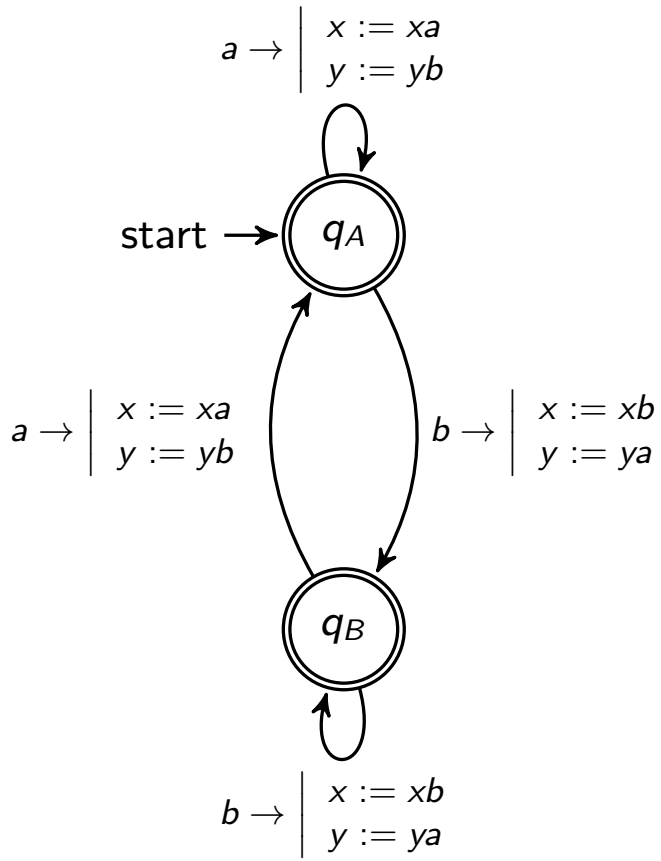


Streaming String Transducer (DSST)

Un DSST, c'est :

- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left|\begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array}\right. \text{, mais } \left|\begin{array}{l} x := y \\ y := x.y \end{array}\right.$$



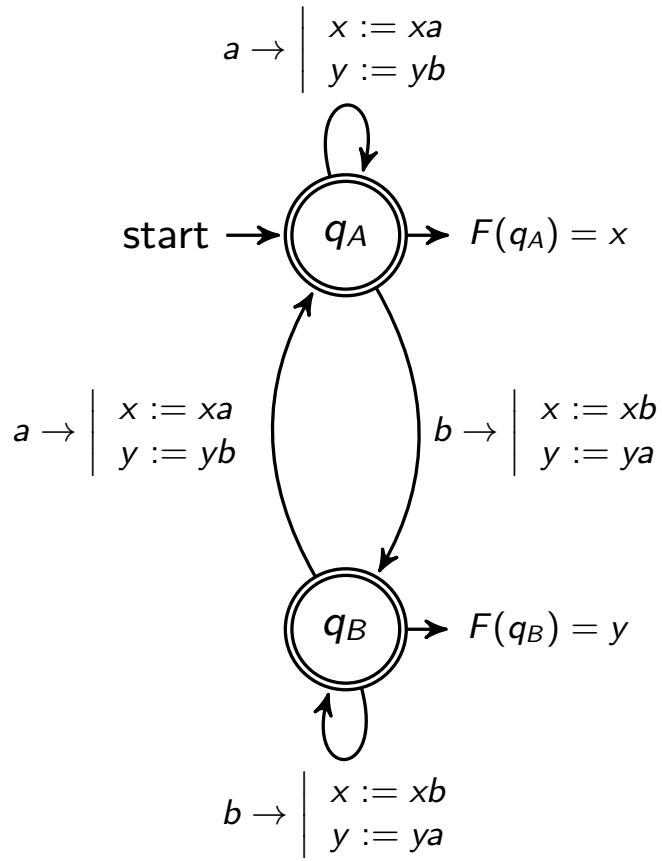
Streaming String Transducer (DSST)

Un DSST, c'est :

- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie



Streaming String Transducer (DSST)

Un DSST, c'est :

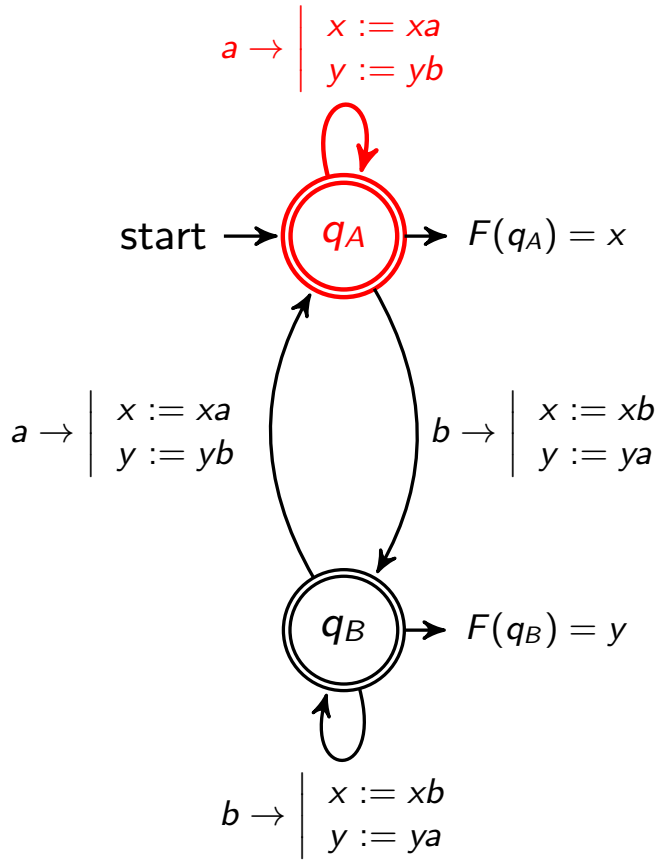
- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$\underset{\uparrow}{a}bbab \rightarrow \begin{cases} x = a \\ y = b \end{cases}$$



Streaming String Transducer (DSST)

Un DSST, c'est :

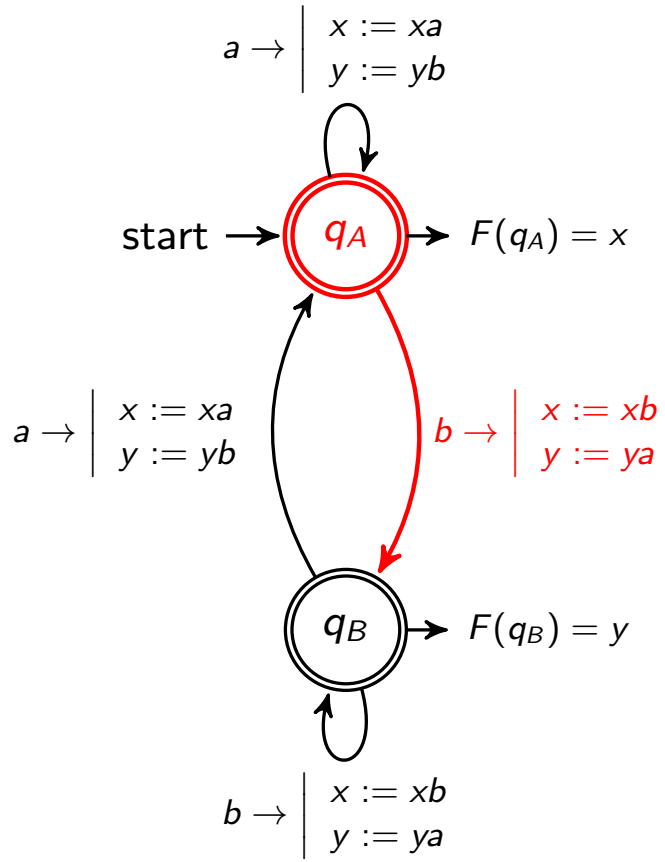
- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$abbab \rightarrow \begin{cases} x = ab \\ y = ba \end{cases}$$



Streaming String Transducer (DSST)

Un DSST, c'est :

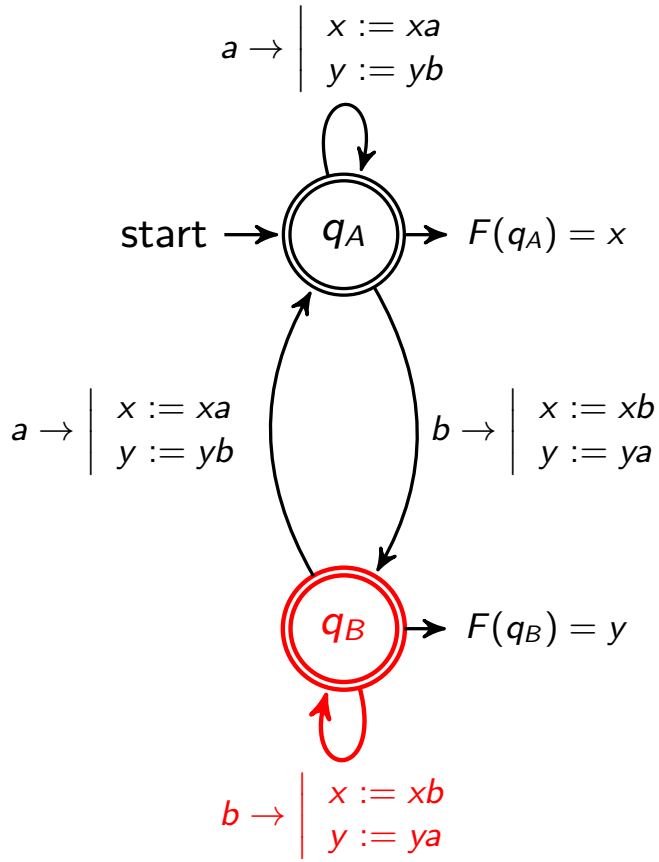
- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$abbab \rightarrow \begin{cases} x = abb \\ y = baa \end{cases}$$



Streaming String Transducer (DSST)

Un DSST, c'est :

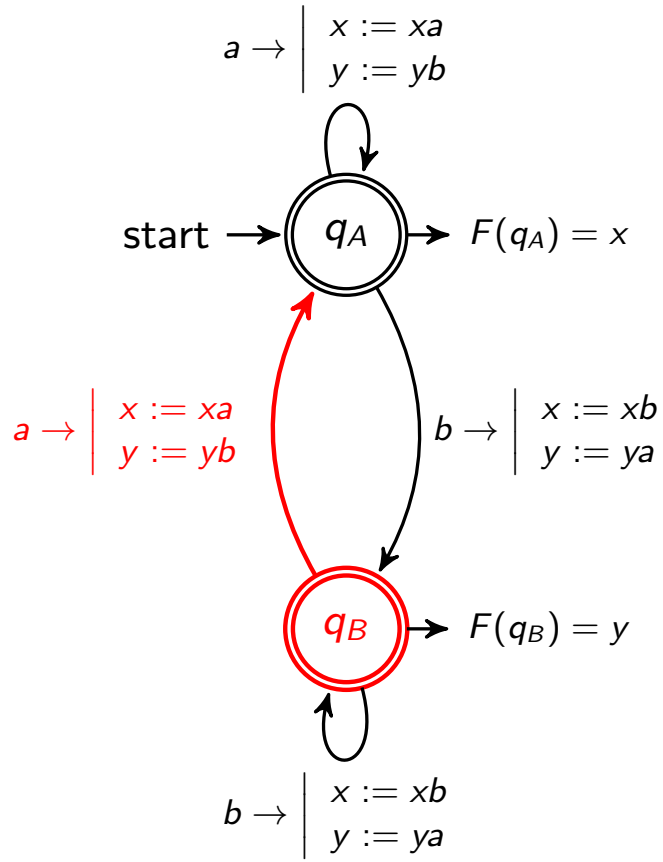
- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$abbab \rightarrow \begin{cases} x = abba \\ y = baab \end{cases}$$



Streaming String Transducer (DSST)

Un DSST, c'est :

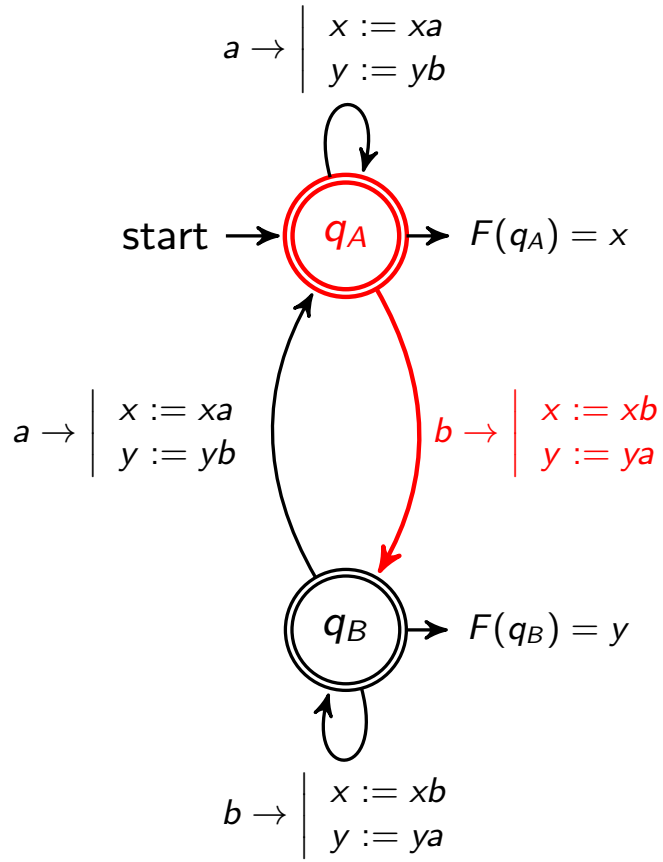
- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$abbab \rightarrow \begin{cases} x = abbab \\ y = baaba \end{cases}$$



Streaming String Transducer (DSST)

Un DSST, c'est :

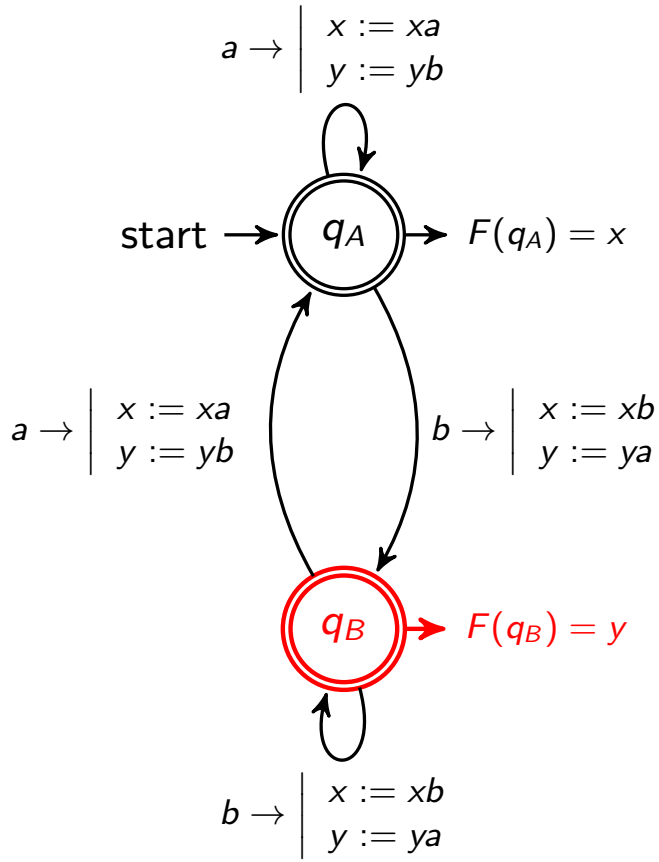
- Similaire à un automate fini
- Avec un ensemble fini X de variables
- Mise à jour en parallèle des variables à chaque transition
- Condition de non-copie :

$$\left| \begin{array}{l} x := \alpha.y.\beta.x \\ y := \gamma \end{array} \right. \text{, mais } \left| \begin{array}{l} x := y \\ y := x.y \end{array} \right.$$

- Fonction d'output définie sur les états finaux : $F(q) \in (X \cup \Gamma)^*$, sans copie

Exemple : Si l'input finit par a, on le recopie. S'il finit par b, on le recopie en échangeant les a et les b.

$$abbab \rightarrow \begin{cases} x = abbab \\ y = baaba \end{cases}$$



Plan

Expressivité des Streaming String Transducers

Cas déterministe :

Théorème (Alur & Cerny)

$\mathrm{DSST} = \mathrm{DMSOS} = 2\mathrm{DGSM}$

Expressivité des Streaming String Transducers

Cas déterministe :

Théorème (Alur & Cerny)

$\mathrm{DSST} = \mathrm{DMSOS} = 2\mathrm{DGSM}$

Inconvénient : Pas de réduction directe entre $2\mathrm{DGSM}$ et DSST.

Expressivité des Streaming String Transducers

Cas déterministe :

Théorème (Alur & Cerny)

$\mathrm{DSST} = \mathrm{DMSOS} = 2\mathrm{DGSM}$

Inconvénient : Pas de réduction directe entre $2\mathrm{DGSM}$ et DSST.

Cas non-déterministe :

Théorème (Alur & Deshmukh)

$\mathrm{NSST} = \mathrm{NMSOS} \neq 2\mathrm{NGSM}$

Expressivité des Streaming String Transducers

Cas déterministe :

Théorème (Alur & Cerny)

$\text{DSST} = \text{DMSOS} = 2\text{DGSM}$

Inconvénient : Pas de réduction directe entre 2DGSM et DSST.

Cas non-déterministe :

Théorème (Alur & Deshmukh)

$\text{NSST} = \text{NMSOS} \neq 2\text{NGSM}$

Aspects algorithmiques :

- Equivalence des NSST : indécidable

Expressivité des Streaming String Transducers

Cas déterministe :

Théorème (Alur & Cerny)

$\text{DSST} = \text{DMSOS} = 2\text{DGSM}$

Inconvénient : Pas de réduction directe entre 2DGSM et DSST.

Cas non-déterministe :

Théorème (Alur & Deshmukh)

$\text{NSST} = \text{NMSOS} \neq 2\text{NGSM}$

Aspects algorithmiques :

- Equivalence des NSST : indécidable
- Equivalence des NSST fonctionnels : PSPACE-COMPLET

Expressivité des Streaming String Transducers

Cas déterministe :

Théorème (Alur & Cerny)

$\text{DSST} = \text{DMSOS} = 2\text{DGSM}$

Inconvénient : Pas de réduction directe entre 2DGSM et DSST.

Cas non-déterministe :

Théorème (Alur & Deshmukh)

$\text{NSST} = \text{NMSOS} \neq 2\text{NGSM}$

Aspects algorithmiques :

- Equivalence des NSST : indécidable
- Equivalence des NSST fonctionnels : PSPACE-COMPLET
- Les NSST fonctionnels sont déterminisables

Travail effectué pendant le stage

- Preuve de $\mathrm{DSST} \subseteq \mathrm{DMSOS}$

Travail effectué pendant le stage

- Preuve de $\text{DSST} \subseteq \text{DMSOS}$
- Réduction directe de DSST vers 2DGSM
  $\rightarrow$ Relation entre le nombre de variables et le nombre de visites ?

Travail effectué pendant le stage

- Preuve de $\text{DSST} \subseteq \text{DMSOS}$
- Réduction directe de DSST vers 2DGSM
  $\rightarrow$ Relation entre le nombre de variables et le nombre de visites ?
- Contre-exemples dans le cas des modèles de base

Travail effectué pendant le stage

- Preuve de $\mathrm{DSST} \subseteq \mathrm{DMSOS}$
- Réduction directe de DSST vers 2DGSM
  $\rightarrow$ Relation entre le nombre de variables et le nombre de visites ?
- Contre-exemples dans le cas des modèles de base
- Preuve dans le cas de machines avec *regular look-around* :
  - $\mathrm{DSST}_{\mathrm{RLA}}$ à k variables $\rightarrow$ $2\mathrm{DGSM}_{\mathrm{RLA}}$ à $(2k+1)$ visites
  - $2\mathrm{DGSM}_{\mathrm{RLA}}$ à k visites $\rightarrow$ $\mathrm{DSST}_{\mathrm{RLA}}$ à $\lceil \frac{k}{2} \rceil$ variables

Travail effectué pendant le stage

- Preuve de $\text{DSST} \subseteq \text{DMSOS}$
- Réduction directe de DSST vers 2DGSM
  $\rightarrow$ Relation entre le nombre de variables et le nombre de visites ?
- Contre-exemples dans le cas des modèles de base
- Preuve dans le cas de machines avec *regular look-around* :
  - DSST_{RLA} à k variables $\rightarrow$ $2\text{DGSM}_{\text{RLA}}$ à $(2k+1)$ visites
  - $2\text{DGSM}_{\text{RLA}}$ à k visites $\rightarrow$ DSST_{RLA} à $\lceil \frac{k}{2} \rceil$ variables
- Cas non-déterministe : équivalence des NSST k-valués ?
  - Preuve de Karhumaki et Culik sur les 2NGSM k-valués
  - k-val. NSST $\sim$ k-val. 2NGSM ?

