Machines de Turing, langages, automates, grammaires

INFO1 - Semaines 42 à 2

Guillaume CONNAN

IUT de Nantes - Dpt d'informatique

Dernière mise à jour : 6 janvier 2012

Sommaire

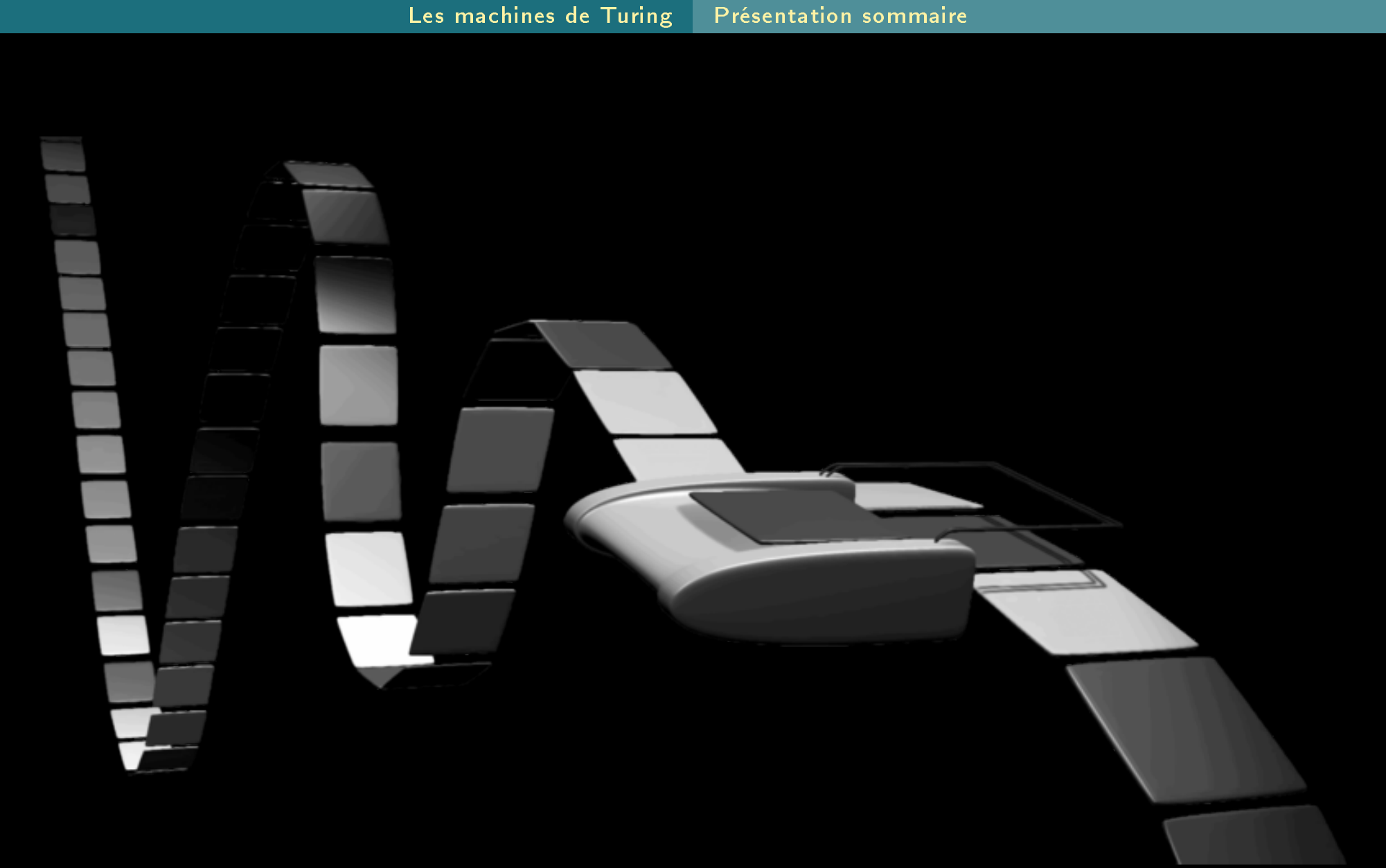
Sommaire

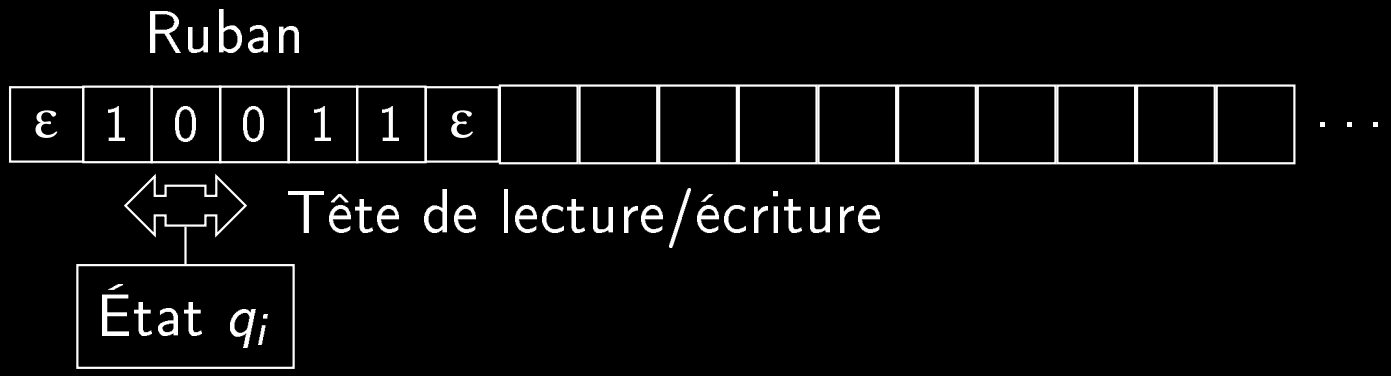
Alonzo Church (1903-1995)

Alan Turing (1912-1954)