Travail effectué pendant le stage

- Preuve de $\text{DSST} \subseteq \text{DMSOS}$
- Réduction directe de DSST vers 2DGSM
  $\rightarrow$ Relation entre le nombre de variables et le nombre de visites ?
- Contre-exemples dans le cas des modèles de base
- Preuve dans le cas de machines avec *regular look-around* :
  - DSST_{RLA} à k variables $\rightarrow$ $2\text{DGSM}_{\text{RLA}}$ à $(2k+1)$ visites
  - $2\text{DGSM}_{\text{RLA}}$ à k visites $\rightarrow$ DSST_{RLA} à $\lceil \frac{k}{2} \rceil$ variables
- Cas non-déterministe : équivalence des NSST k-valués ?
  - Preuve de Karhumaki et Culik sur les 2NGSM k-valués
  - k-val. NSST $\sim$ k-val. 2NGSM ?

Plan

Propriété de Hopcroft

On utilise le théorème suivant :

Théorème (Hopcroft & Ullman)

$\text{DGSM} \circ 2\text{DGSM} \subseteq 2\text{DGSM}$

Propriété de Hopcroft

On utilise le théorème suivant :

Théorème (Hopcroft & Ullman)

$\text{DGSM} \circ 2\text{DGSM} \subseteq 2\text{DGSM}$

On part d'un DSST W. L'idée est la suivante :

- DGSM A : input $\rightarrow$ opérations de W (relabeling)

Propriété de Hopcroft

On utilise le théorème suivant :

Théorème (Hopcroft & Ullman)

$\text{DGSM} \circ 2\text{DGSM} \subseteq 2\text{DGSM}$

On part d'un DSST W. L'idée est la suivante :

- DGSM A : input $\rightarrow$ opérations de W (relabeling)
- 2DGSM B : opérations $\rightarrow$ output

Propriété de Hopcroft

On utilise le théorème suivant :

Théorème (Hopcroft & Ullman)

$\text{DGSM} \circ 2\text{DGSM} \subseteq 2\text{DGSM}$

On part d'un DSST W. L'idée est la suivante :

- DGSM A : input $\rightarrow$ opérations de W (relabeling)
- 2DGSM B : opérations $\rightarrow$ output
- D'après le théorème, $A \circ B$ est réalisable par un 2DGSM

Propriété de Hopcroft

On utilise le théorème suivant :

Théorème (Hopcroft & Ullman)

$\text{DGSM} \circ 2\text{DGSM} \subseteq 2\text{DGSM}$

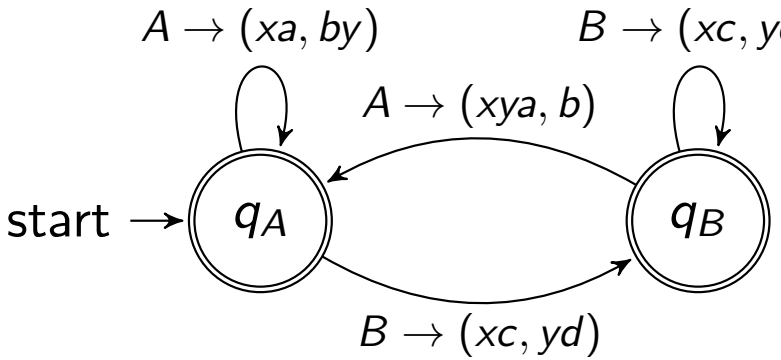
On part d'un DSST W. L'idée est la suivante :

- DGSM A : input $\rightarrow$ opérations de W (relabeling)
- 2DGSM B : opérations $\rightarrow$ output
- D'après le théorème, $A \circ B$ est réalisable par un 2DGSM

Notation : $\left| \begin{array}{l} x := \alpha x \beta \\ y := \varepsilon \end{array} \right.$ sera noté $(\alpha x \beta, \varepsilon)$

Idée générale

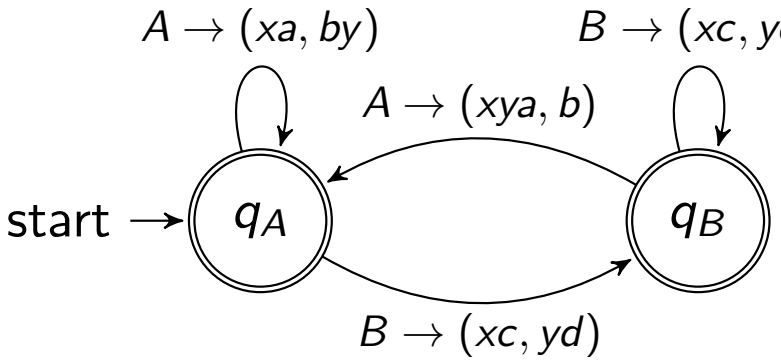
Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



$F(q_A) = F(q_B) = xy$

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

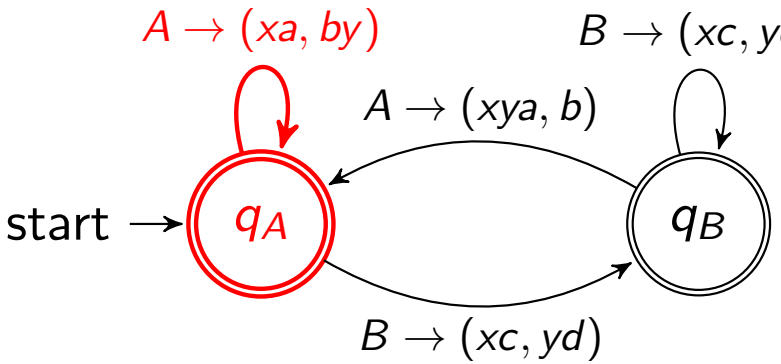


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	
B	
A	
B	

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

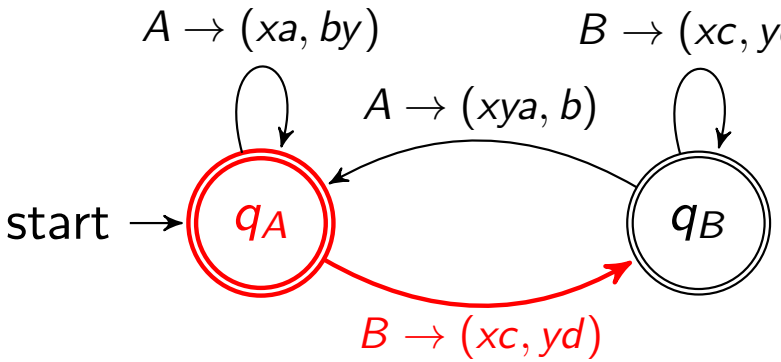


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	
A	
B	

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

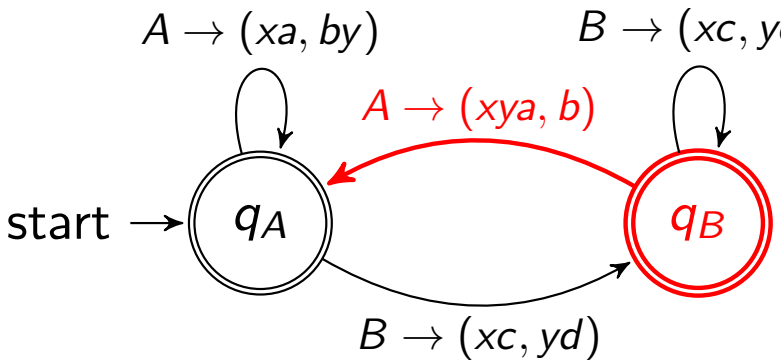


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	
B	

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

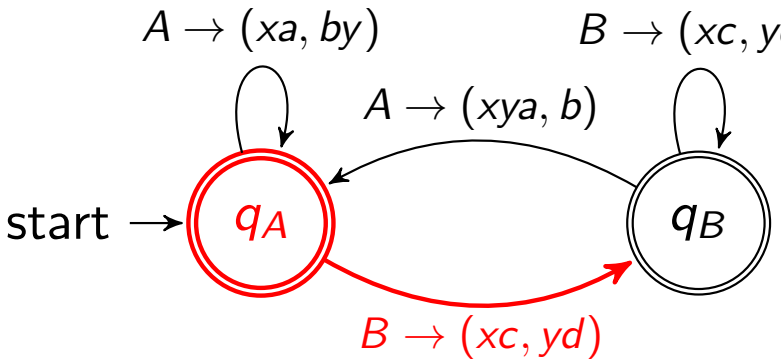


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

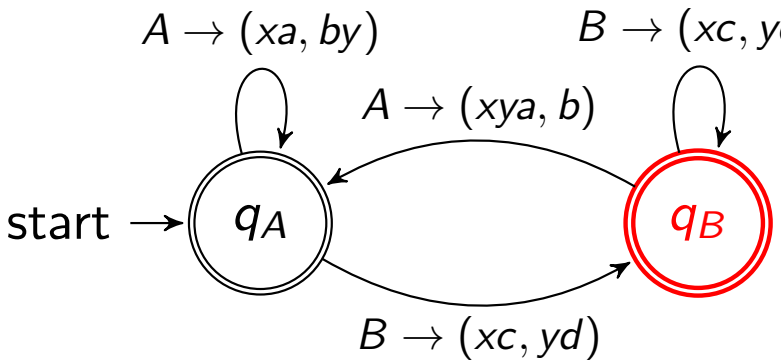


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

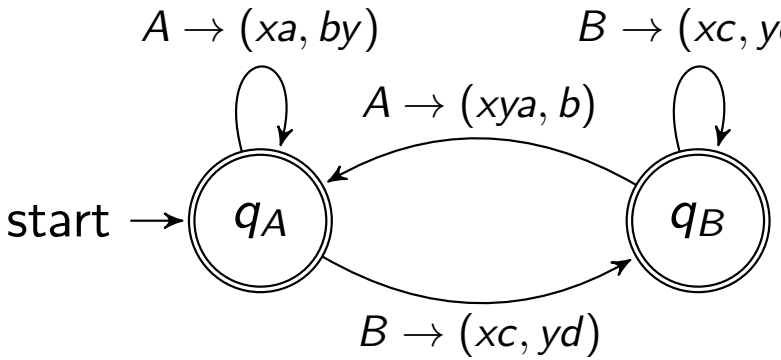


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

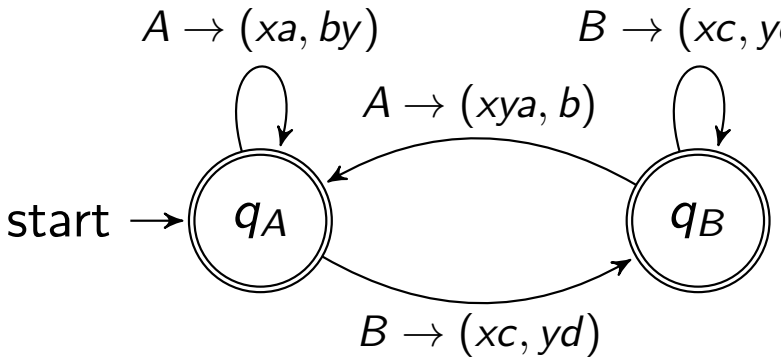


$F(q_A) = F(q_B) = xy$

Input	Opérations	
A	(xa, by)	•
B	(xc, yd)	
A	(xya, b)	
B	(xc, yd)	
	output(xy)	

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

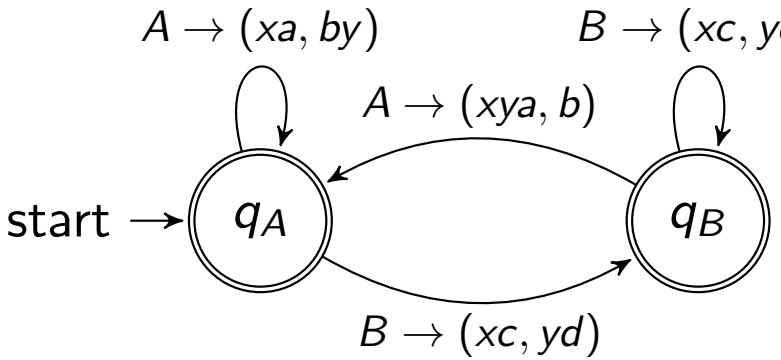


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

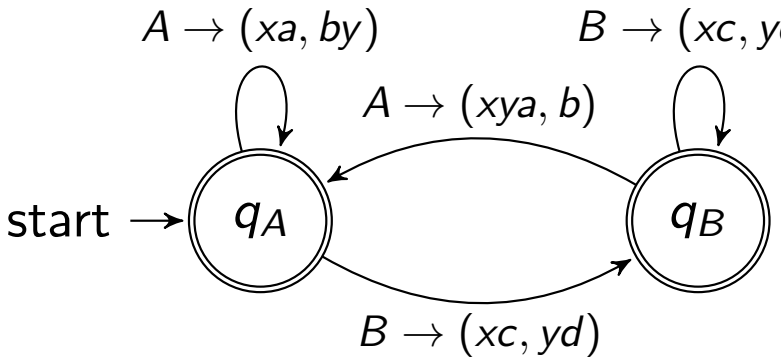


$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

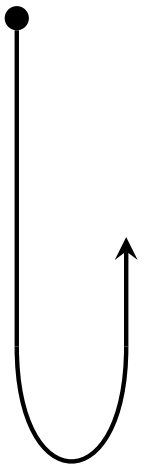
Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



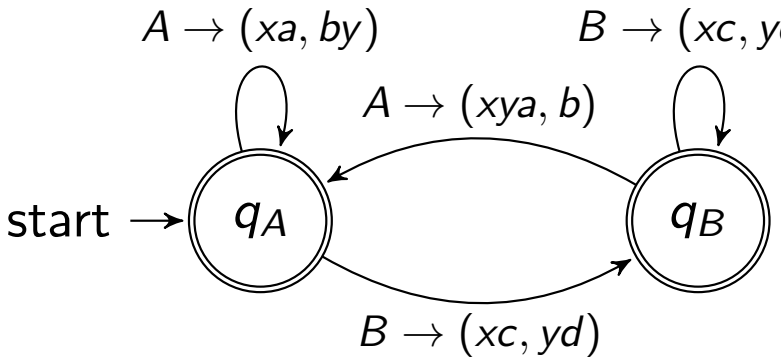
$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)



Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

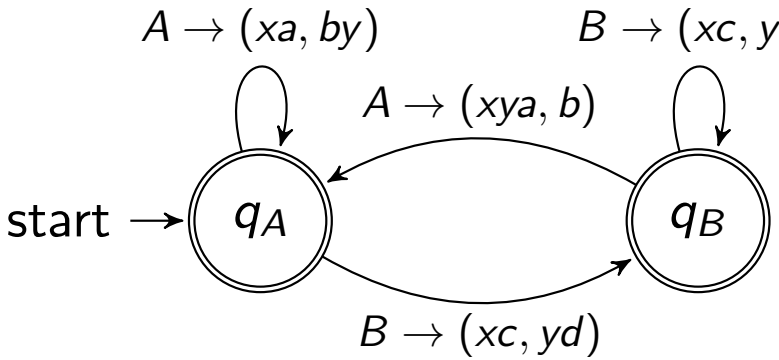


$F(q_A) = F(q_B) = xy$

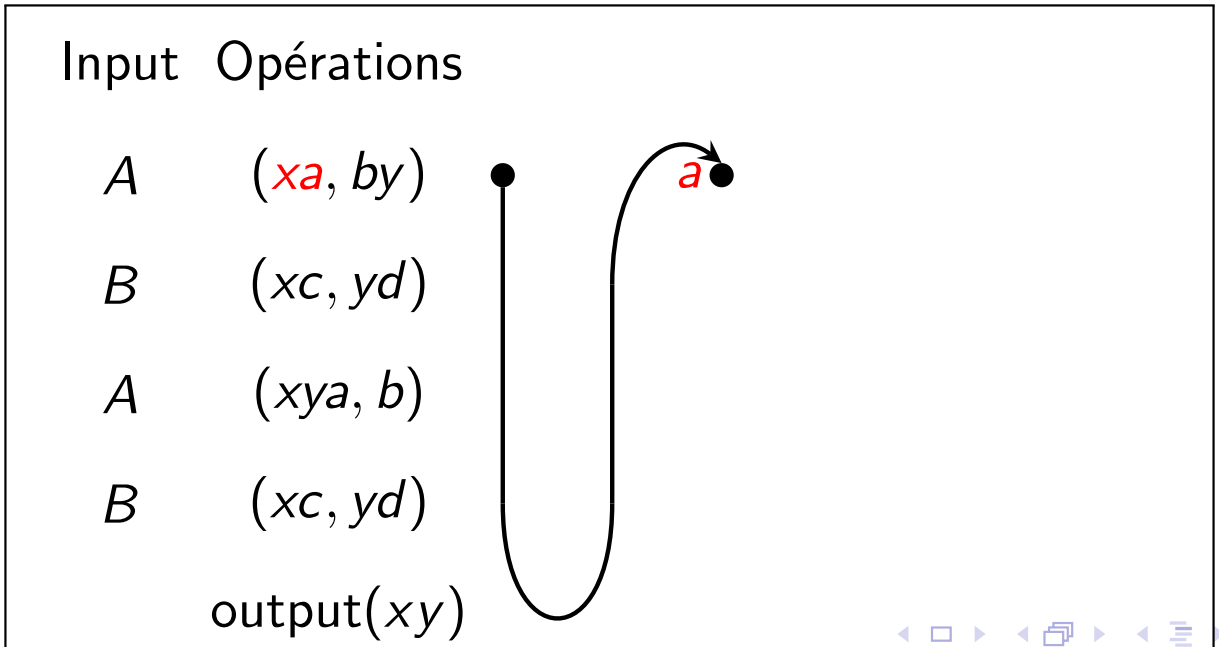
Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

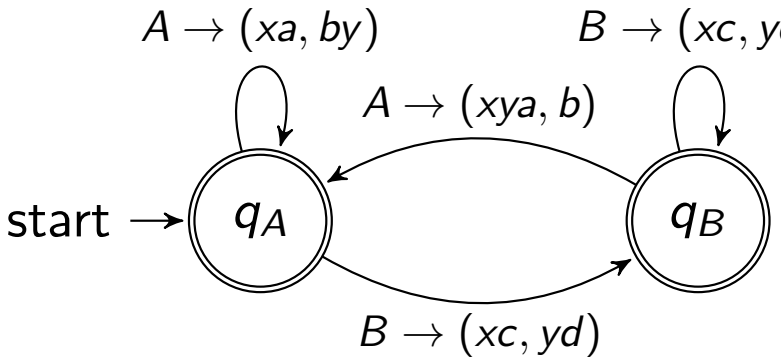


$F(q_A) = F(q_B) = xy$

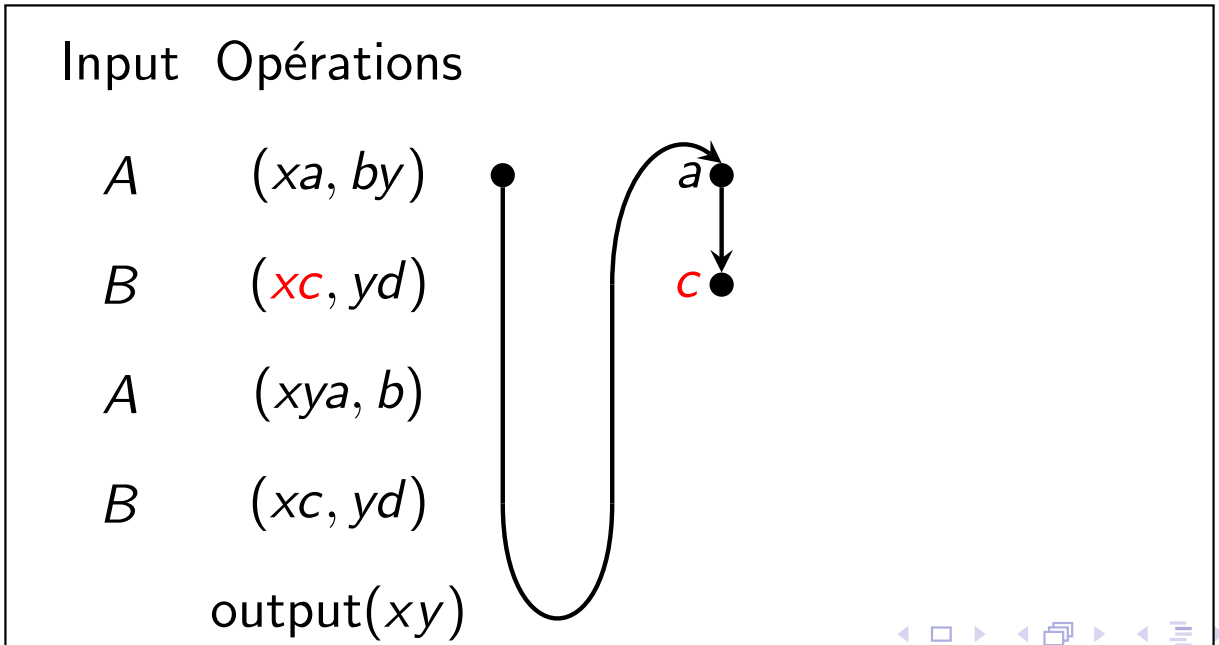


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

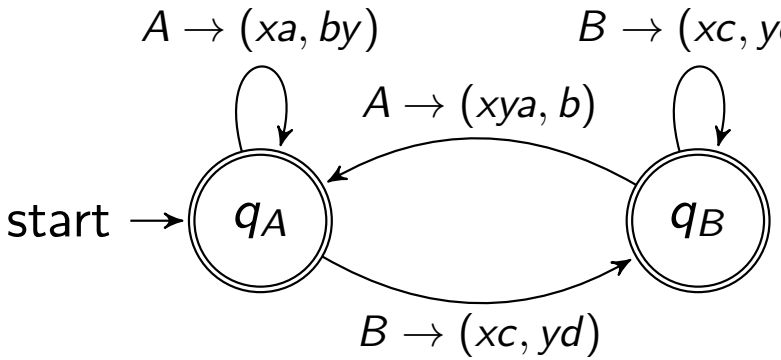


$F(q_A) = F(q_B) = xy$



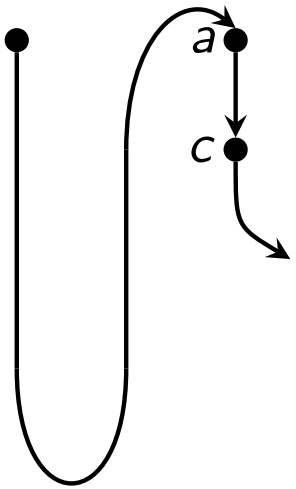
Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