Sommaire

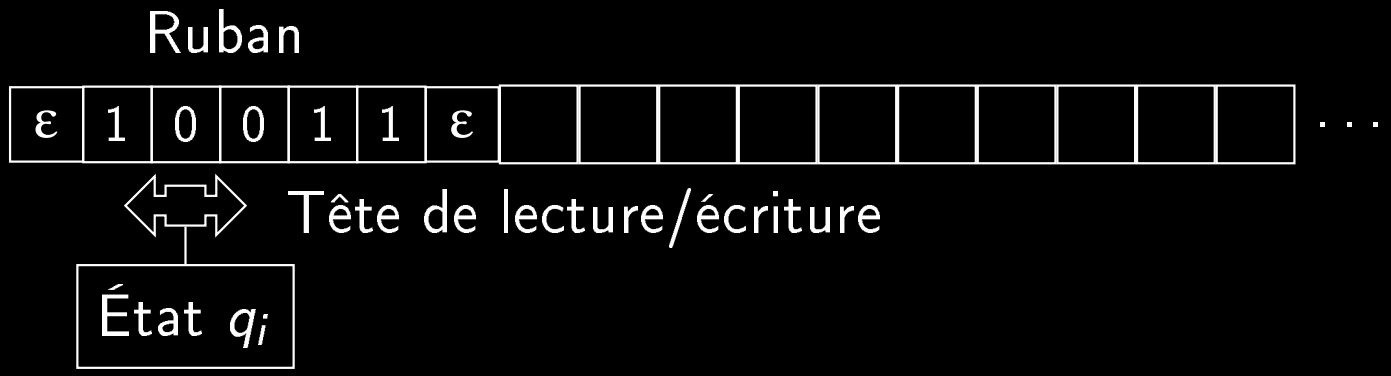
Sommaire





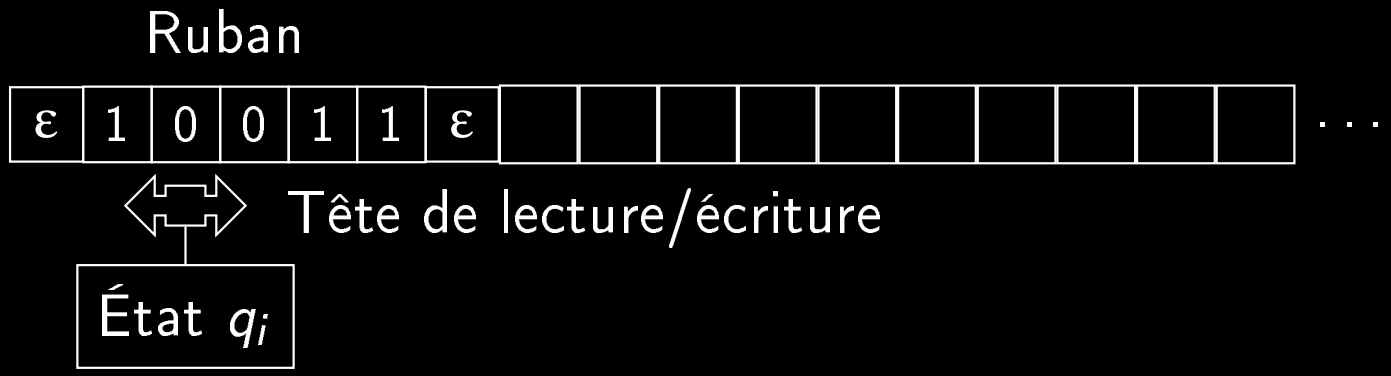
L'ensemble des symboles $\{0,1,\varepsilon\}$ forme un **alphabet**.

- changer l'état du pointeur ;



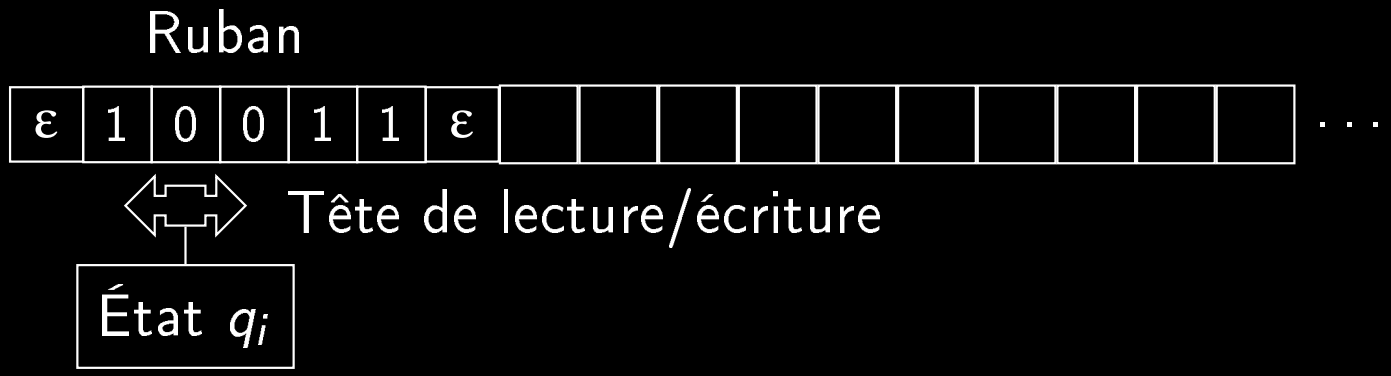
L'ensemble des symboles $\{0,1,\varepsilon\}$ forme un **alphabet**.

- changer l'état du pointeur ;
- changer le caractère pointé sur le ruban ;
- déplacer le pointeur d'une case vers la gauche ou vers la droite.