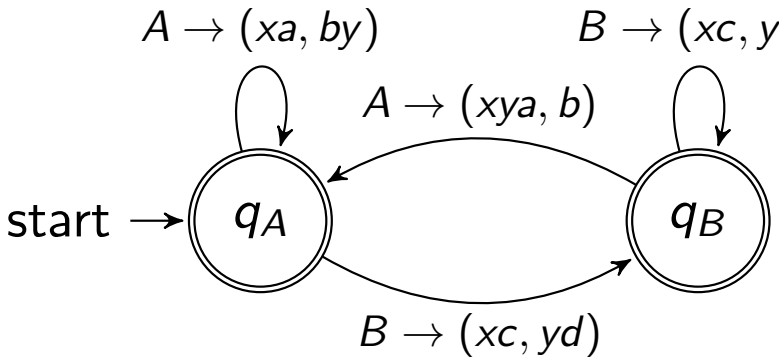
$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

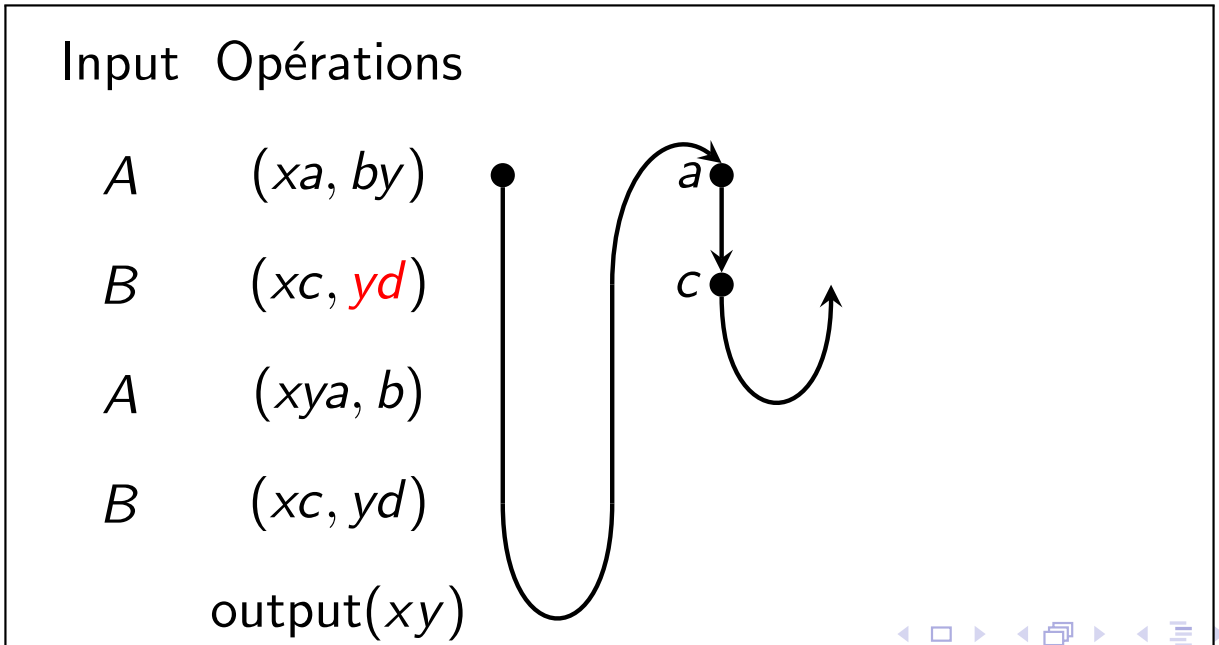


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

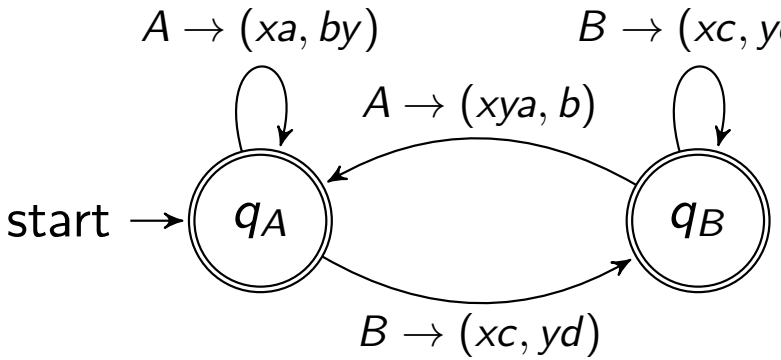


$F(q_A) = F(q_B) = xy$



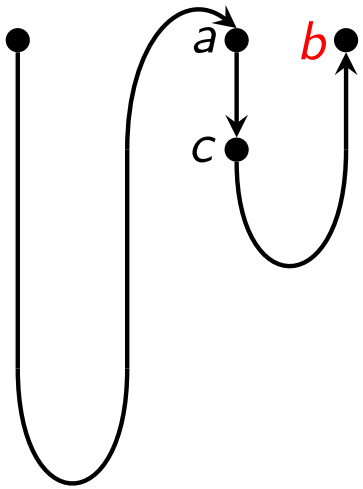
Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



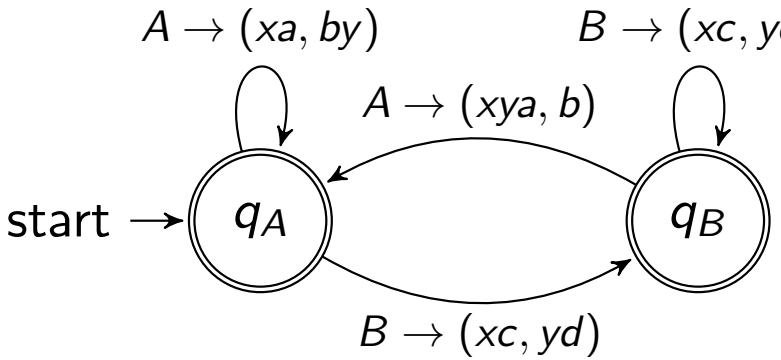
$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)



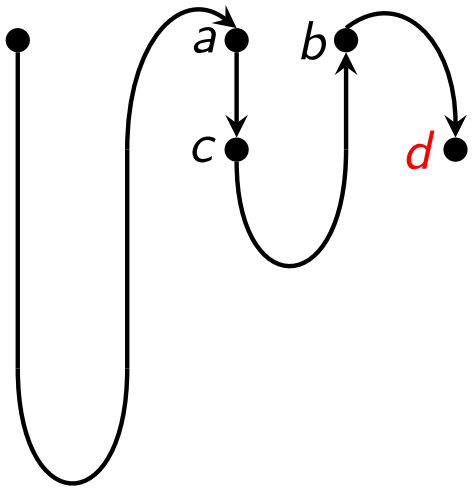
Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



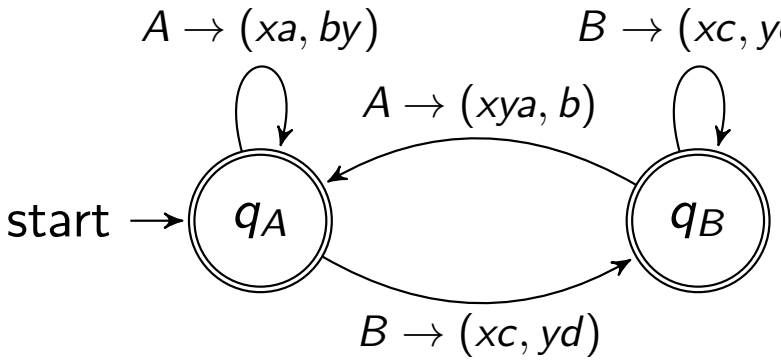
$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)



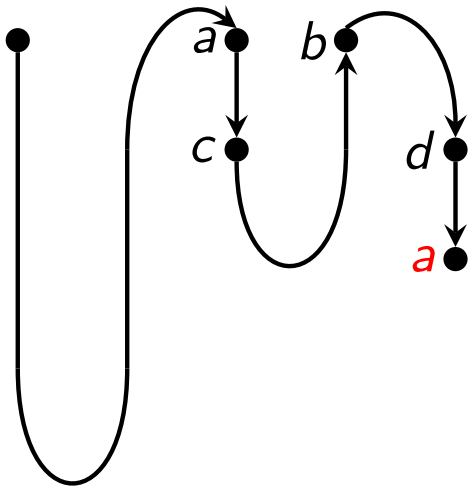
Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