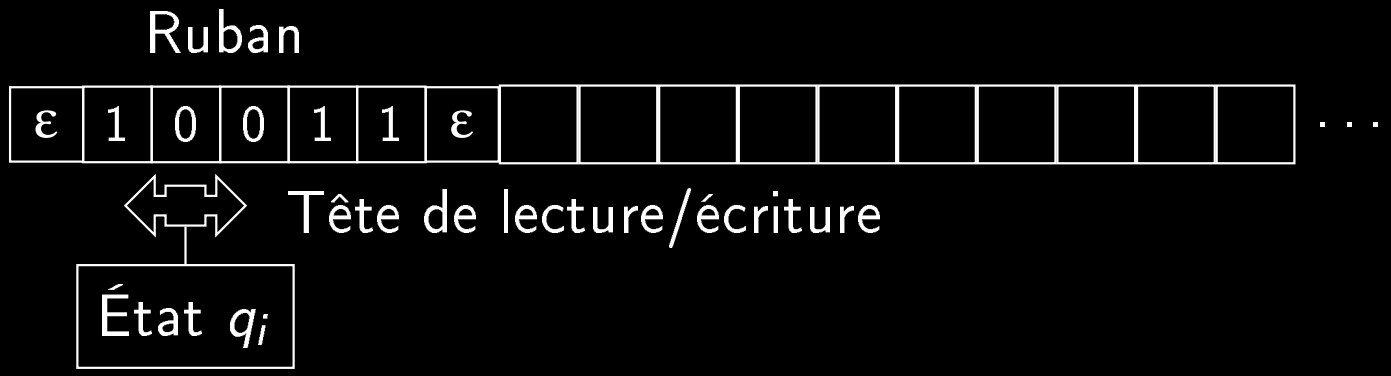
L'ensemble des symboles $\{0,1,\varepsilon\}$ forme un **alphabet**.

- changer l'état du pointeur ;
- changer le caractère pointé sur le ruban ;
- déplacer le pointeur d'une case vers la gauche ou vers la droite.



L'ensemble des symboles $\{0,1,\varepsilon\}$ forme un **alphabet**.

- changer l'état du pointeur ;
- changer le caractère pointé sur le ruban ;
- déplacer le pointeur d'une case vers la gauche ou vers la droite.



L'ensemble des symboles $\{0,1,\varepsilon\}$ forme un **alphabet**.

- changer l'état du pointeur ;
- changer le caractère pointé sur le ruban ;
- déplacer le pointeur d'une case vers la gauche ou vers la droite.

Sommaire

On entre un « mot » écrit à partir de l'alphabet $\{0,1,\varepsilon\}$.

Nous voudrions changer les 0 en 1 et vice-versa puis remettre le pointeur dans sa position initiale.

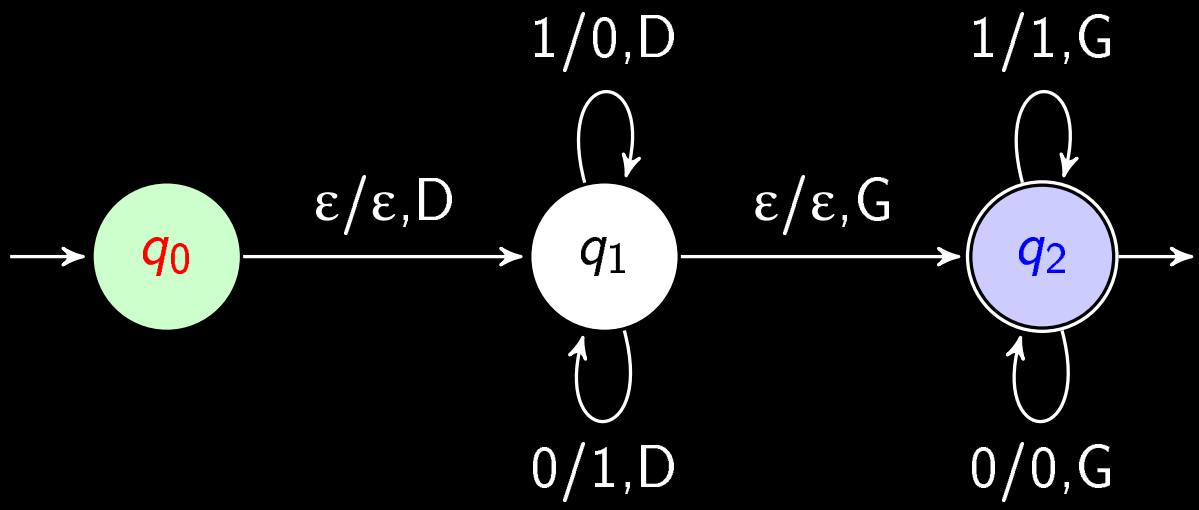
On entre un « mot » écrit à partir de l'alphabet $\{0,1,\varepsilon\}$.
Nous voudrions changer les 0 en 1 et vice-versa puis remettre le pointeur dans sa position initiale.

- lorsque le pointeur rencontre le premier blanc, il se déplace vers la droite ;
- chaque fois que le pointeur rencontre un 0 ou un 1, il le change en son complémentaire et se déplace vers la droite ;
- lorsqu'il rencontre le deuxième blanc, il recule d'une case vers la gauche jusqu'à ce qu'il rencontre le premier blanc.

- lorsque le pointeur rencontre le premier blanc, il se déplace vers la droite ;
- chaque fois que le pointeur rencontre un 0 ou un 1, il le change en son complémentaire et se déplace vers la droite ;
- lorsqu'il rencontre le deuxième blanc, il recule d'une case vers la gauche jusqu'à ce qu'il rencontre le premier blanc.

- lorsque le pointeur rencontre le premier blanc, il se déplace vers la droite ;
- chaque fois que le pointeur rencontre un 0 ou un 1, il le change en son complémentaire et se déplace vers la droite ;
- lorsqu'il rencontre le deuxième blanc, il recule d'une case vers la gauche jusqu'à ce qu'il rencontre le premier blanc.