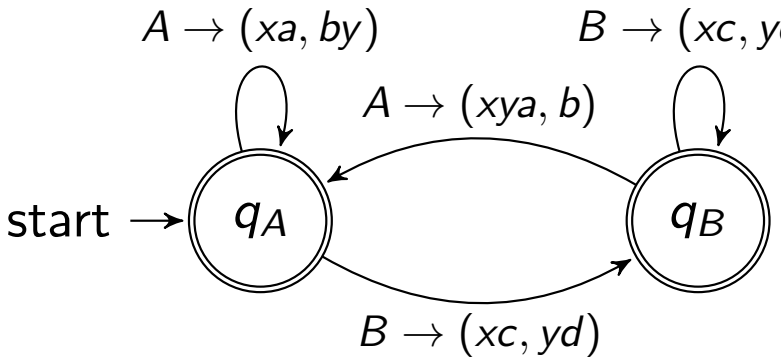
$F(q_A) = F(q_B) = xy$

Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

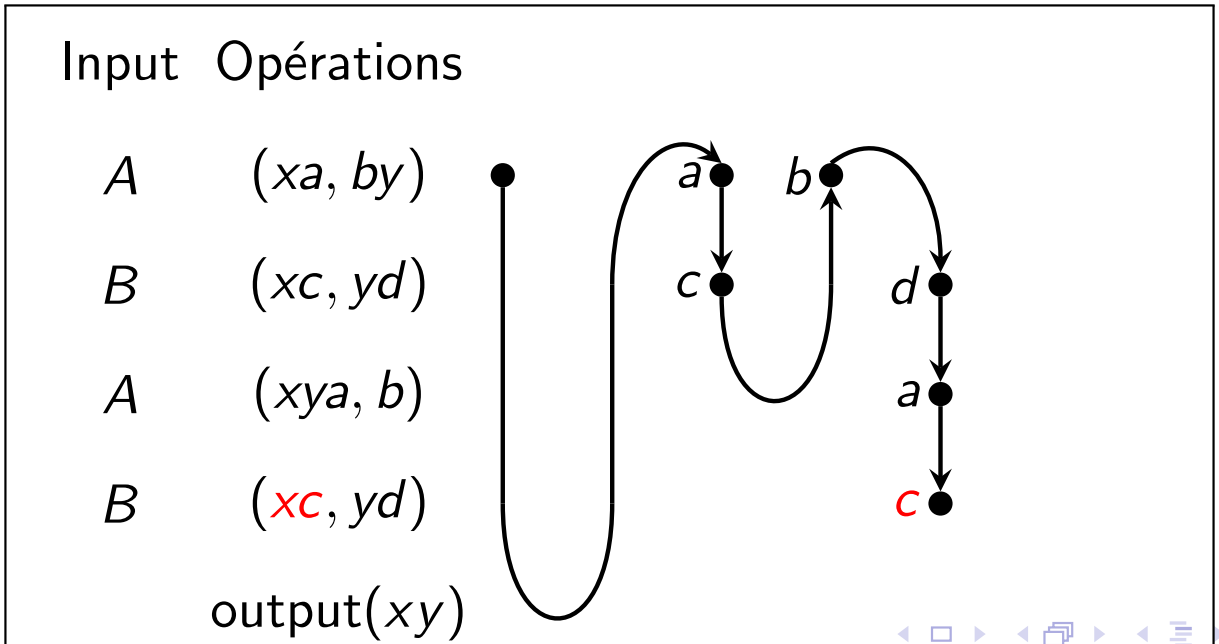


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

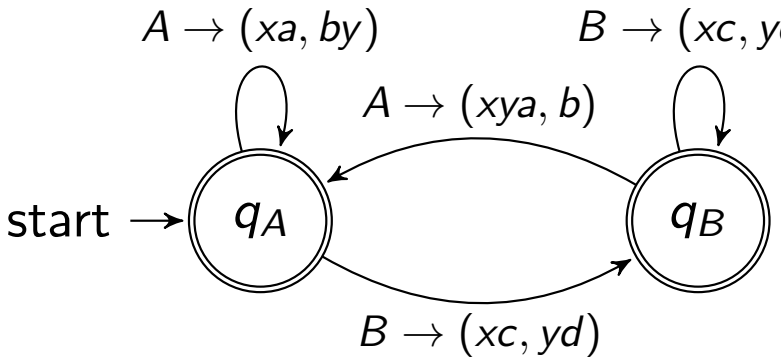


$F(q_A) = F(q_B) = xy$

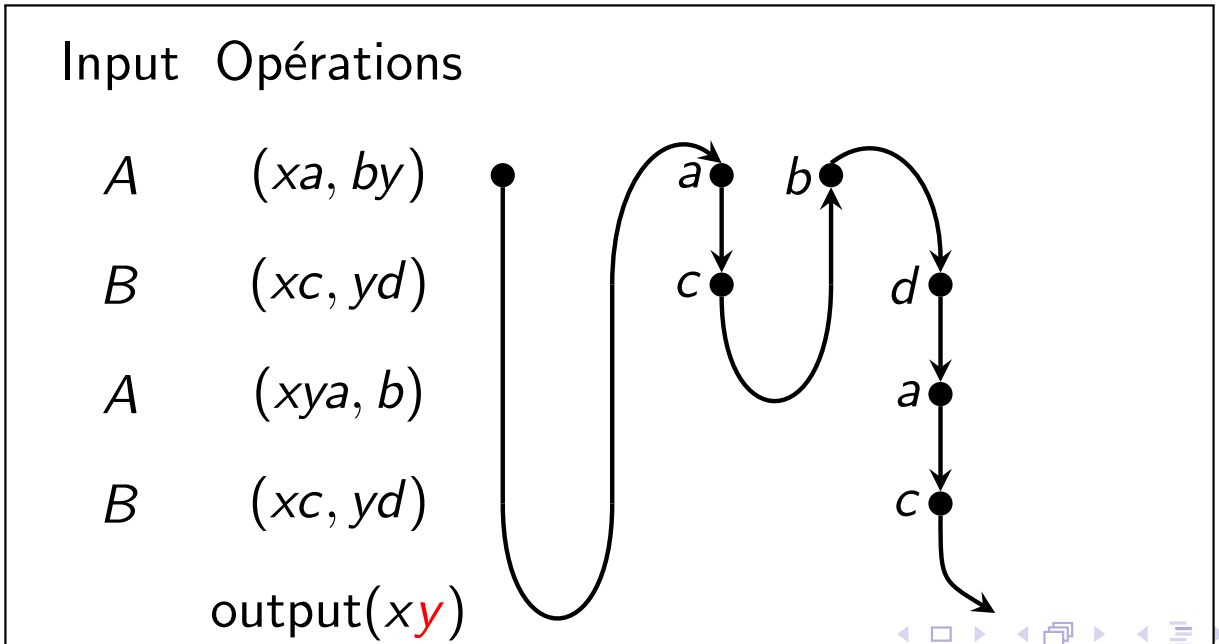


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

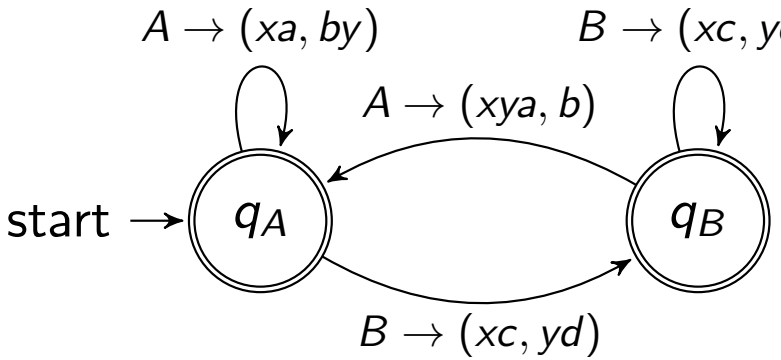


$F(q_A) = F(q_B) = xy$

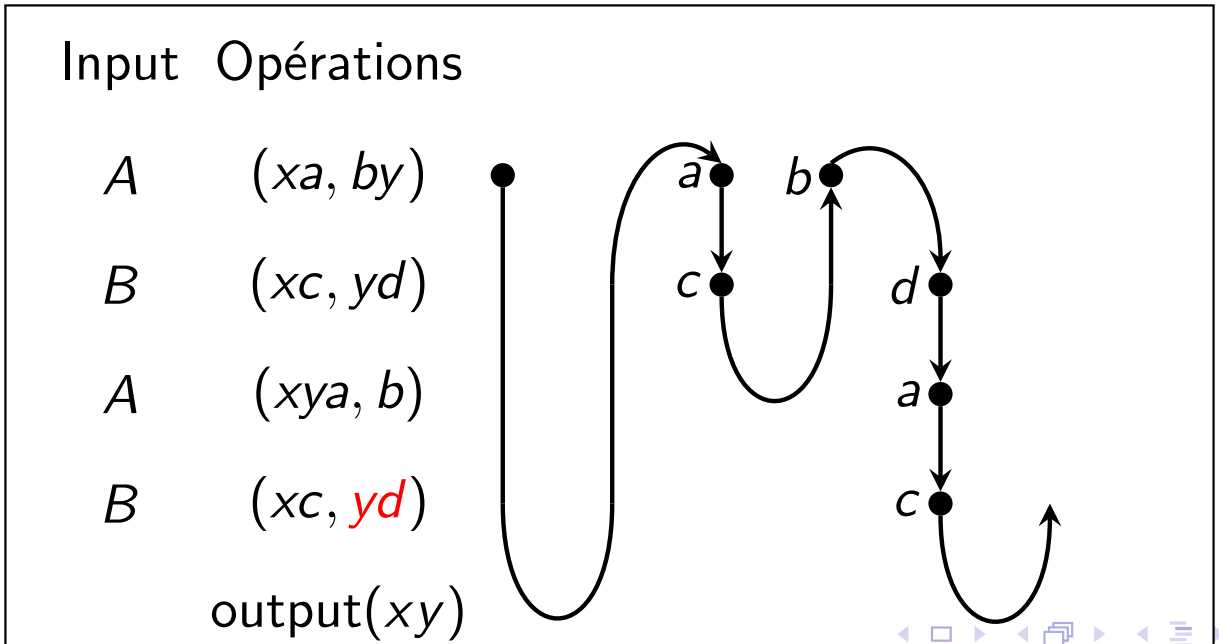


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

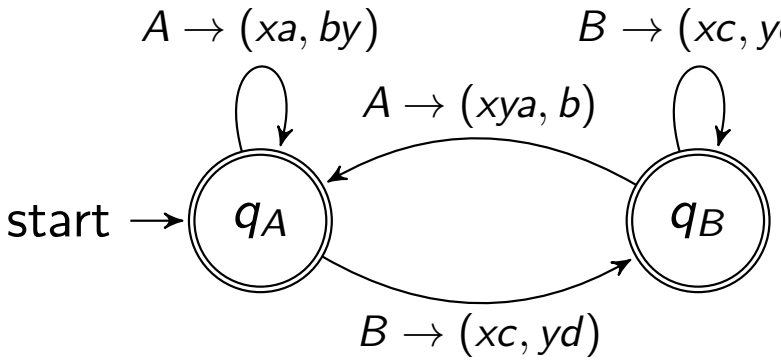


$F(q_A) = F(q_B) = xy$

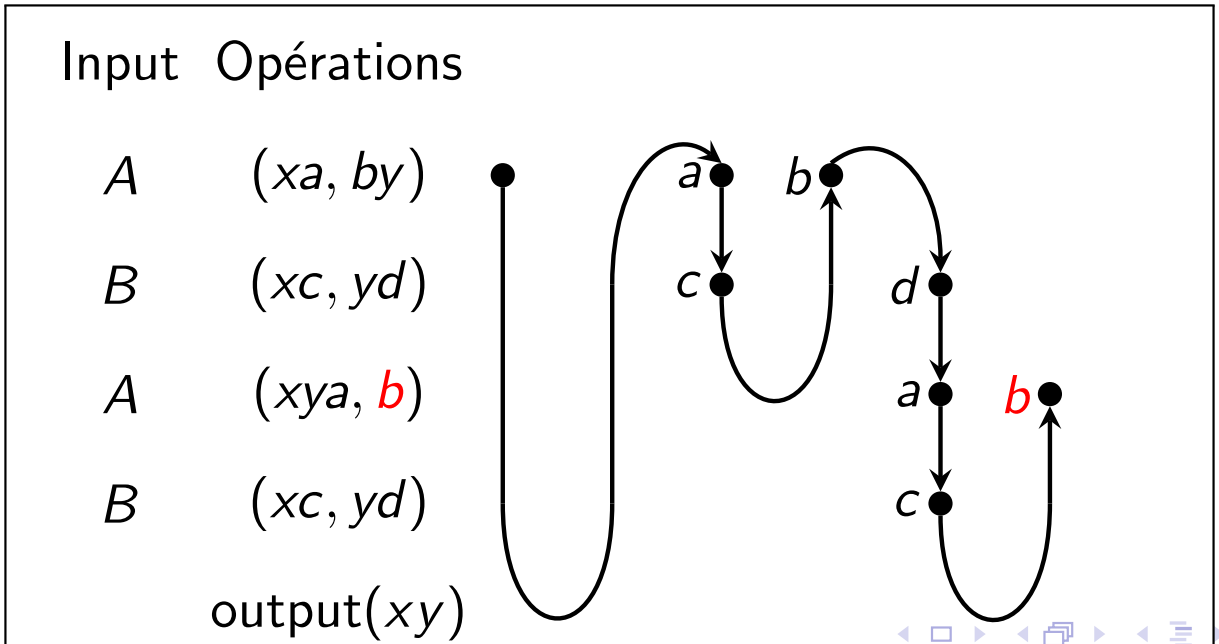


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

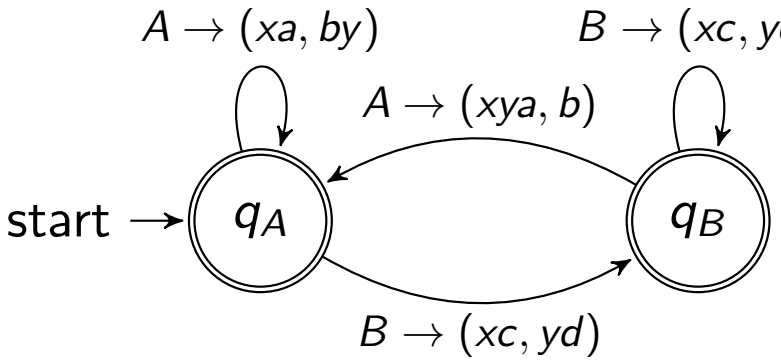


$F(q_A) = F(q_B) = xy$

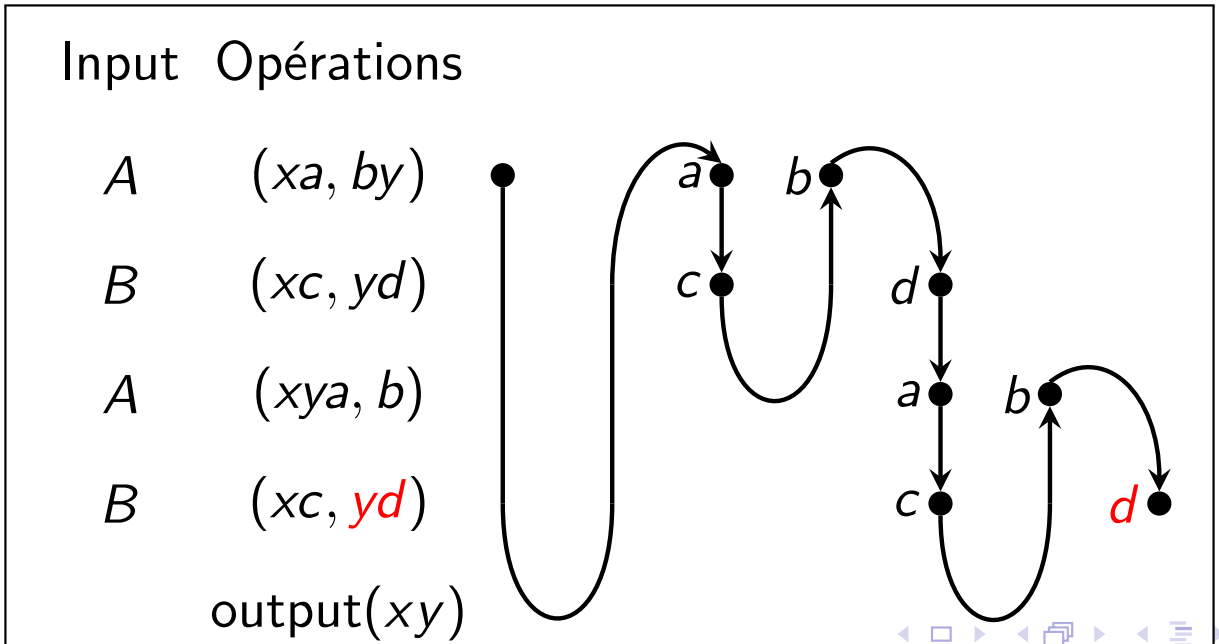


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$

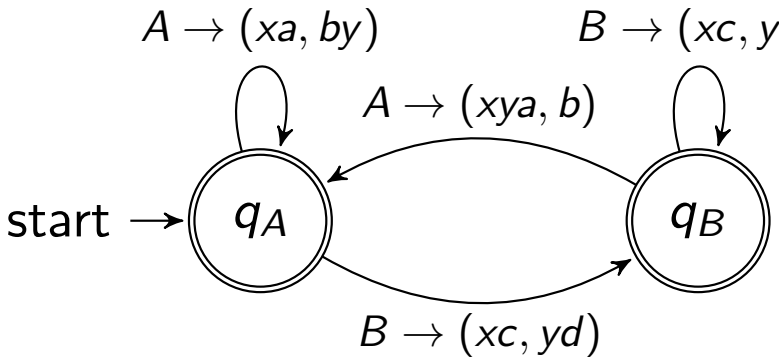


$F(q_A) = F(q_B) = xy$

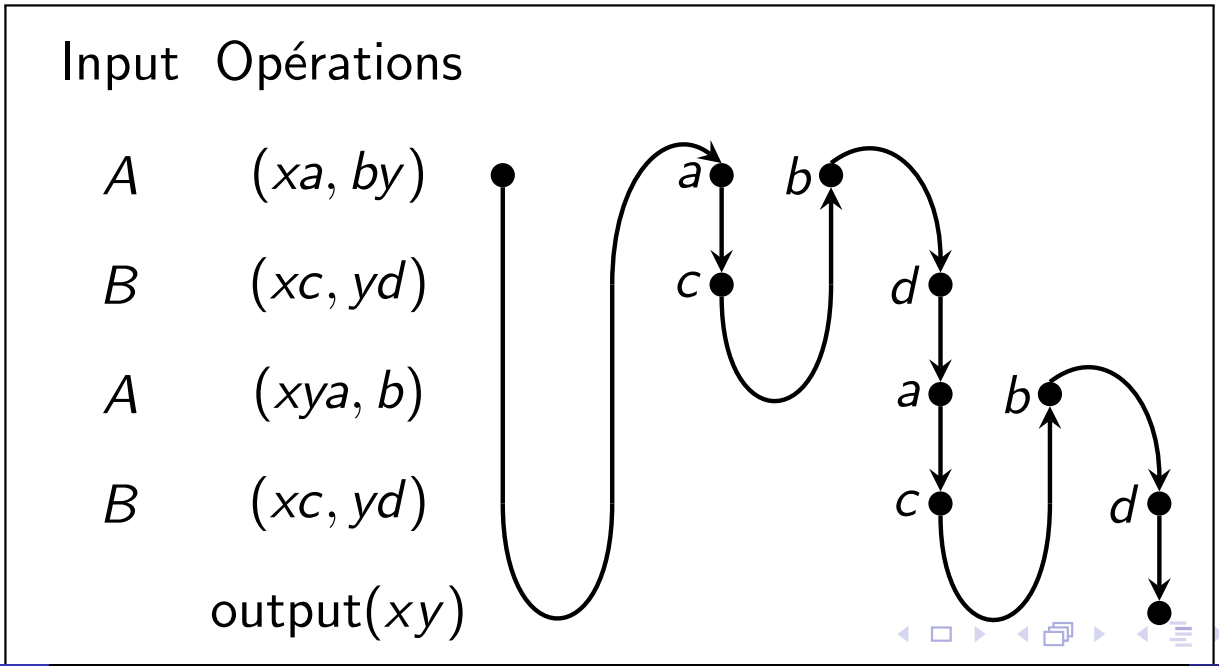


Idée générale

Exemple : La transduction $(A^n B^p)^k \to (a^n c^p b^n d^p)^k$



$F(q_A) = F(q_B) = xy$



Preuve de correction

$\vdash$

(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy

Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction

$\vdash$

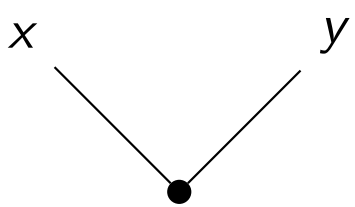
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction

$\vdash$

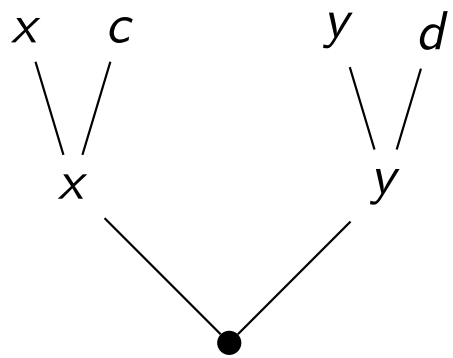
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction

$\vdash$

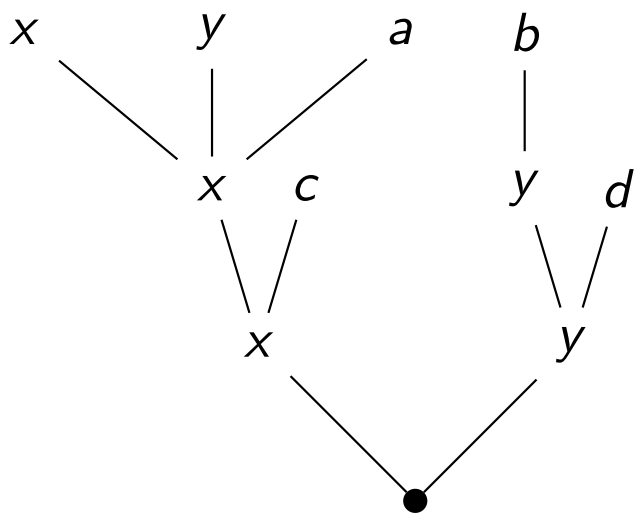
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction

$\vdash$

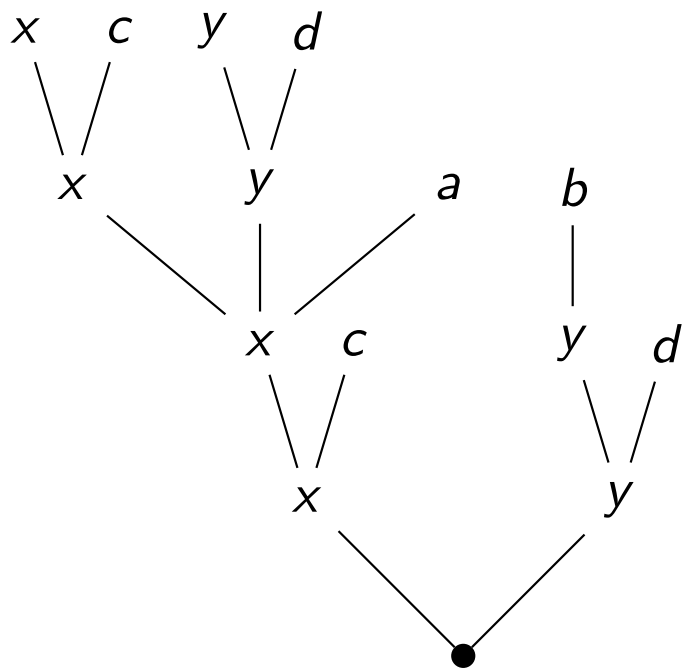
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction

$\vdash$

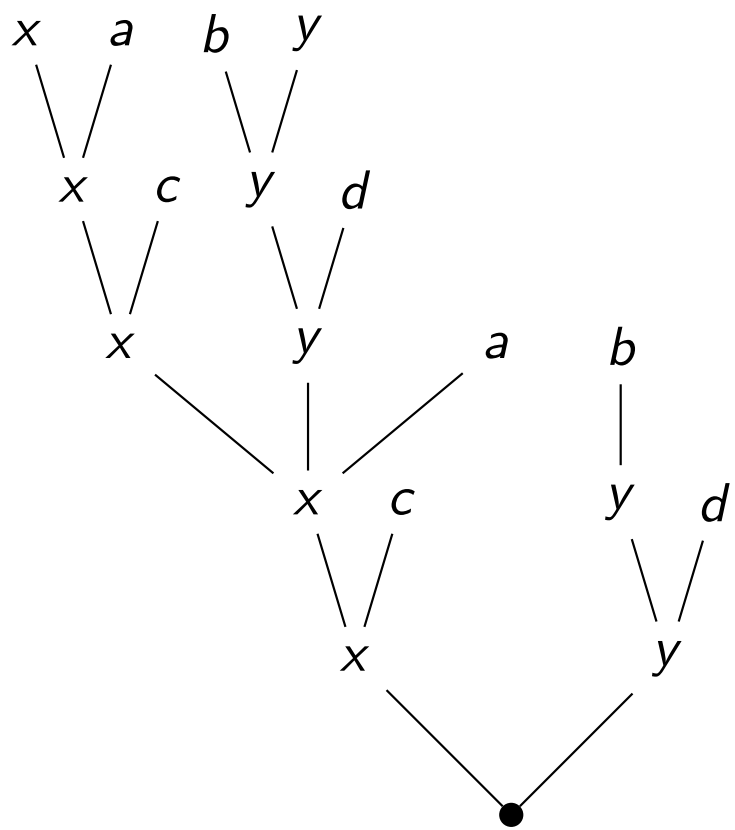
(xa, by)

(xc, yd)

(xya, b)

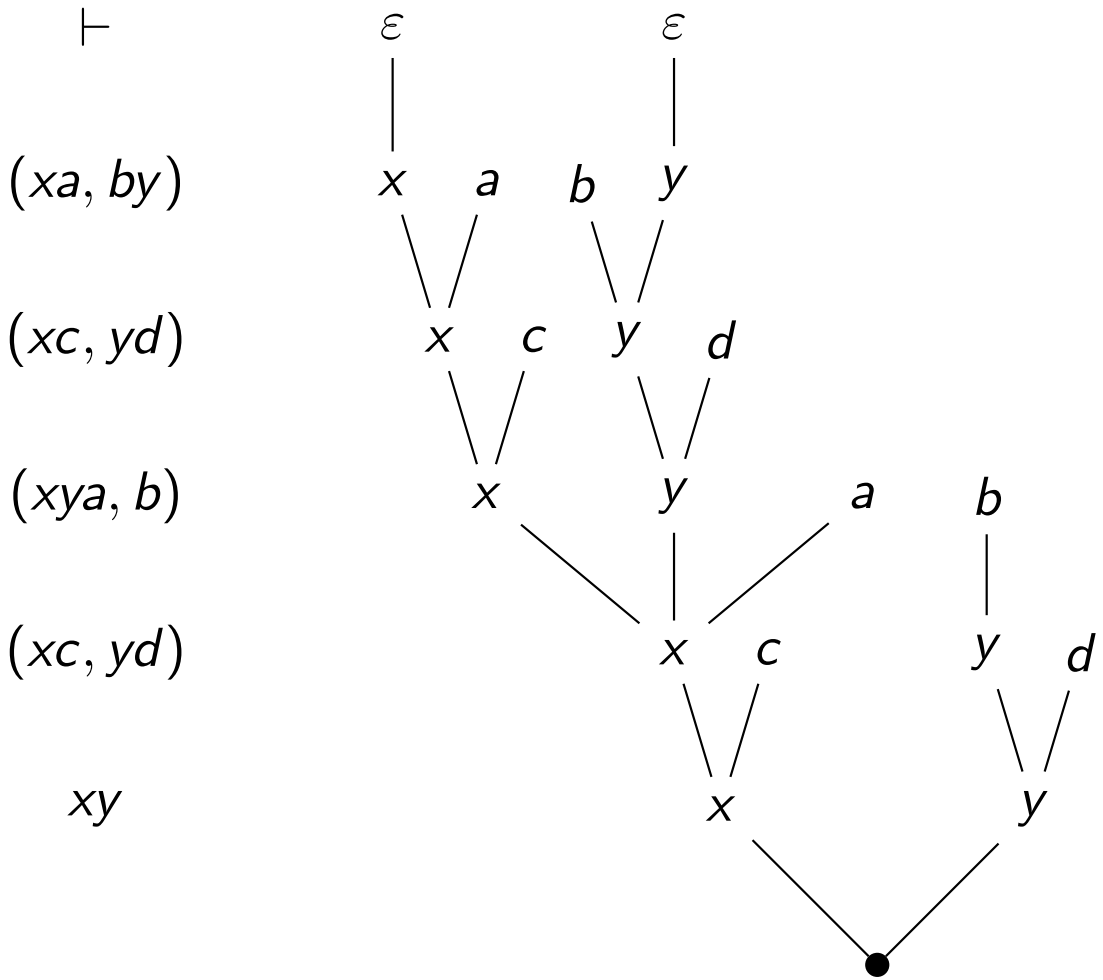
(xc, yd)

xy



Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction



Idée : remplacer petit à petit les valeurs de x et y

Preuve de correction

$\vdash$

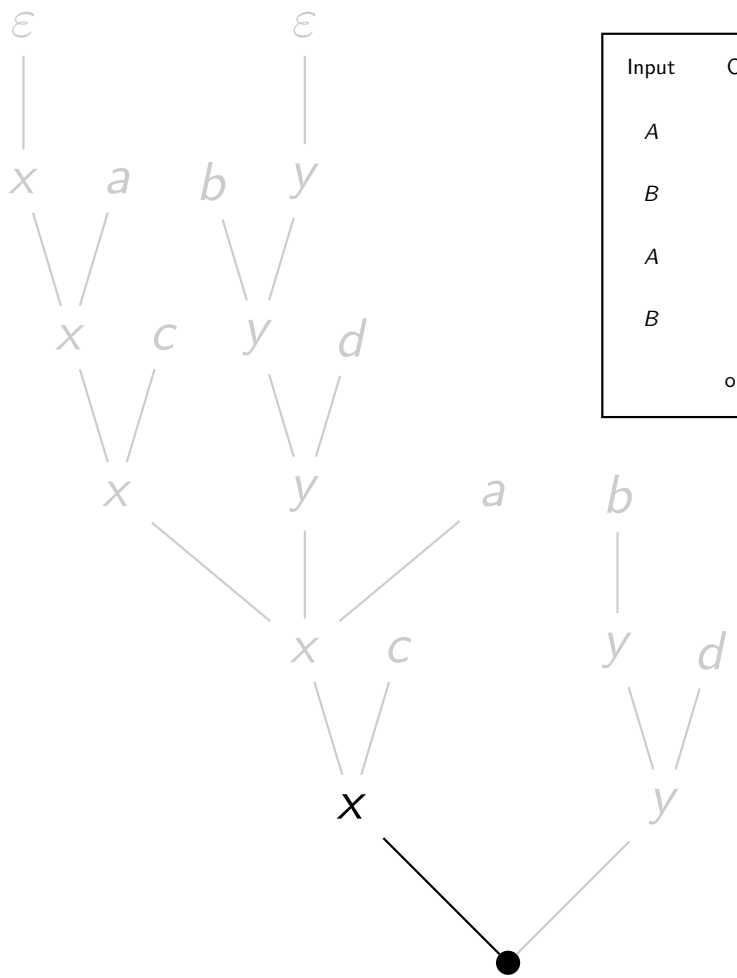
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Résultat :