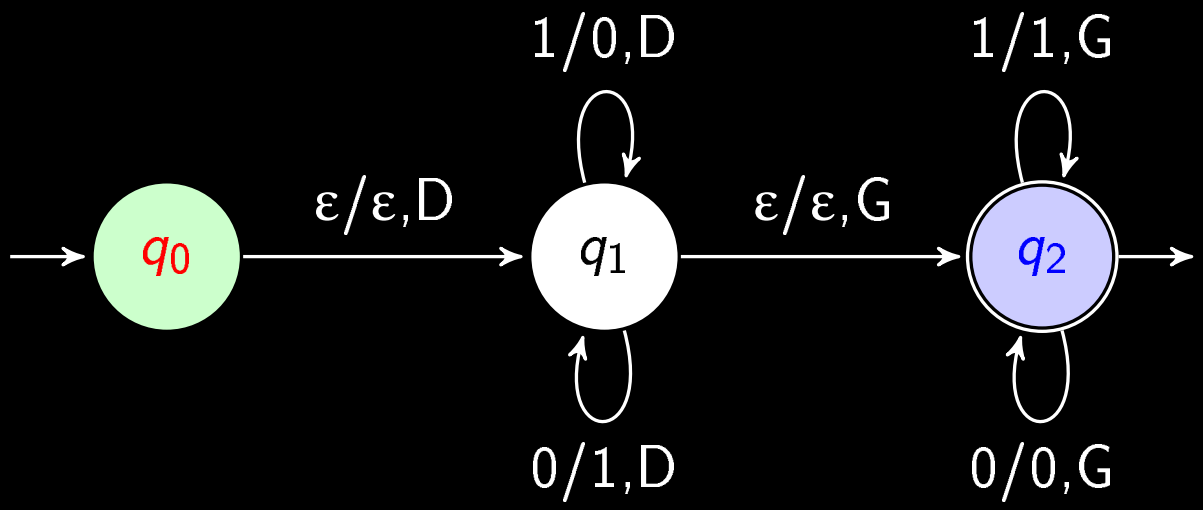
automate



Exemple

Testez cette machine sur la chaîne ε10011ε

automate



Exemple

Testez cette machine sur la chaîne ε10011ε.

Sommaire

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_0 \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\epsilon \in X$ est le symbole blanc.

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_o \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\varepsilon \in X$ est le symbole blanc.

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_o \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\varepsilon \in X$ est le symbole blanc.

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_o \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\varepsilon \in X$ est le symbole blanc.

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_o \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\varepsilon \in X$ est le symbole blanc.

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_o \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\varepsilon \in X$ est le symbole blanc.

Définition 1 (Machine de Turing)

Une machine de Turing standard (il en existe d'autres) est un triplet :

$$M = (Q, X, \varphi)$$

où

- Q est un ensemble fini appelé *alphabet d'état* ;
- X est un autre ensemble fini appelé *alphabet de ruban* ;
- φ est une fonction de domaine $D \subseteq Q \times X$ et à valeurs dans $Q \times X \times \{G, R, D\}$ avec G, R, D les options du mouvement du pointeur (gauche, reste immobile, droite).

On choisira en général Q et X de telle sorte que leur intersection soit vide.
Quelques éléments particuliers :

- $q_o \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\varepsilon \in X$ est le symbole blanc.

- $Q = \{q_0, q_1, q_2\}$
- $X = \{\varepsilon, 0, 1\}$
- $Q_f = \{q_2\}$
- φ est définie à l'aide du tableau suivant :

$\varphi(q_i, x)$	q_0	q_1	q_2
ε	(q_1, ε, D)	(q_2, ε, G)	
0		$(q_1, 1, D)$	$(q_2, 0, G)$
1		$(q_1, 0, D)$	$(q_2, 1, G)$

- $Q = \{q_0, q_1, q_2\}$
- $X = \{\varepsilon, 0, 1\}$
- $Q_f = \{q_2\}$
- φ est définie à l'aide du tableau suivant :

$\varphi(q_i, x)$	q_0	q_1	q_2
ε	(q_1, ε, D)	(q_2, ε, G)	
0		$(q_1, 1, D)$	$(q_2, 0, G)$
1		$(q_1, 0, D)$	$(q_2, 1, G)$

- $Q = \{q_0, q_1, q_2\}$
- $X = \{\varepsilon, 0, 1\}$
- $Q_f = \{q_2\}$
- φ est définie à l'aide du tableau suivant :

$\varphi(q_i, x)$	q_0	q_1	q_2
ε	(q_1, ε, D)	(q_2, ε, G)	
0		$(q_1, 1, D)$	$(q_2, 0, G)$
1		$(q_1, 0, D)$	$(q_2, 1, G)$

- $Q = \{q_0, q_1, q_2\}$
- $X = \{\varepsilon, 0, 1\}$
- $Q_f = \{q_2\}$
- φ est définie à l'aide du tableau suivant :

$\varphi(q_i, x)$	q_0	q_1	q_2
ε	(q_1, ε, D)	(q_2, ε, G)	
0		$(q_1, 1, D)$	$(q_2, 0, G)$
1		$(q_1, 0, D)$	$(q_2, 1, G)$

En fait, une machine de Turing agit sur des *mots* construits à partir d'un *alphabet*.

Définition 2

Soit X un alphabet. Soit λ le mot ne comportant aucun caractère.
L'ensemble X^* des mots construits avec l'alphabet X est défini par :

1. $\lambda \in X^*$;
2. si $a \in X$ et $m \in X^*$, alors le mot construit à partir de m en ajoutant a à droite est un mot de X^* ;
3. μ est un élément de X^* seulement s'il peut être obtenu à partir de λ par application de l'étape 2 un nombre fini de fois.

Définition 2

Soit X un alphabet. Soit λ le mot ne comportant aucun caractère.
L'ensemble X^* des mots construits avec l'alphabet X est défini par :

1. $\lambda \in X^*$;
2. si $a \in X$ et $m \in X^*$, alors le mot construit à partir de m en ajoutant a à droite est un mot de X^* ;
3. μ est un élément de X^* seulement s'il peut être obtenu à partir de λ par application de l'étape 2 un nombre fini de fois.

Définition 2

Soit X un alphabet. Soit λ le mot ne comportant aucun caractère.
L'ensemble X^* des mots construits avec l'alphabet X est défini par :

1. $\lambda \in X^*$;
2. si $a \in X$ et $m \in X^*$, alors le mot construit à partir de m en ajoutant a à droite est un mot de X^* ;
3. μ est un élément de X^* seulement s'il peut être obtenu à partir de λ par application de l'étape 2 un nombre fini de fois.

Définition 2

Soit X un alphabet. Soit λ le mot ne comportant aucun caractère.
L'ensemble X^* des mots construits avec l'alphabet X est défini par :

1. $\lambda \in X^*$;
2. si $a \in X$ et $m \in X^*$, alors le mot construit à partir de m en ajoutant a à droite est un mot de X^* ;
3. μ est un élément de X^* seulement s'il peut être obtenu à partir de λ par application de l'étape 2 un nombre fini de fois.

Définition 3 (Concaténation)

Soient m_1 et m_2 deux mots de X^*. La **concaténation** de ces deux mots est le mot $m_1 m_2 \in X^*$ obtenu en écrivant m_2 à la suite de m_1.

Sommaire

Exemple 4

Nous allons construire une machine de Turing qui calcule le successeur de la représentation unaire d'un entier.

Définition 5 (Représentation unaire d'un entier)

Un nombre entier $n \in \mathbb{N}$ a pour représentation unaire 1^{n+1} c'est-à-dire la concaténation de $n+1$ symboles de l'alphabet $X = \{1\}$. Ainsi, la représentation de 0 est 1, celle de 1 est 11, celle de 2 est 111, etc. On notera $\overline{n}$ la représentation de l'entier n.

Définition 5 (Représentation unaire d'un entier)

Un nombre entier $n \in \mathbb{N}$ a pour représentation unaire 1^{n+1} c'est-à-dire la concaténation de $n+1$ symboles de l'alphabet $X = \{1\}$. Ainsi, la représentation de 0 est 1, celle de 1 est 11, celle de 2 est 111, etc. On notera $\overline{n}$ la représentation de l'entier n.

Définition 5 (Représentation unaire d'un entier)

Un nombre entier $n \in \mathbb{N}$ a pour représentation unaire 1^{n+1} c'est-à-dire la concaténation de $n+1$ symboles de l'alphabet $X = \{1\}$. Ainsi, la représentation de 0 est 1, celle de 1 est 11, celle de 2 est 111, etc. On notera $\overline{n}$ la représentation de l'entier n.

Définition 5 (Représentation unaire d'un entier)

Un nombre entier $n \in \mathbb{N}$ a pour représentation unaire 1^{n+1} c'est-à-dire la concaténation de $n+1$ symboles de l'alphabet $X = \{1\}$. Ainsi, la représentation de 0 est 1, celle de 1 est 11, celle de 2 est 111, etc. On notera $\overline{n}$ la représentation de l'entier n.

Décrivons une machine de Turing qui calcule cette fonction :

- la fonction successeur s est définie sur $D = \{\lambda, \varepsilon 1 \varepsilon, \varepsilon 11 \varepsilon, \varepsilon 111 \varepsilon, ...\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole ε, toutes les autres cases étant occupées par des ε ;
- l'alphabet de ruban est $X = \{1, \varepsilon\}$;
- l'alphabet d'état est $\{q_0, q_1, q_2\}$;
- $Q_f = \{q_2\}$;
- la fonction φ est décrite par l'automate suivant :

Décrivons une machine de Turing qui calcule cette fonction :

- la fonction successeur s est définie sur $D = \{\lambda, \varepsilon 1 \varepsilon, \varepsilon 11 \varepsilon, \varepsilon 111 \varepsilon, ...\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole ε, toutes les autres cases étant occupées par des ε ;
- l'alphabet de ruban est $X = \{1, \varepsilon\}$;
- l'alphabet d'état est $\{q_0, q_1, q_2\}$;
- $Q_f = \{q_2\}$;
- la fonction φ est décrite par l'automate suivant :

Décrivons une machine de Turing qui calcule cette fonction :

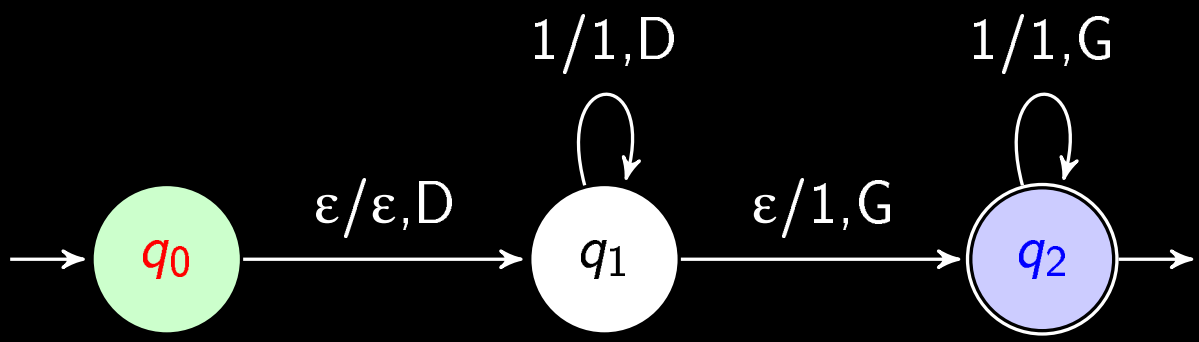
- la fonction successeur s est définie sur $D = \{\lambda, \varepsilon 1 \varepsilon, \varepsilon 11 \varepsilon, \varepsilon 111 \varepsilon, ...\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole ε, toutes les autres cases étant occupées par des ε ;
- l'alphabet de ruban est $X = \{1, \varepsilon\}$;
- l'alphabet d'état est $\{q_0, q_1, q_2\}$;
- $Q_f = \{q_2\}$;
- la fonction φ est décrite par l'automate suivant :

Décrivons une machine de Turing qui calcule cette fonction :

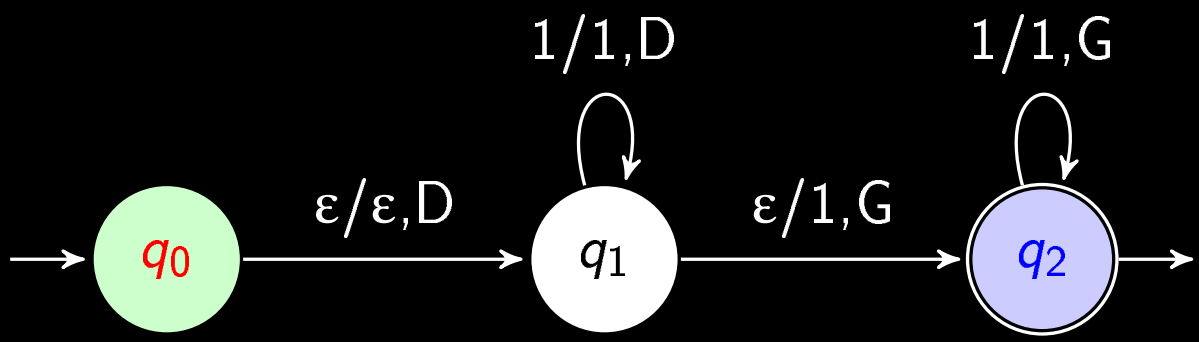
- la fonction successeur s est définie sur $D = \{\lambda, \varepsilon 1 \varepsilon, \varepsilon 11 \varepsilon, \varepsilon 111 \varepsilon, ...\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole ε, toutes les autres cases étant occupées par des ε ;
- l'alphabet de ruban est $X = \{1, \varepsilon\}$;
- l'alphabet d'état est $\{q_0, q_1, q_2\}$;
- $Q_f = \{q_2\}$;
- la fonction φ est décrite par l'automate suivant :