Preuve de correction

$\vdash$

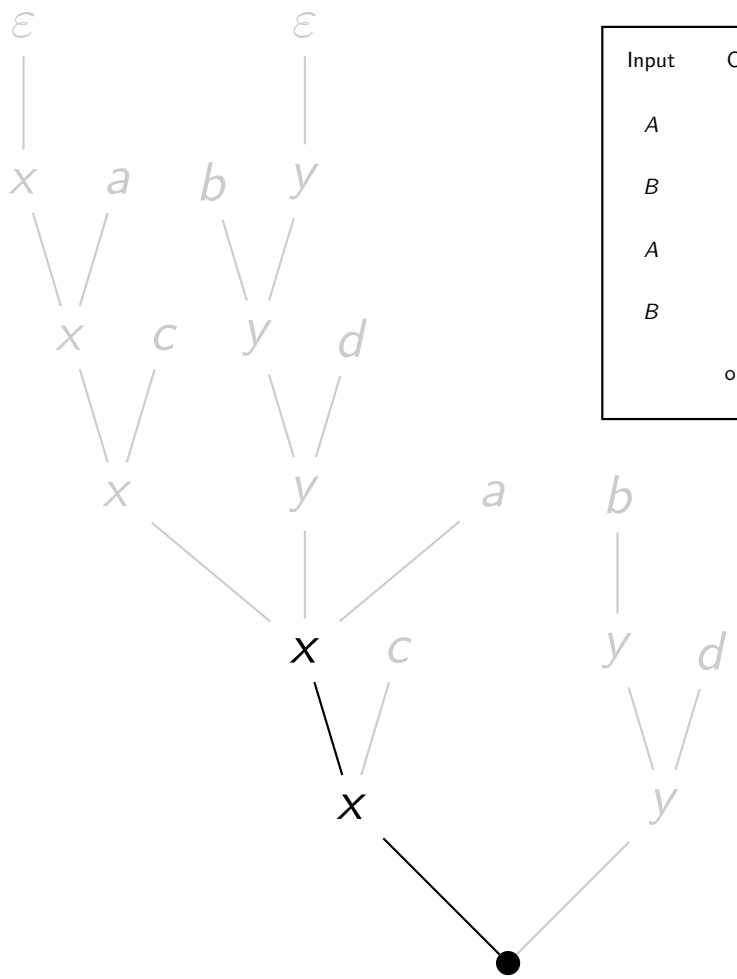
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Résultat :

Preuve de correction

$\vdash$

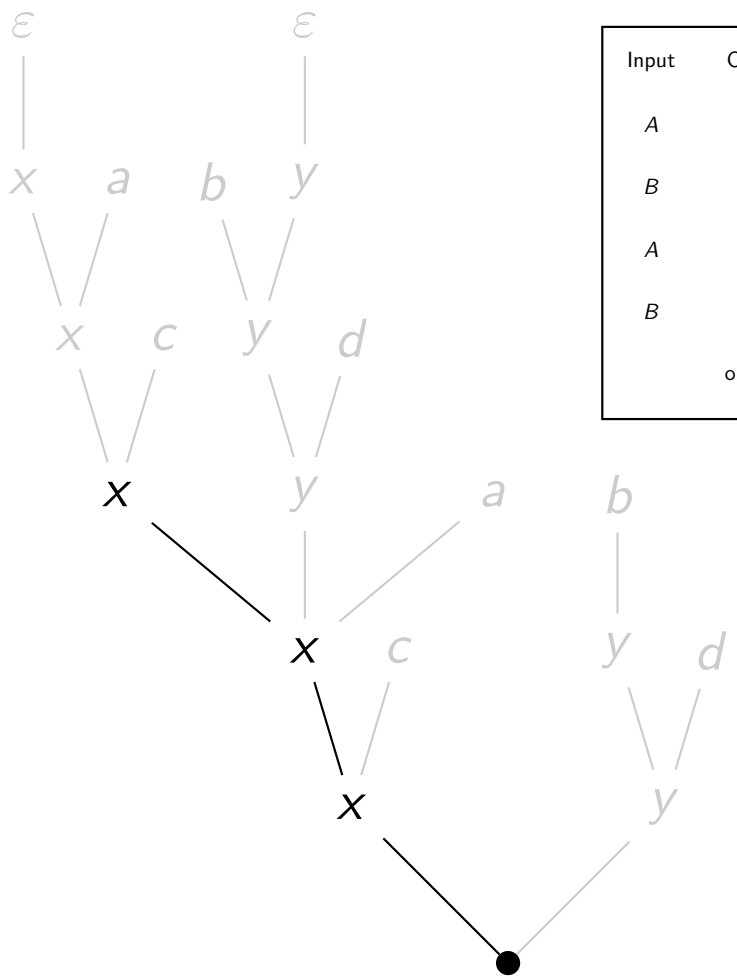
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Résultat :

Preuve de correction

$\vdash$

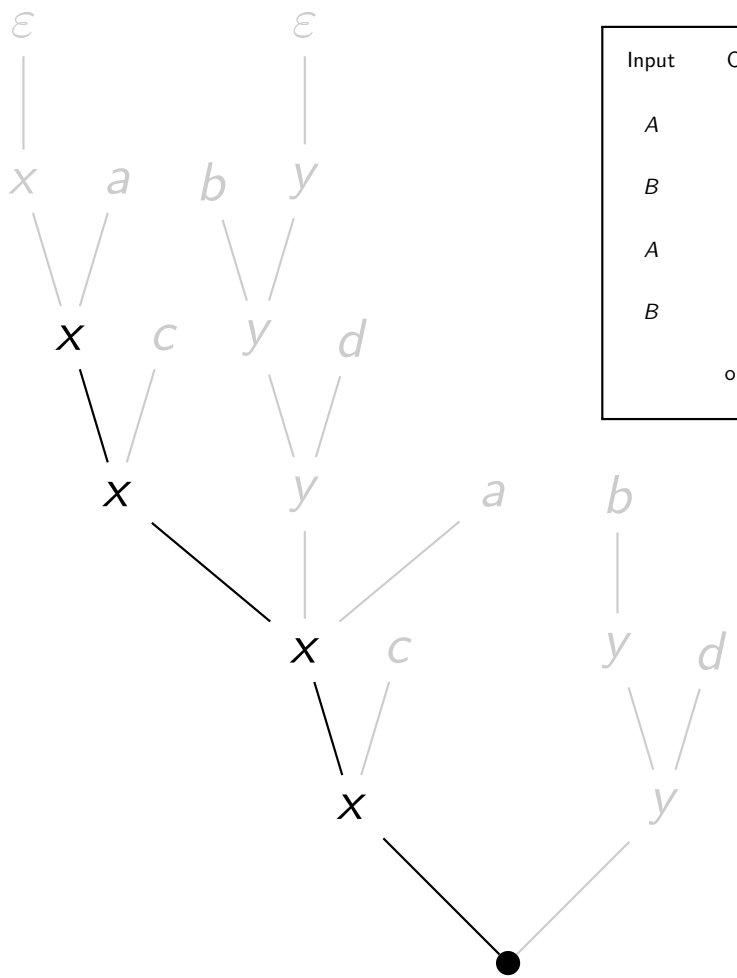
(xa, by)

(xc, yd)

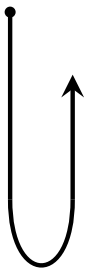
(xya, b)

(xc, yd)

xy



Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)



Résultat :

Preuve de correction

$\vdash$

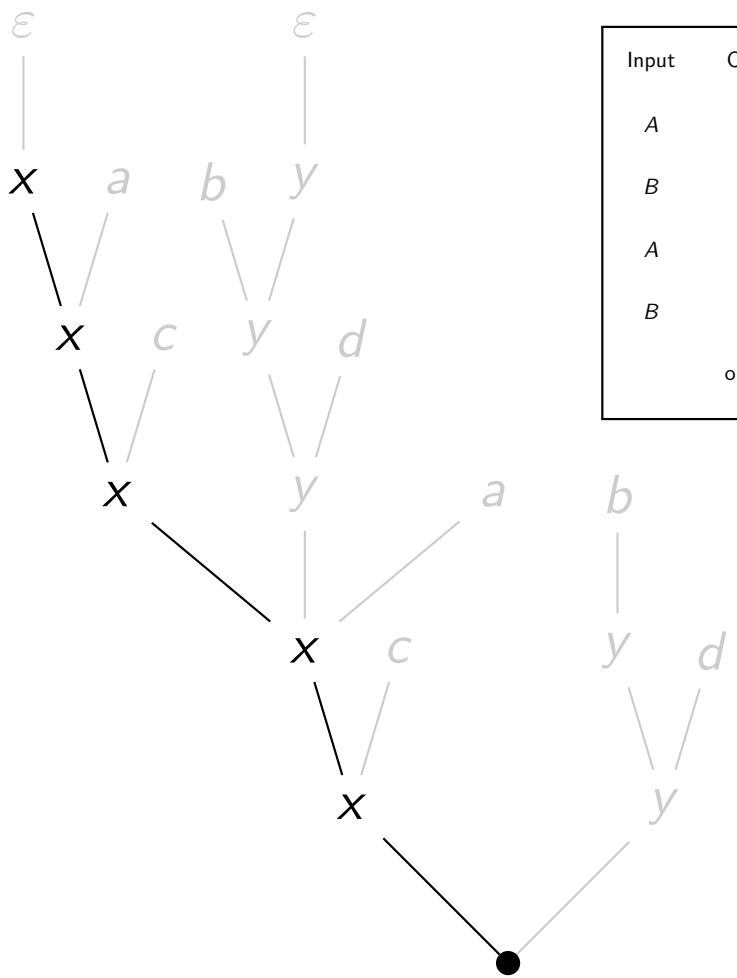
(xa, by)

(xc, yd)

(xya, b)

(xc, yd)

xy



Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Résultat :

Preuve de correction

$\vdash$ ε ε

(xa, by) x a b y

(xc, yd) x c y d

(xya, b) x y a b

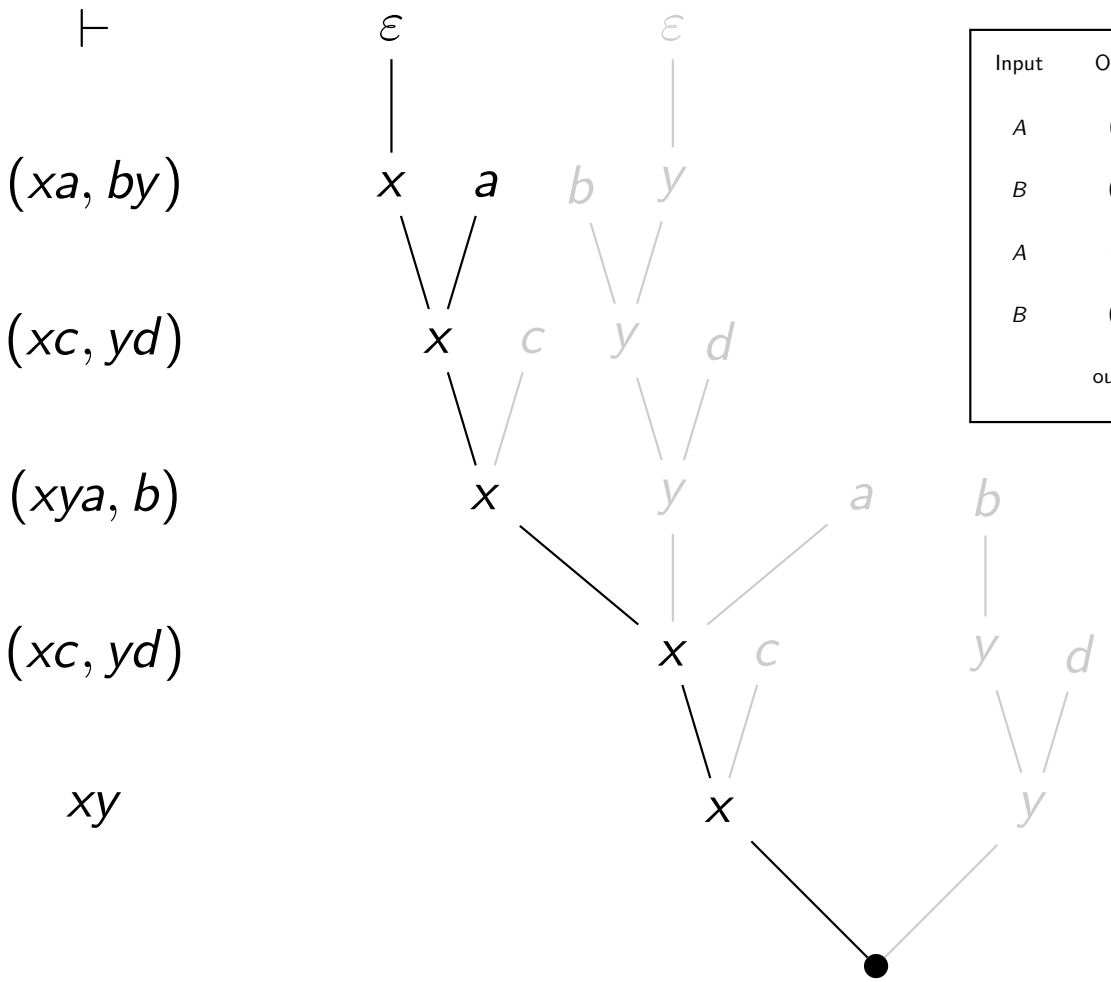
(xc, yd) x c y d

xy x y

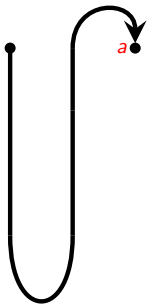
Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

Résultat : ε

Preuve de correction

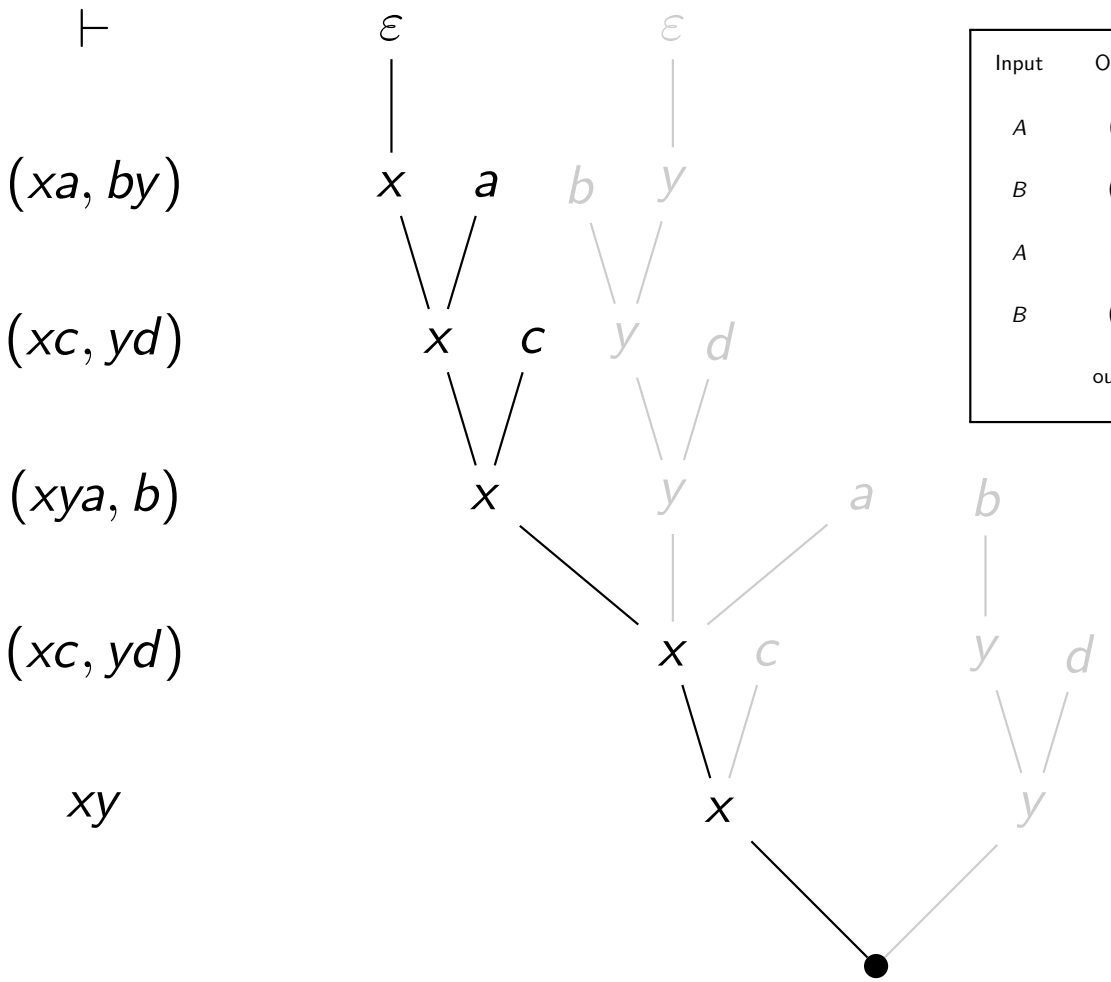


Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

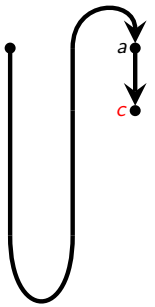


Résultat : εa

Preuve de correction

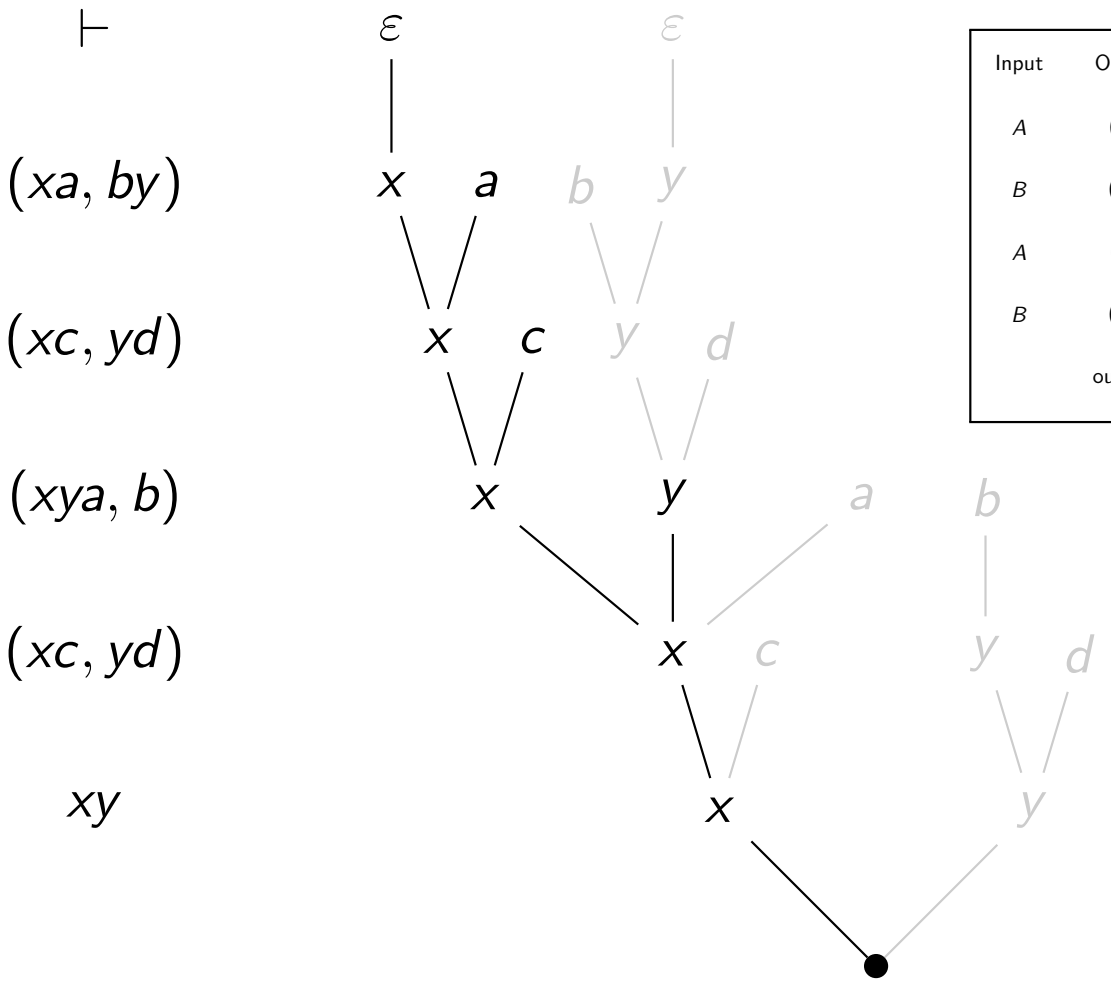


Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)