Décrivons une machine de Turing qui calcule cette fonction :

- la fonction successeur s est définie sur $D = \{\lambda, \varepsilon 1 \varepsilon, \varepsilon 11 \varepsilon, \varepsilon 111 \varepsilon, ...\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole ε, toutes les autres cases étant occupées par des ε ;
- l'alphabet de ruban est $X = \{1, \varepsilon\}$;
- l'alphabet d'état est $\{q_0, q_1, q_2\}$;
- $Q_f = \{q_2\}$;
- la fonction φ est décrite par l'automate suivant :

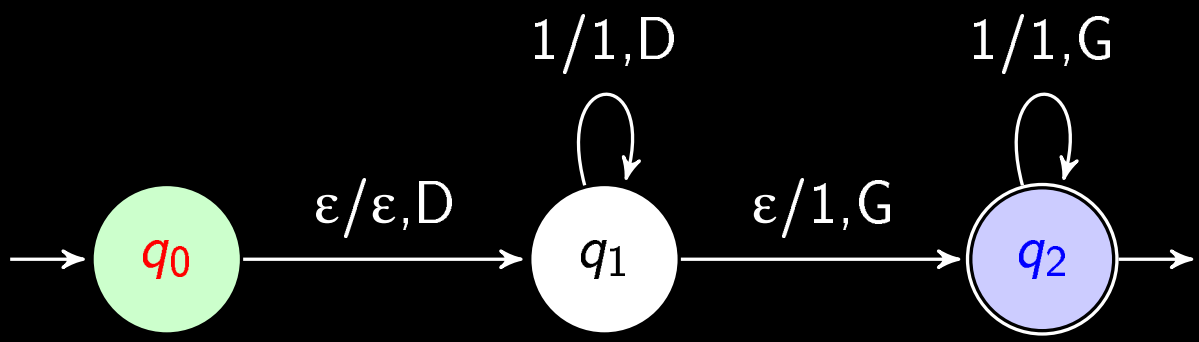


La configuration initiale est le mot $q_0\varepsilon\overline{n}\varepsilon$.
La configuration finale est $q_2\varepsilon\overline{n}1\varepsilon$.



La configuration initiale est le mot $q_0\varepsilon\overline{n}\varepsilon$.

La configuration finale est $q_2\varepsilon\overline{n}1\varepsilon$.



La configuration initiale est le mot $q_0\varepsilon\overline{n}\varepsilon$.
La configuration finale est $q_2\varepsilon\overline{n}1\varepsilon$.

Définition 6 (Fonction MT-calculable)

Soit $f : D \subseteq X^* \to X^*$ une fonction et $M = (Q, X, \varphi)$ une machine de Turing. M calcule f si :

- il existe une unique transition de q_0 et que sa forme est $\varphi(q_0, \varepsilon) = (q_i, \varepsilon, D)$ avec $q_i \neq q_0$ (*il faut quitter l'état initial à partir de* ε) ;
- il n'existe pas de transition de la forme $\varphi(q_i, x) = (q_0, x', m)$ avec $i \neq 0$, $(x, x') \in X^2$ et $m \in \{G, R, D\}$ (*il ne faut pas revenir à l'état initial*) ;
- il n'existe pas de transition de la forme $\varphi(q_f, \varepsilon)$ où $q_f \in Q_f$ (*il ne faut pas que la machine continue à fonctionner dans son état final en présence d'un blanc car il y en a une infinité et la machine ne s'arrêterait jamais*) ;
- pour tout $\mu in D$, notons $\nu = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \varepsilon \mu \varepsilon$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \varepsilon \nu \varepsilon$ (*la machine doit calculer* $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (*la machine ne calcule* $f(\mu)$ *que si ce nombre est défini*).

Définition 6 (Fonction MT-calculable)

Soit $f : D \subseteq X^* \rightarrow X^*$ une fonction et $M = (Q, X, \varphi)$ une machine de Turing. M calcule f si :

- il existe une unique transition de q_0 et que sa forme est $\varphi(q_0, \varepsilon) = (q_i, \varepsilon, D)$ avec $q_i \neq q_0$ (*il faut quitter l'état initial à partir de* ε) ;
- il n'existe pas de transition de la forme $\varphi(q_i, x) = (q_0, x', m)$ avec $i \neq 0$, $(x, x') \in X^2$ et $m \in \{G, R, D\}$ (*il ne faut pas revenir à l'état initial*) ;
- il n'existe pas de transition de la forme $\varphi(q_f, \varepsilon)$ où $q_f \in Q_f$ (*il ne faut pas que la machine continue à fonctionner dans son état final en présence d'un blanc car il y en a une infinité et la machine ne s'arrêterait jamais*) ;
- pour tout $\mu in D$, notons $\nu = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0\varepsilon\mu\varepsilon$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f\varepsilon\nu\varepsilon$ (*la machine doit calculer* $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (*la machine ne calcule* $f(\mu)$ *que si ce nombre est défini*).

Définition 6 (Fonction MT-calculable)

Soit $f : D \subseteq X^* \rightarrow X^*$ une fonction et $M = (Q, X, \varphi)$ une machine de Turing. M calcule f si :

- il existe une unique transition de q_0 et que sa forme est $\varphi(q_0, \varepsilon) = (q_i, \varepsilon, D)$ avec $q_i \neq q_0$ (*il faut quitter l'état initial à partir de* ε) ;
- il n'existe pas de transition de la forme $\varphi(q_i, x) = (q_0, x', m)$ avec $i \neq 0$, $(x, x') \in X^2$ et $m \in \{G, R, D\}$ (*il ne faut pas revenir à l'état initial*) ;
- il n'existe pas de transition de la forme $\varphi(q_f, \varepsilon)$ où $q_f \in Q_f$ (*il ne faut pas que la machine continue à fonctionner dans son état final en présence d'un blanc car il y en a une infinité et la machine ne s'arrêterait jamais*) ;
- pour tout $\mu in D$, notons $\nu = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \varepsilon \mu \varepsilon$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \varepsilon \nu \varepsilon$ (*la machine doit calculer* $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (*la machine ne calcule* $f(\mu)$ *que si ce nombre est défini*).