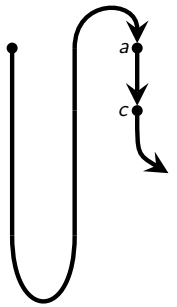


Résultat : εac

Preuve de correction

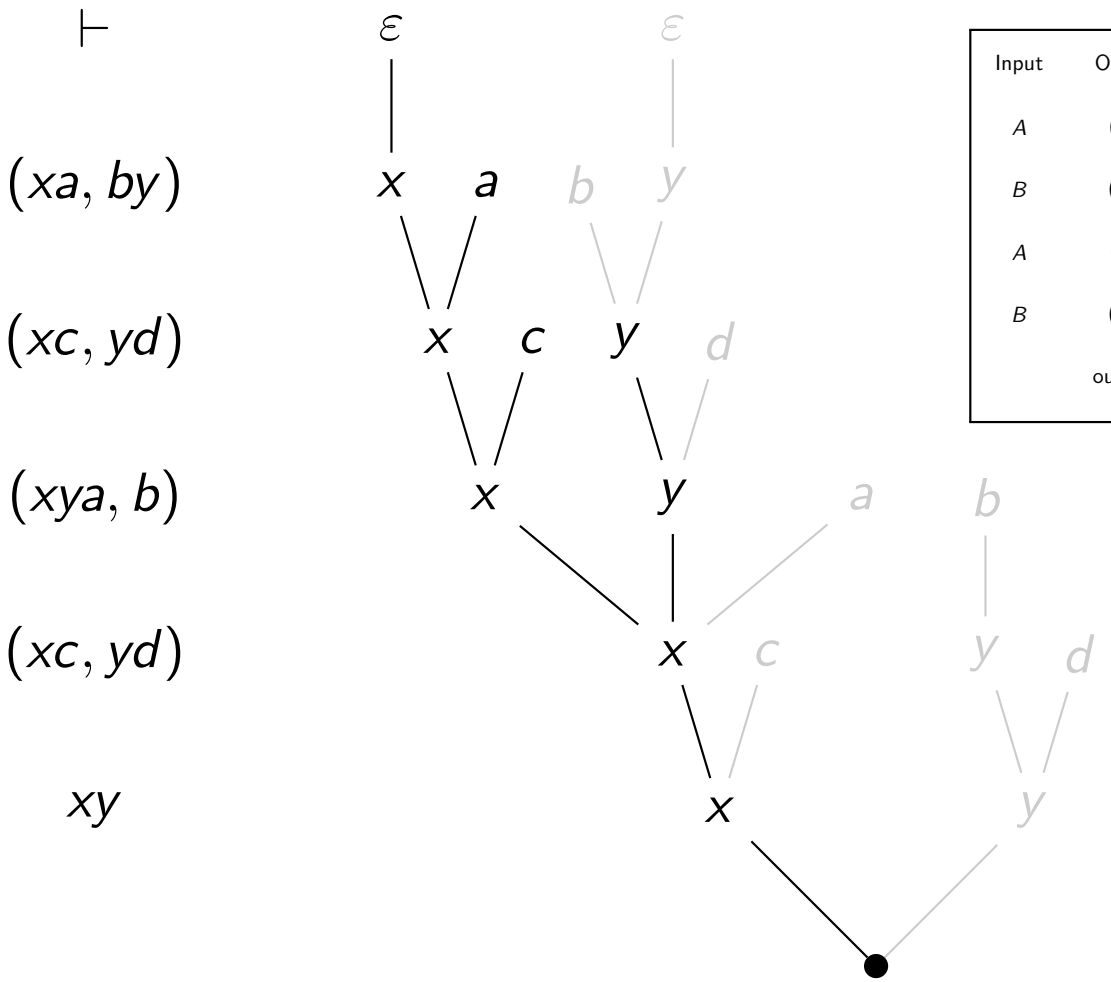


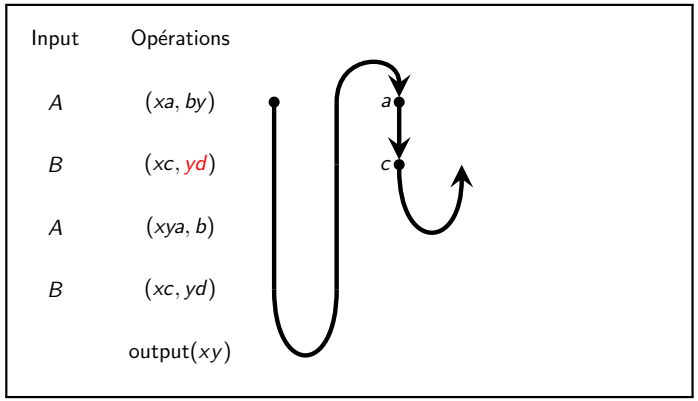
Input	Opérations
A	(xa, by)
B	(xc, yd)
A	(xya, b)
B	(xc, yd)
	output(xy)



Résultat : εac

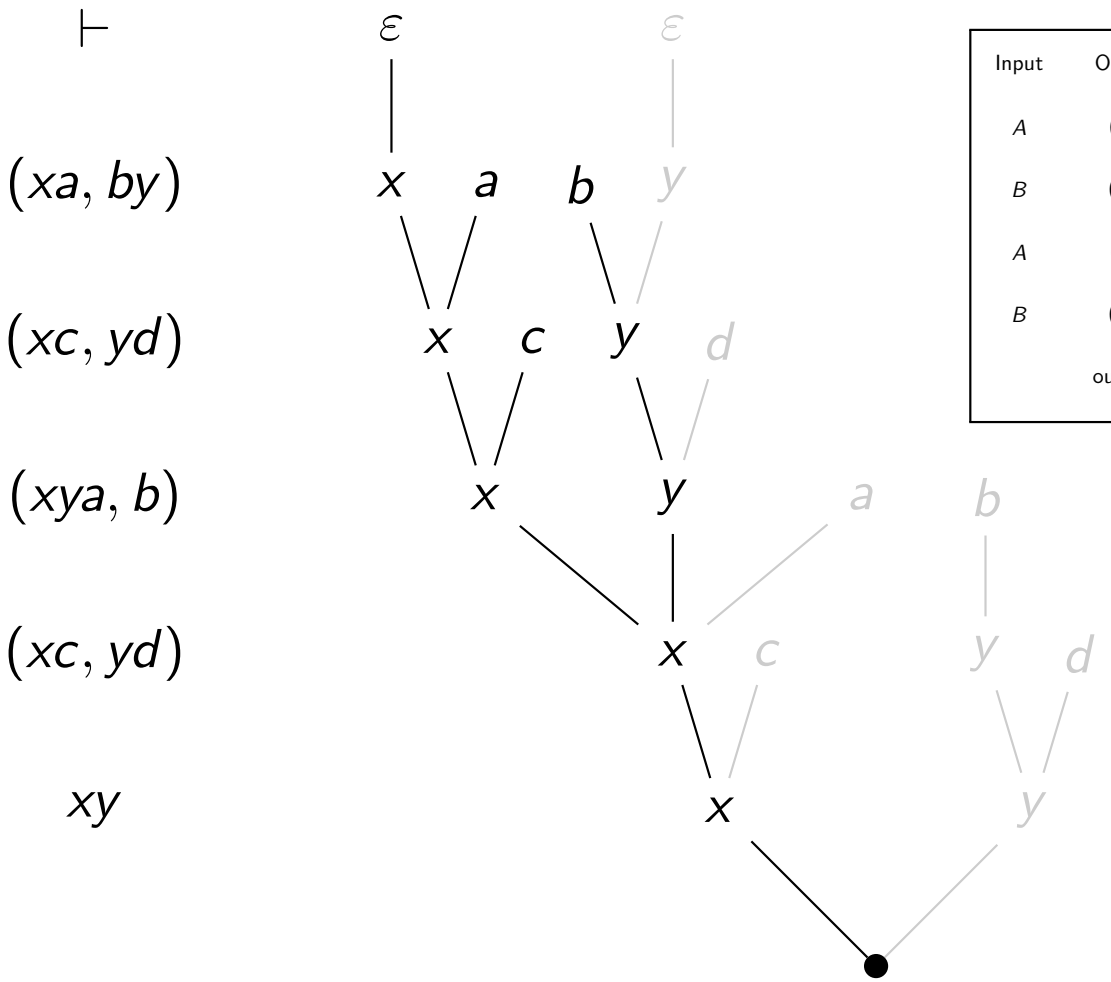
Preuve de correction

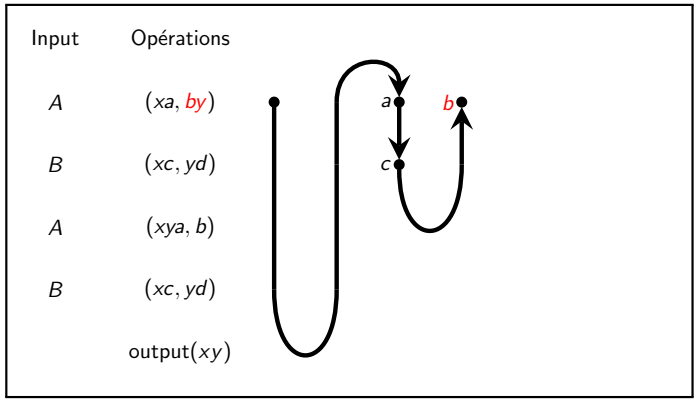




Résultat : εac

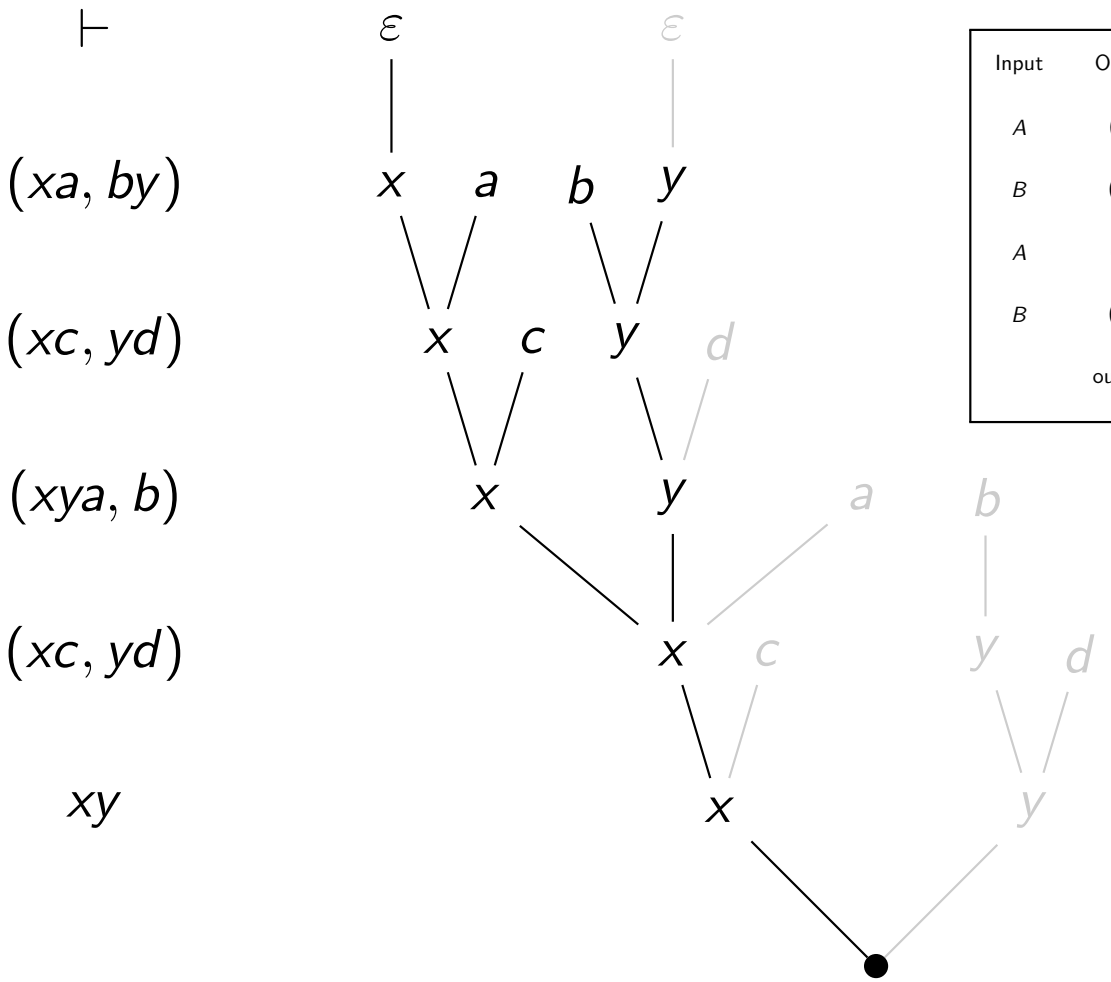
Preuve de correction

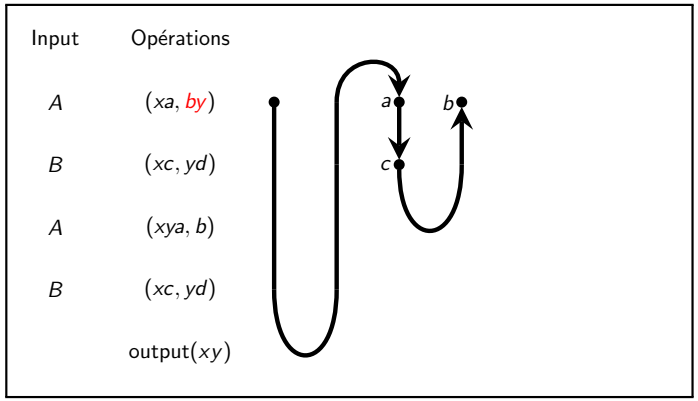




Résultat : εacb

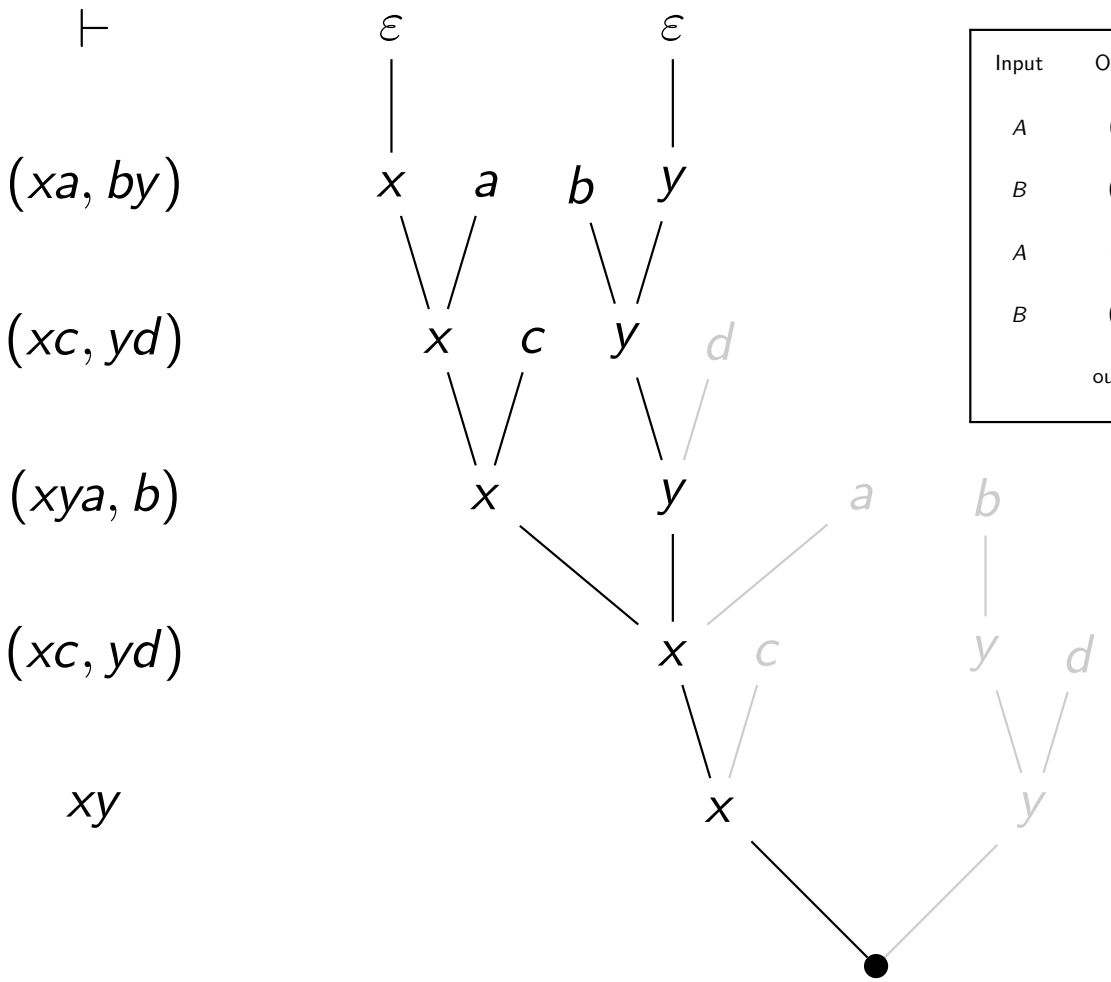
Preuve de correction

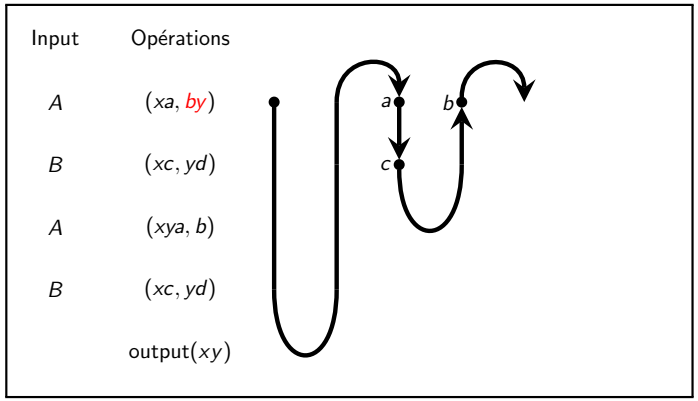




Résultat : εacb

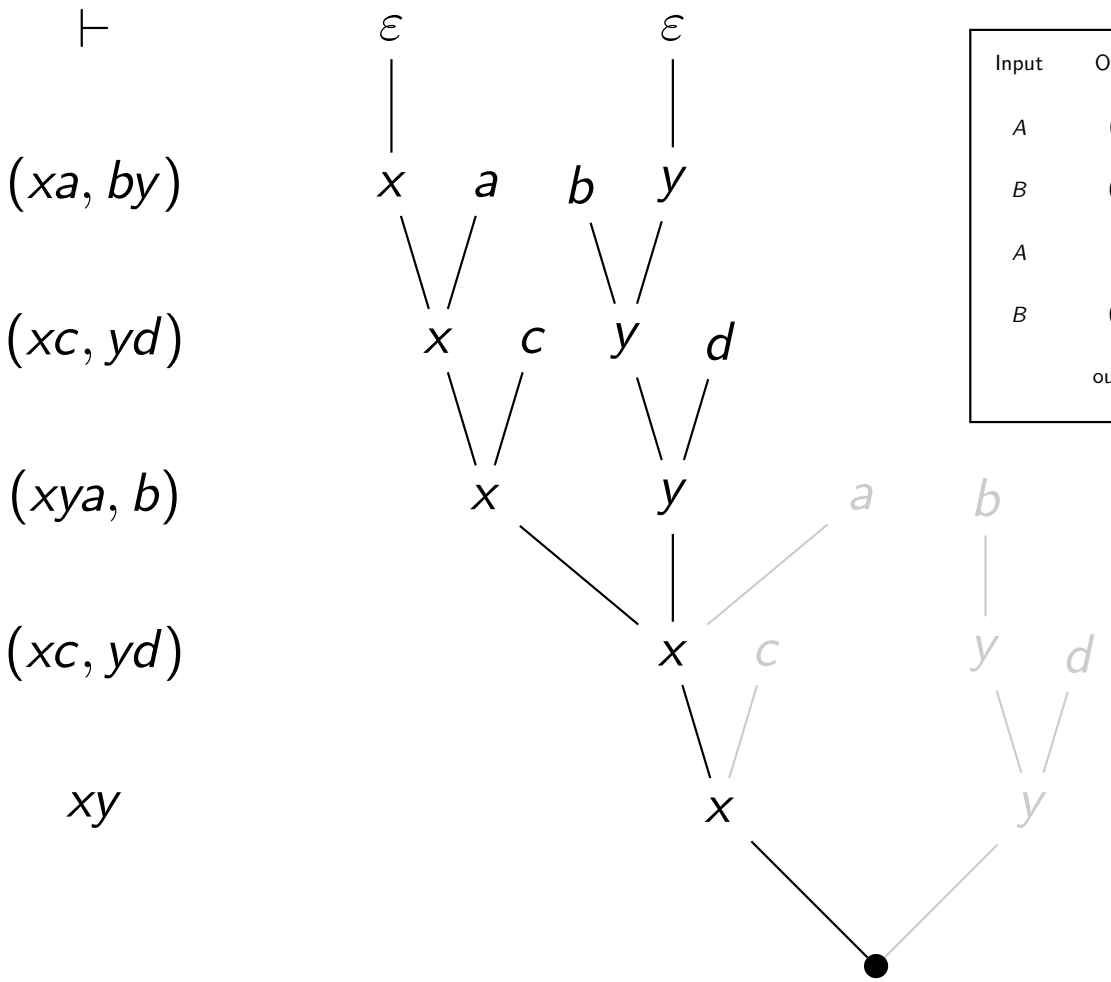
Preuve de correction

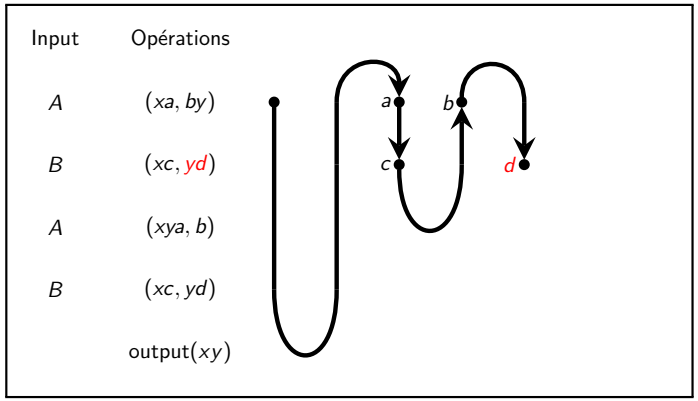




Résultat : $\varepsilon acb\varepsilon$

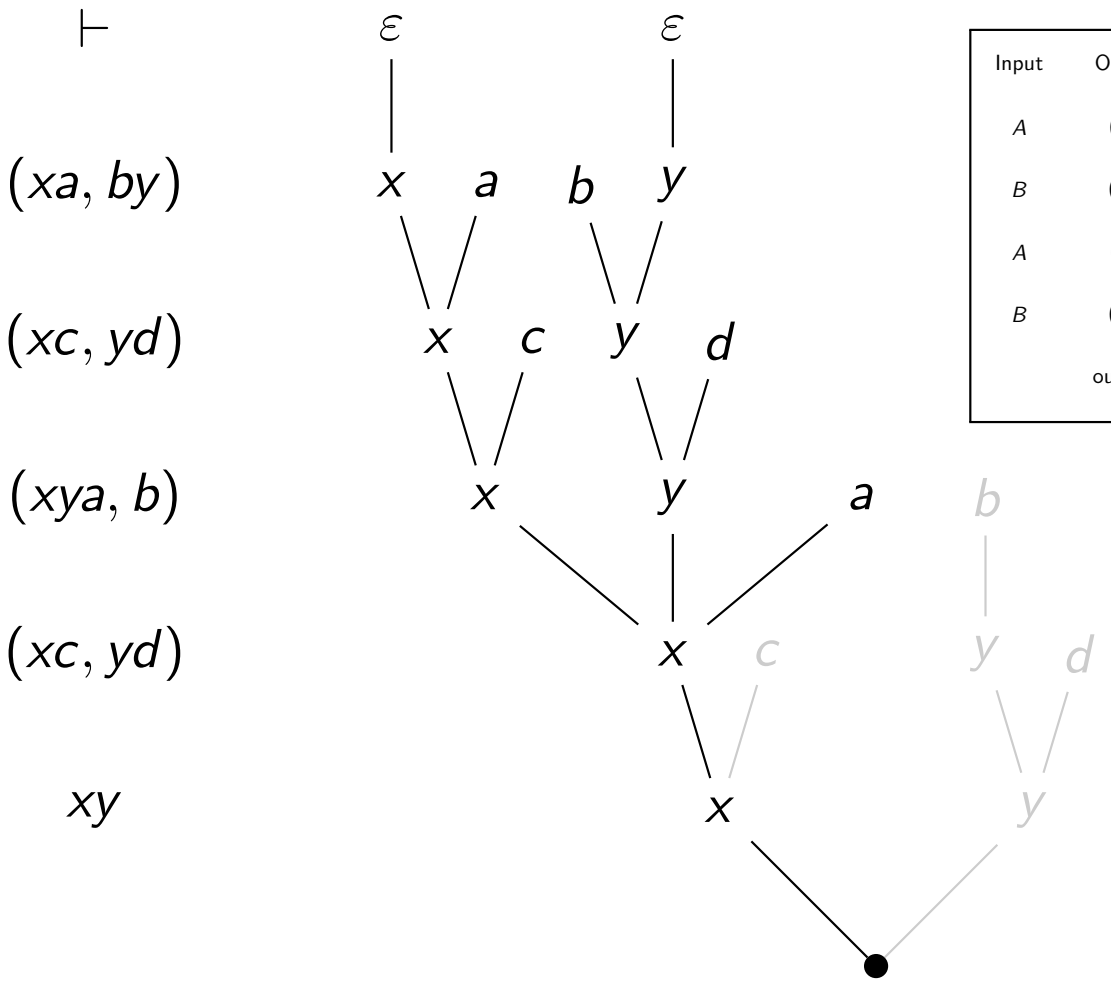
Preuve de correction

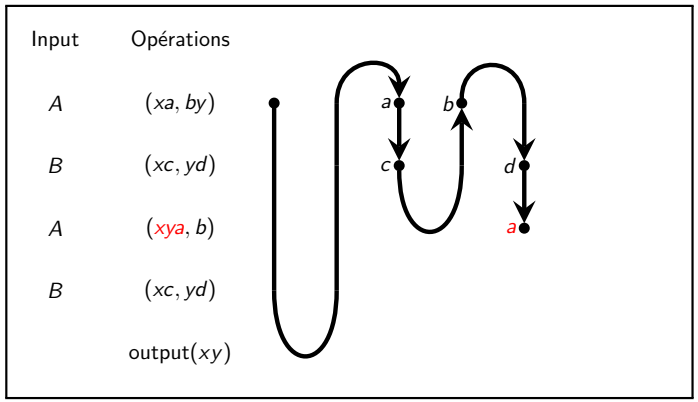




Résultat : $\varepsilon acb\varepsilon d$

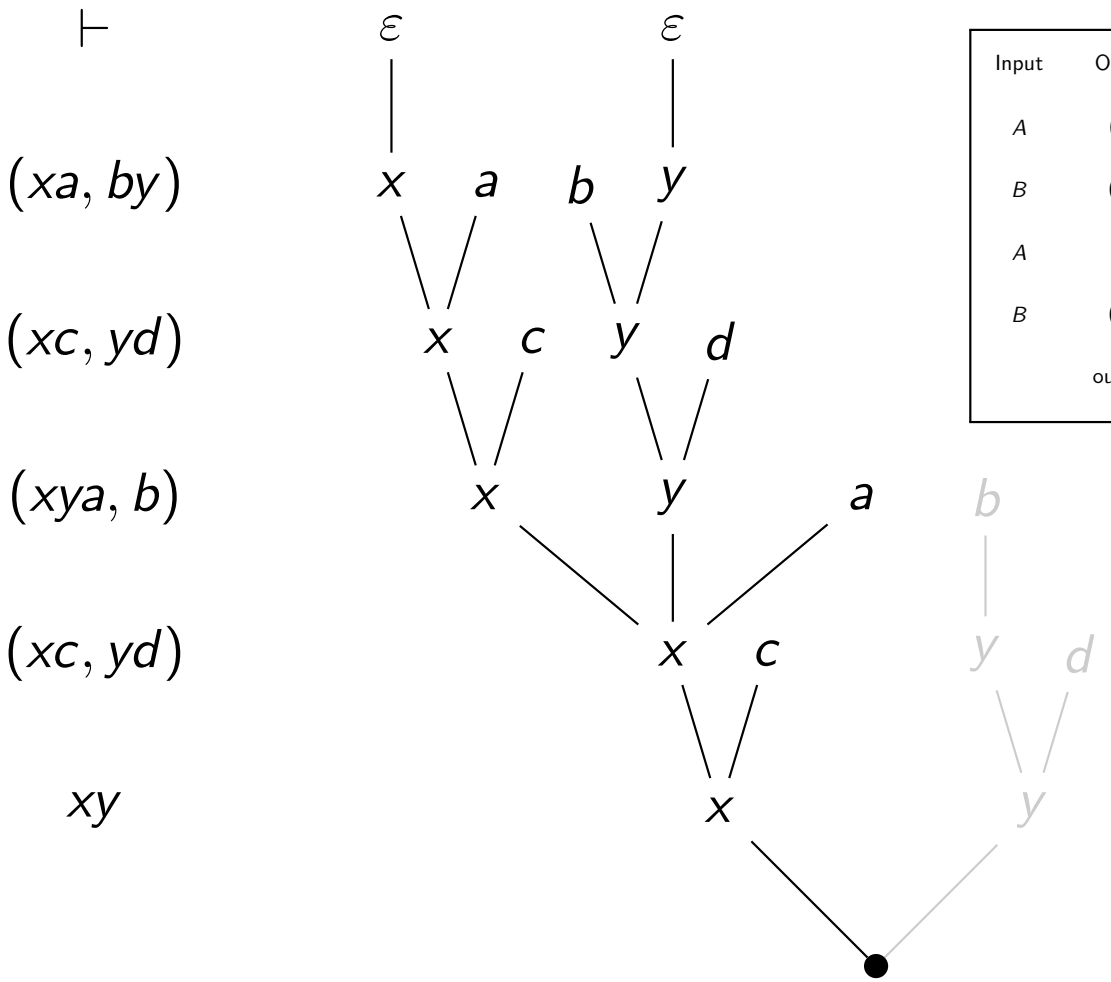
Preuve de correction

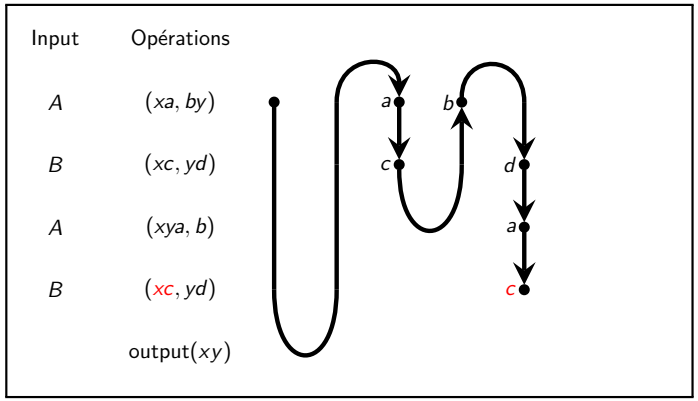




Résultat : $\varepsilon acb\varepsilon da$

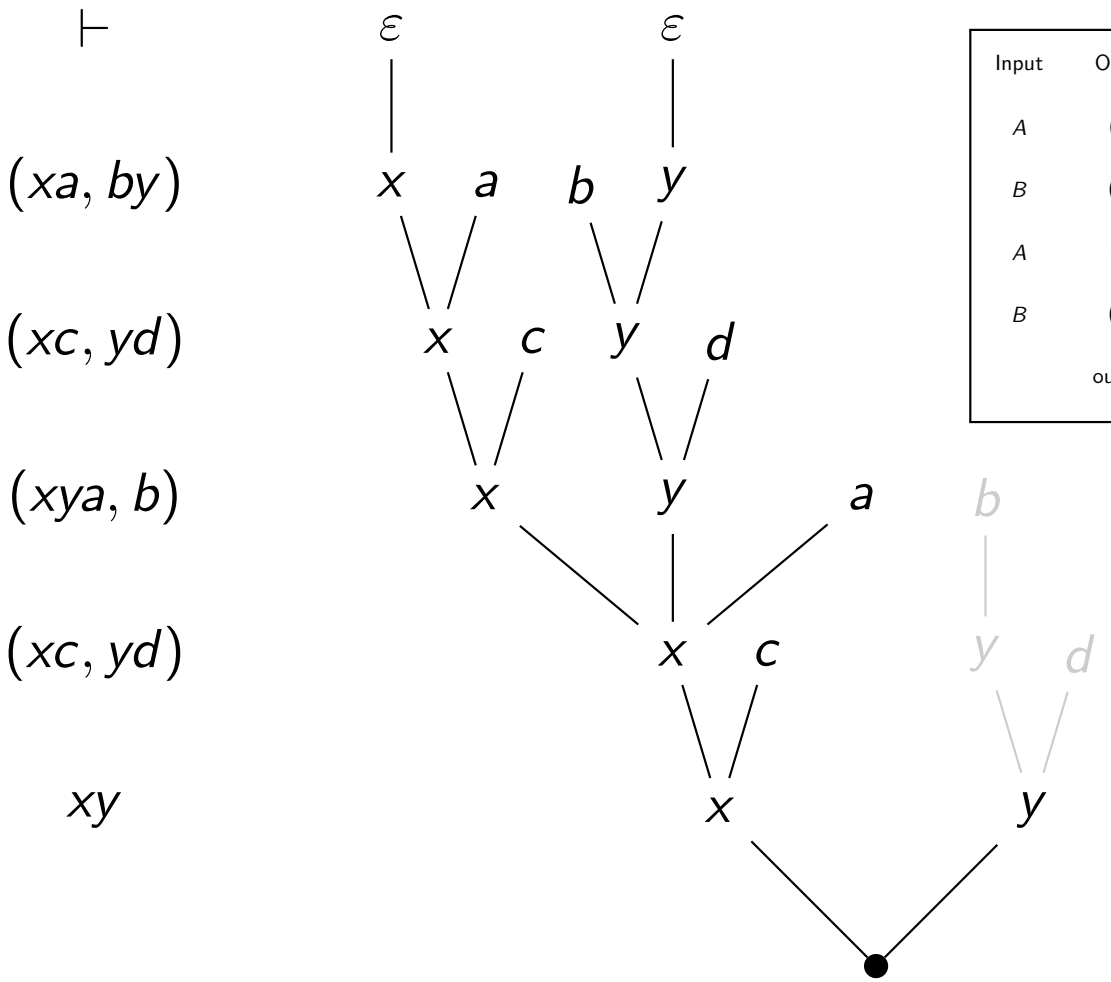
Preuve de correction

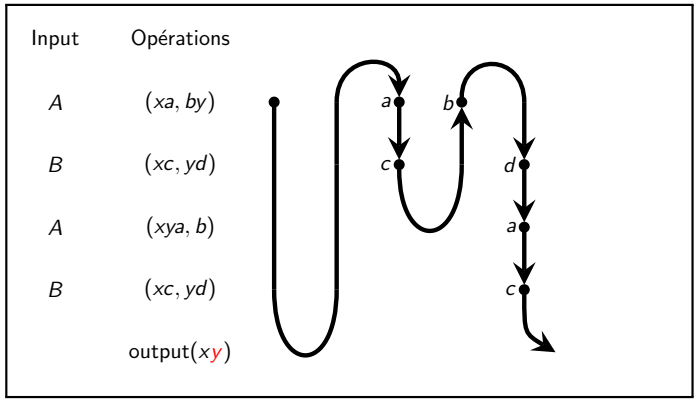




Résultat : $\varepsilon acb\varepsilon dac$

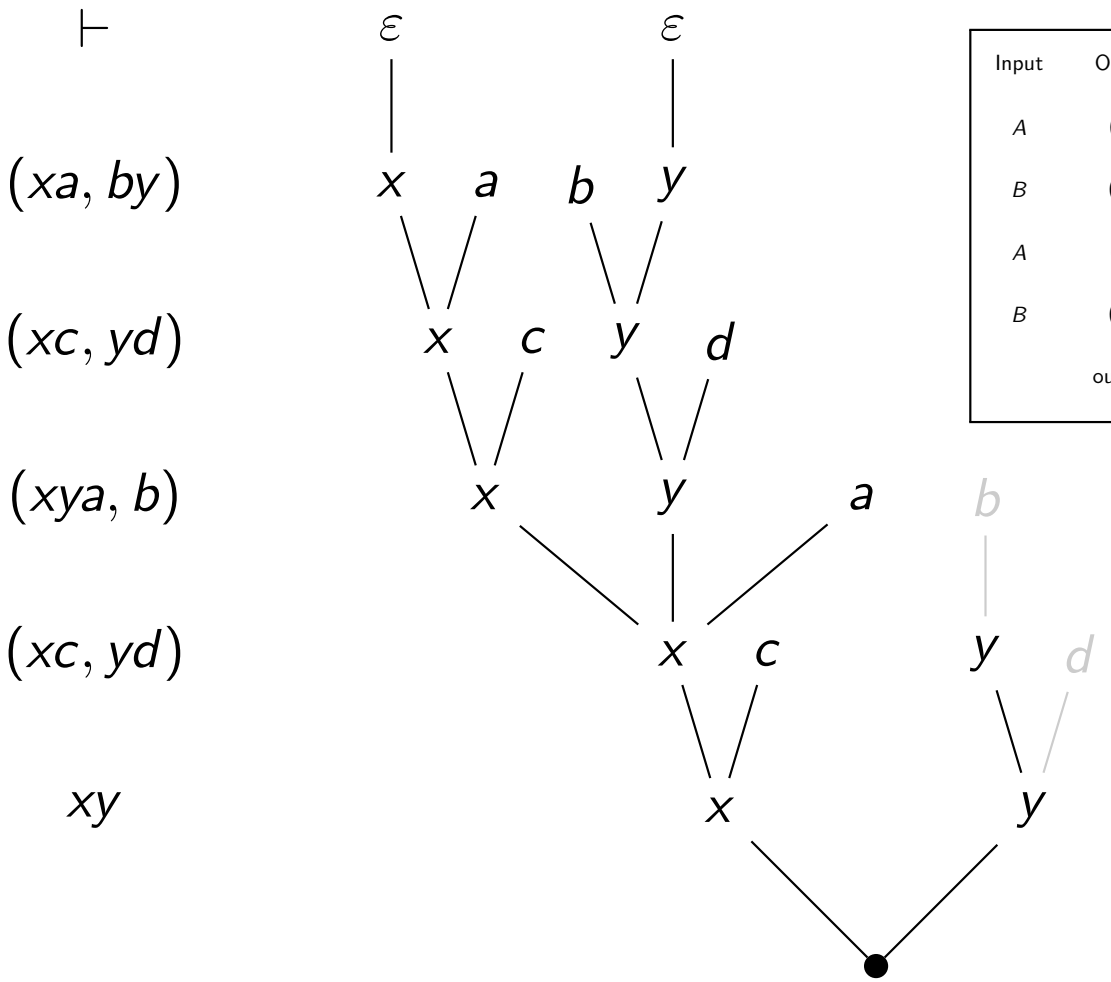
Preuve de correction

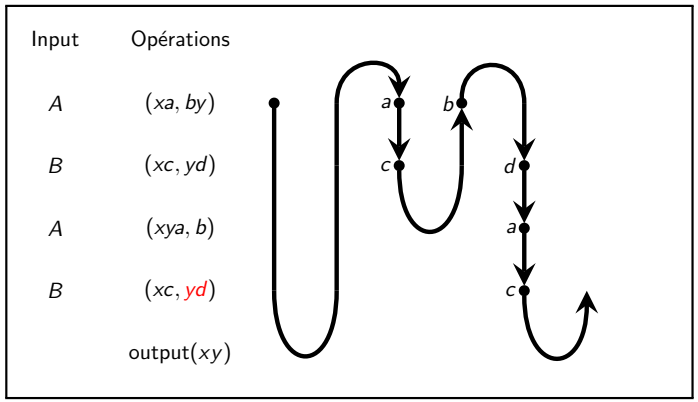




Résultat : $\varepsilon acb\varepsilon dac$

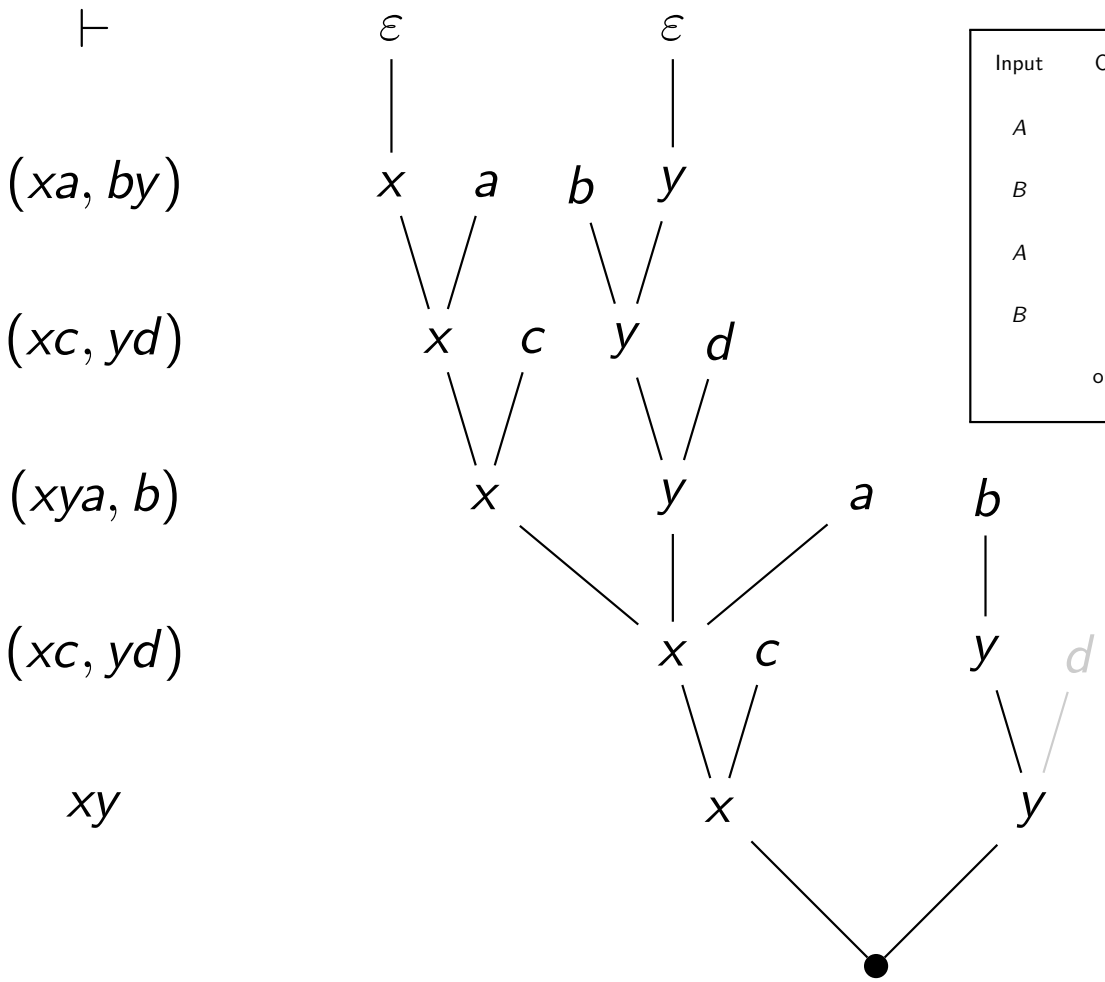
Preuve de correction

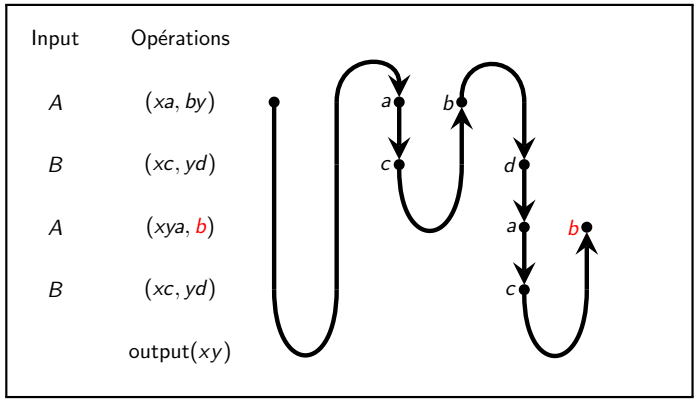




Résultat : $\varepsilon acb\varepsilon dac$

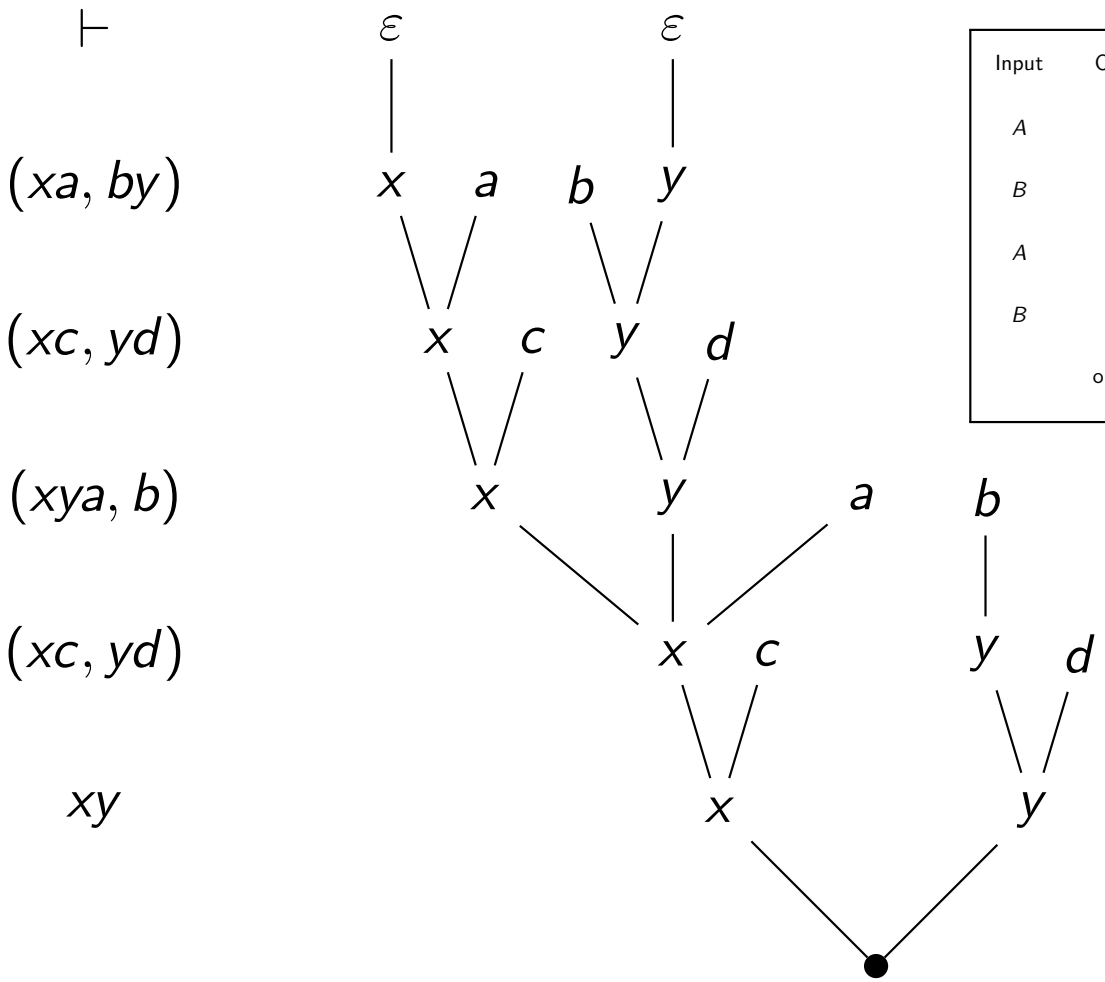
Preuve de correction

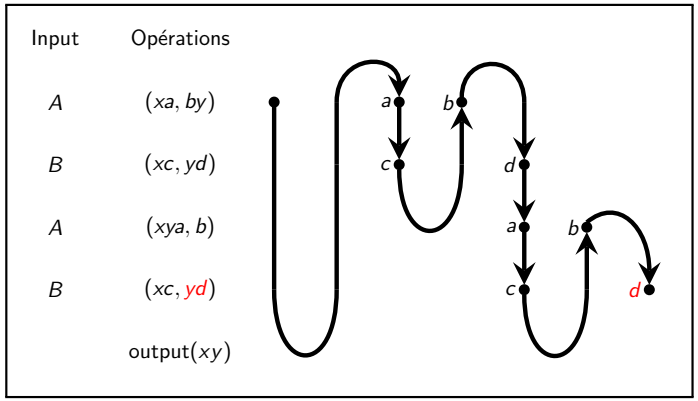




Résultat : $\varepsilon acb\varepsilon dacb$

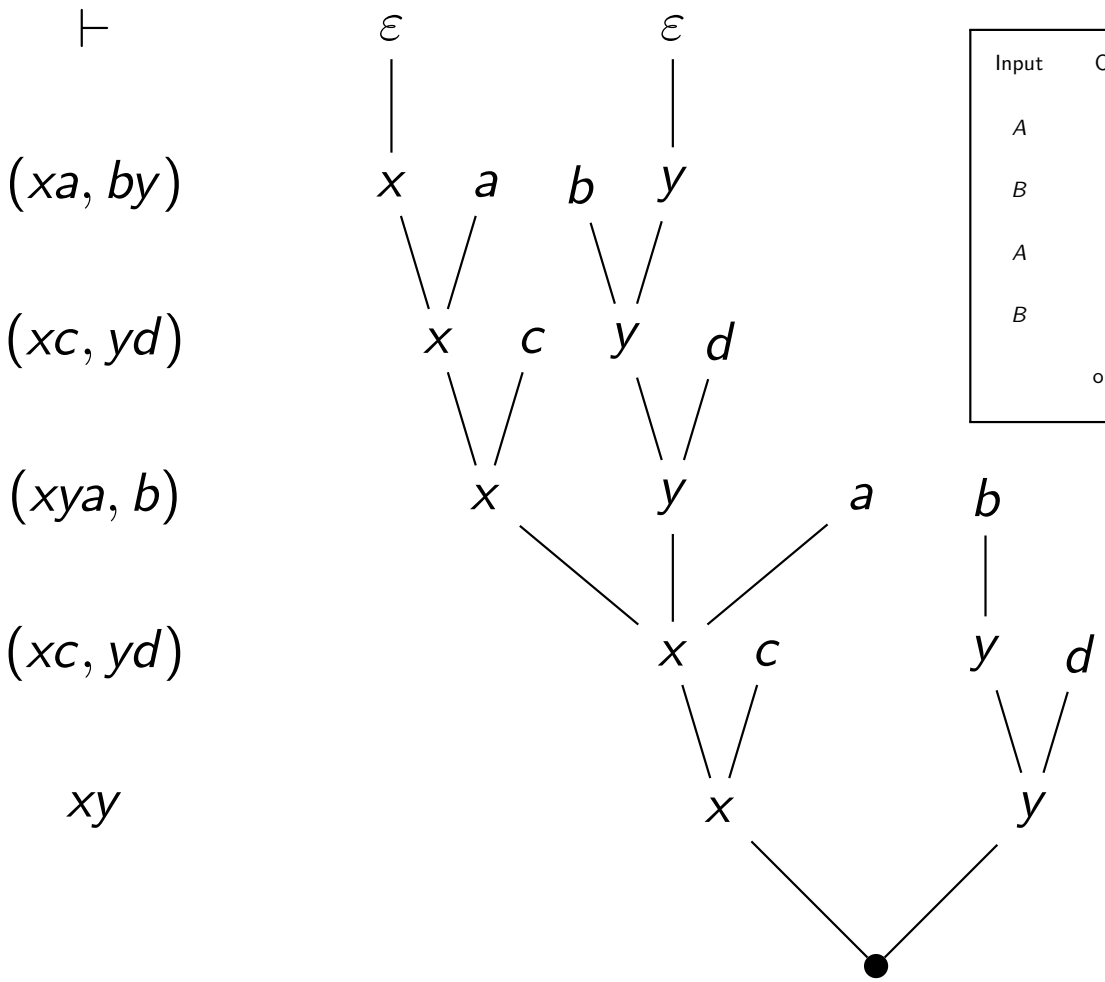
Preuve de correction

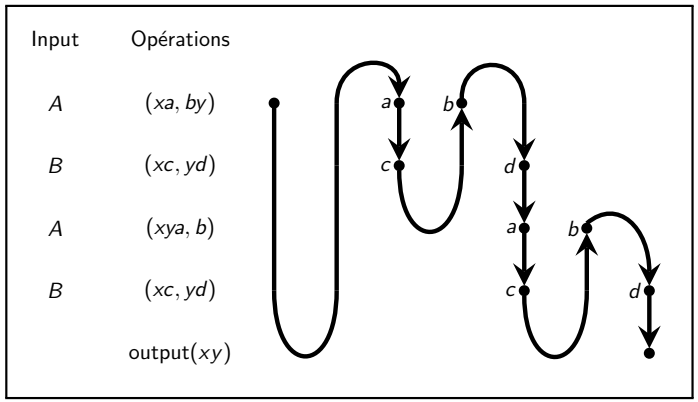




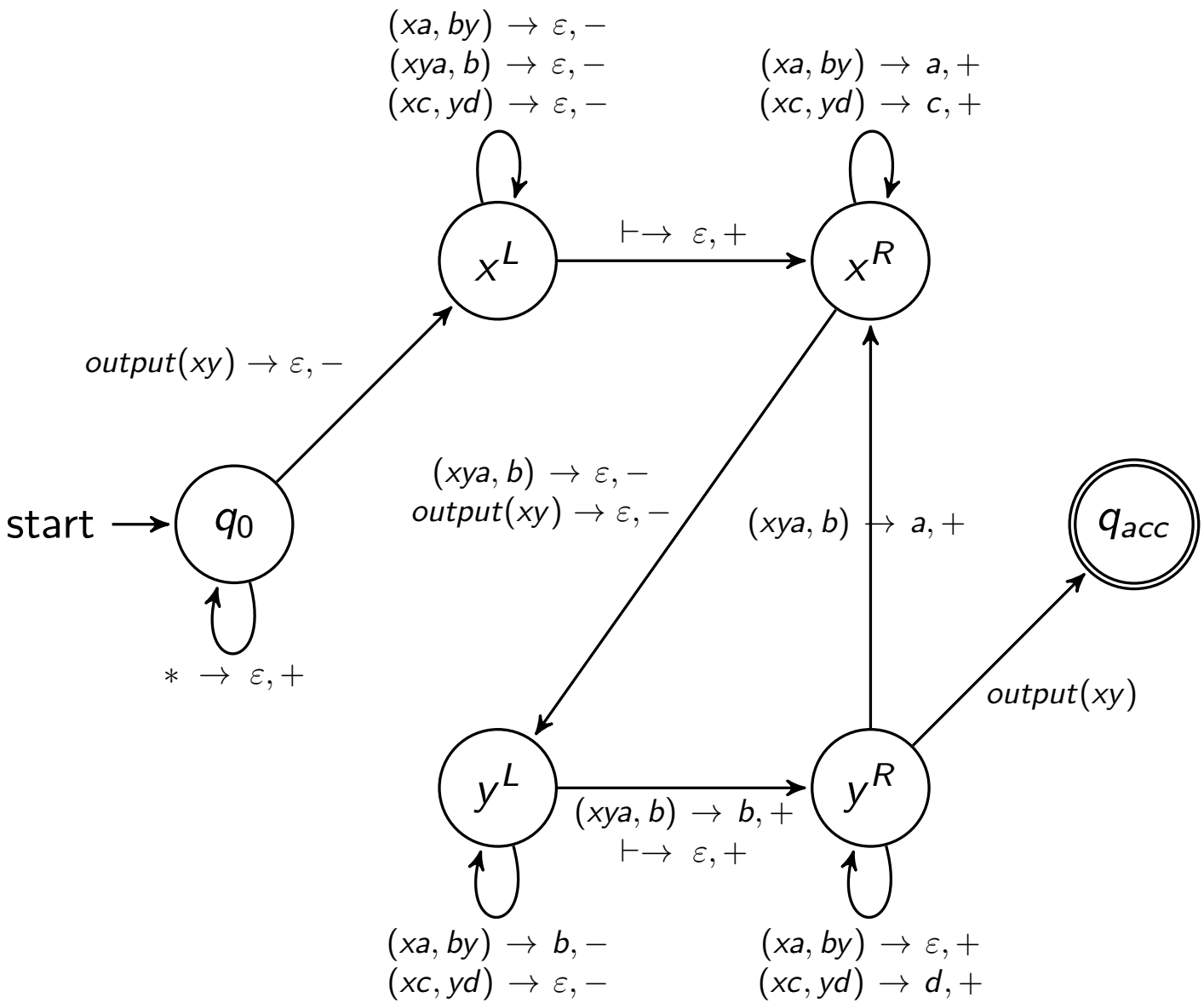
Résultat : $\varepsilon acb\varepsilon dacbd$

Preuve de correction





Résultat : $\varepsilon acb\varepsilon dacbd = acbdacbd$


$(xa, by) \to \varepsilon, -$
$(xya, b) \to \varepsilon, -$
$(xc, yd) \to \varepsilon, -$
$(xa, by) \to a, +$
$(xc, yd) \to c, +$
x^L
$\vdash\!\to \varepsilon, +$
x^R
$output(xy) \to \varepsilon, -$
start
q_0
$(xya, b) \to \varepsilon, -$
$output(xy) \to \varepsilon, -$
$(xya, b) \to a, +$
q_{acc}
$* \to \varepsilon, +$
$output(xy)$
y^L
$(xya, b) \to b, +$
$\vdash\!\to \varepsilon, +$
y^R
$(xa, by) \to b, -$
$(xc, yd) \to \varepsilon, -$