Définition 6 (Fonction MT-calculable)

Soit $f : D \subseteq X^* \to X^*$ une fonction et $M = (Q, X, \varphi)$ une machine de Turing. M calcule f si :

- il existe une unique transition de q_0 et que sa forme est $\varphi(q_0, \varepsilon) = (q_i, \varepsilon, D)$ avec $q_i \neq q_0$ (*il faut quitter l'état initial à partir de* ε) ;
- il n'existe pas de transition de la forme $\varphi(q_i, x) = (q_0, x', m)$ avec $i \neq 0$, $(x, x') \in X^2$ et $m \in \{G, R, D\}$ (*il ne faut pas revenir à l'état initial*) ;
- il n'existe pas de transition de la forme $\varphi(q_f, \varepsilon)$ où $q_f \in Q_f$ (*il ne faut pas que la machine continue à fonctionner dans son état final en présence d'un blanc car il y en a une infinité et la machine ne s'arrêterait jamais*) ;
- pour tout $\mu in D$, notons $\nu = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \varepsilon \mu \varepsilon$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \varepsilon \nu \varepsilon$ (*la machine doit calculer* $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (*la machine ne calcule* $f(\mu)$ *que si ce nombre est défini*).

Définition 6 (Fonction MT-calculable)

Soit $f : D \subseteq X^* \rightarrow X^*$ une fonction et $M = (Q, X, \varphi)$ une machine de Turing. M calcule f si :

- il existe une unique transition de q_0 et que sa forme est $\varphi(q_0, \varepsilon) = (q_i, \varepsilon, D)$ avec $q_i \neq q_0$ (*il faut quitter l'état initial à partir de* ε) ;
- il n'existe pas de transition de la forme $\varphi(q_i, x) = (q_0, x', m)$ avec $i \neq 0$, $(x, x') \in X^2$ et $m \in \{G, R, D\}$ (*il ne faut pas revenir à l'état initial*) ;
- il n'existe pas de transition de la forme $\varphi(q_f, \varepsilon)$ où $q_f \in Q_f$ (*il ne faut pas que la machine continue à fonctionner dans son état final en présence d'un blanc car il y en a une infinité et la machine ne s'arrêterait jamais*) ;
- pour tout $\mu in D$, notons $\nu = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \varepsilon \mu \varepsilon$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \varepsilon \nu \varepsilon$ (*la machine doit calculer* $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (*la machine ne calcule* $f(\mu)$ *que si ce nombre est défini*).

Définition 6 (Fonction MT-calculable)

Soit $f : D \subseteq X^* \rightarrow X^*$ une fonction et $M = (Q, X, \varphi)$ une machine de Turing. M calcule f si :

- il existe une unique transition de q_0 et que sa forme est $\varphi(q_0, \varepsilon) = (q_i, \varepsilon, D)$ avec $q_i \neq q_0$ (*il faut quitter l'état initial à partir de* ε) ;
- il n'existe pas de transition de la forme $\varphi(q_i, x) = (q_0, x', m)$ avec $i \neq 0$, $(x, x') \in X^2$ et $m \in \{G, R, D\}$ (*il ne faut pas revenir à l'état initial*) ;
- il n'existe pas de transition de la forme $\varphi(q_f, \varepsilon)$ où $q_f \in Q_f$ (*il ne faut pas que la machine continue à fonctionner dans son état final en présence d'un blanc car il y en a une infinité et la machine ne s'arrêterait jamais*) ;
- pour tout $\mu in D$, notons $\nu = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \varepsilon \mu \varepsilon$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \varepsilon \nu \varepsilon$ (*la machine doit calculer* $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (*la machine ne calcule* $f(\mu)$ *que si ce nombre est défini*).

Nous définirons en TD :

- la fonction identiquement nulle ;
- la fonction addition ;
- les fonctions projections :

$$\pi_i^{(n)}(x_1, x_2, ..., x_n) = x_i, \quad 1 \leqslant i \leqslant n$$

Nous définirons en TD :

- la fonction identiquement nulle ;
- la fonction addition ;
- les fonctions projections :

$$\pi_i^{(n)}(x_1, x_2, ..., x_n) = x_i, \quad 1 \leqslant i \leqslant n$$

Nous définirons en TD :

- la fonction identiquement nulle ;
- la fonction addition ;
- les fonctions projections :

$$\pi_i^{(n)}(x_1, x_2, ..., x_n) = x_i, \quad 1 \leqslant i \leqslant n$$

Sommaire

Sommaire

Les fonctions de base des fonctions primitives récursives sont les fonctions précédemment étudiées :

- la fonction successeur s : $s(x) = x + 1$;
- la fonction identiquement nulle z : $z(x) = 0$;
- les fonctions projection $\pi_i^{(n)}$.

Nous appellerons fonction arithmétique une fonction de $\mathbb{N}^k$ dans $\mathbb{N}$.

Les fonctions de base des fonctions primitives récursives sont les fonctions précédemment étudiées :

- la fonction successeur s : $s(x) = x + 1$;
- la fonction identiquement nulle z : $z(x) = 0$;
- les fonctions projection $\pi_i^{(n)}$.

Nous appellerons **fonction arithmétique** une fonction de $\mathbb{N}^k$ dans $\mathbb{N}$.

Les fonctions de base des fonctions primitives récursives sont les fonctions précédemment étudiées :

- la fonction successeur s : $s(x) = x + 1$;
- la fonction identiquement nulle z : $z(x) = 0$;
- les fonctions projection $\pi_i^{(n)}$.

Nous appellerons **fonction arithmétique** une fonction de $\mathbb{N}^k$ dans $\mathbb{N}$.

Les fonctions de base des fonctions primitives récursives sont les fonctions précédemment étudiées :

- la fonction successeur s : $s(x) = x + 1$;
- la fonction identiquement nulle z : $z(x) = 0$;
- les fonctions projection $\pi_i^{(n)}$.