$(xa, by) \to \varepsilon, +$
$(xc, yd) \to d, +$

Théorème

(1) A partir d'un DSST à k variables et n états, on construit un 2DGSM équivalent de taille $\mathcal{O}(n^n \times k)$.

Théorème

(1) A partir d'un DSST à k variables et n états, on construit un 2DGSM équivalent de taille $\mathcal{O}(n^n \times k)$.

(2) A partir d'un DSST_{RLA} à k variables et n états, on construit un $\text{2DGSM}_{\text{RLA}}$ équivalent de taille $\mathcal{O}(k)$.
Son nombre de visites est au plus $2k + 1$.

Plan

Conclusion

Les SST sont un modèle intéressant de transducteurs :

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)
- Equivalent à la logique MSO, même non-déterministe

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)
- Equivalent à la logique MSO, même non-déterministe
- Décidabilité de l'équivalence dans le cas fonctionnel

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)
- Equivalent à la logique MSO, même non-déterministe
- Décidabilité de l'équivalence dans le cas fonctionnel

C'est de plus un modèle récent, avec beaucoup de voies à explorer :

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)
- Equivalent à la logique MSO, même non-déterministe
- Décidabilité de l'équivalence dans le cas fonctionnel

C'est de plus un modèle récent, avec beaucoup de voies à explorer :

- Variantes : string-to-tree, tree-to-tree

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)
- Equivalent à la logique MSO, même non-déterministe
- Décidabilité de l'équivalence dans le cas fonctionnel

C'est de plus un modèle récent, avec beaucoup de voies à explorer :

- Variantes : string-to-tree, tree-to-tree
- *Cost register automata* : calcul de fonctions numériques

Conclusion

Les SST sont un modèle intéressant de transducteurs :

- Modèle simple (one-way)
- Equivalent à la logique MSO, même non-déterministe
- Décidabilité de l'équivalence dans le cas fonctionnel

C'est de plus un modèle récent, avec beaucoup de voies à explorer :

- Variantes : string-to-tree, tree-to-tree
- *Cost register automata* : calcul de fonctions numériques
- Beaucoup de questions restent ouvertes, par exemple l'équivalence des NSST k-valués