Nous appellerons **fonction arithmétique** une fonction de $\mathbb{N}^k$ dans $\mathbb{N}$.

Les fonctions de base des fonctions primitives récursives sont les fonctions précédemment étudiées :

- la fonction successeur s : $s(x) = x + 1$;
- la fonction identiquement nulle z : $z(x) = 0$;
- les fonctions projection $\pi_i^{(n)}$.

Nous appellerons **fonction arithmétique** une fonction de $\mathbb{N}^k$ dans $\mathbb{N}$.

Définition 7 (Composition de fonctions arithmétiques)

Soient $g_1, g_2, ..., g_k$ k fonctions arithmétiques de k variables et h une fonction arithmétique de k variables. Soit f la fonction définie par :

$$f(x_1, x_2, ..., x_n) = h(g_1(x_1, ..., x_n), ..., g_k(x_1, ..., x_n))$$

alors f est appelée la **composition** de h et de $g_1, g_2, ..., g_n$ et se note :

$$f = h \circ (g_1, g_2, ..., g_n)$$

Exemple 8

Par exemple, si $h(x_1, x_2) = s(x_1) + s(x_2)$, $g_1(x) = 2x$ et $g_2(x) = x^2 + 37$ alors :

$$f(x) =$$

Définition 9 (Récurrence)

Soient g et h des fonctions arithmétiques totales de n et $n+2$ variables respectivement. La fonction f de $n+1$ variables définie par :

- $f(x_1, ..., x_n, 0) = g(x_1, ..., x_n)$;
- $f(x_1, x_n, y+1) = h(x_1, ..., x_n, y, f(x_1, x_2, ..., x_n, y))$,

est appelée **récurrence de base g et de pas h**.
On conviendra qu'une fonction de zéro variable est constante.

Définition 9 (Récurrence)

Soient g et h des fonctions arithmétiques totales de n et $n+2$ variables respectivement. La fonction f de $n+1$ variables définie par :

- $f(x_1, ..., x_n, 0) = g(x_1, ..., x_n)$;
- $f(x_1, x_n, y+1) = h(x_1, ..., x_n, y, f(x_1, x_2, ..., x_n, y))$,

est appelée **récurrence de base g et de pas h**.
On conviendra qu'une fonction de zéro variable est constante.

Définition 9 (Récurrence)

Soient g et h des fonctions arithmétiques totales de n et $n+2$ variables respectivement. La fonction f de $n+1$ variables définie par :

- $f(x_1,...,x_n,0) = g(x_1,...,x_n)$;
- $f(x_1,x_n,y+1) = h(x_1,...,x_n,y,f(x_1,x_2,...,x_n,y))$,

est appelée **récurrence de base g et de pas h**.
On conviendra qu'une fonction de zéro variable est constante.

Définition 10 (Fonctions récursives primitives)

Une fonction est **primitive récursive** si elle peut être obtenue à partir des fonctions *successeur*, *zéro* et *projection* par l'application d'un nombre fini de compositions et de récurrences.

Exemple 11 (La fonction add)

Il est possible de définir l'addition de deux entiers à partir de la fonction s, des projections $\pi_1^{(1)}$ et $\pi_3^{(3)}$ et d'une récurrence de base $g(x) = \pi_1^{(1)}(x)$ et de pas $h = s \circ \pi_3^{(3)}$:

$$\begin{cases} \text{add}(m,0) = g(m) = m \\ \text{add}(m,n+1) = h(m,n,add(m,n)) = s(add(m,n)) \end{cases}$$

Exemple 11 (La fonction add)

Il est possible de définir l'addition de deux entiers à partir de la fonction s, des projections $\pi_1^{(1)}$ et $\pi_3^{(3)}$ et d'une récurrence de base $g(x) = \pi_1^{(1)}(x)$ et de pas $h = s \circ \pi_3^{(3)}$:

$$\begin{cases} \text{add}(m,0) = g(m) = m \\ \text{add}(m,n+1) = h(m,n,add(m,n)) = s(add(m,n)) \end{cases}$$

Exemple 11 (La fonction add)

Il est possible de définir l'addition de deux entiers à partir de la fonction s, des projections $\pi_1^{(1)}$ et $\pi_3^{(3)}$ et d'une récurrence de base $g(x) = \pi_1^{(1)}(x)$ et de pas $h = s \circ \pi_3^{(3)}$:

$$\begin{cases} \text{add}(m,0) = g(m) = m \\ \text{add}(m,n+1) = h(m,n,add(m,n)) = s(add(m,n)) \end{cases}$$

Cependant, certaines fonctions ne le sont pas comme par exemple la fonction d'Ackermann :

$$\begin{cases} A(0,y) = y+1 \\ A(x+1,0) = A(x,1) \\ A(x+1,y+1) = A(x,A(x+1,y)) \end{cases}$$

Sommaire

- On pourrait généraliser en définissant les fonctions récursives.
- Alors les fonctions MT-calculables sont les fonctions récursives.
- « un être humain muni d'un papier, d'un crayon, d'un ruban et soumis à une stricte discipline est en fait une machine universelle ».
- Comme tous nos ordinateurs *actuels* sont équivalents à des machines de Turing, ils sont donc également des modèles d'êtres humains en train d'effectuer des calculs de manière disciplinée mais non intelligente.
- Le programmeur est lui-même une machine de Turing s'il a écrit son programme (et prouvé qu'il fonctionne...).
- Un langage est une machine de Turing.

- On pourrait généraliser en définissant les fonctions récursives.
- Alors les fonctions MT-calculables sont les fonctions récursives.
- « un être humain muni d'un papier, d'un crayon, d'un ruban et soumis à une stricte discipline est en fait une machine universelle ».
- Comme tous nos ordinateurs *actuels* sont équivalents à des machines de Turing, ils sont donc également des modèles d'êtres humains en train d'effectuer des calculs de manière disciplinée mais non intelligente.
- Le programmeur est lui-même une machine de Turing s'il a écrit son programme (et prouvé qu'il fonctionne...).
- Un langage est une machine de Turing.

- On pourrait généraliser en définissant les fonctions récursives.
- Alors les fonctions MT-calculables sont les fonctions récursives.
- « un être humain muni d'un papier, d'un crayon, d'un ruban et soumis à une stricte discipline est en fait une machine universelle ».
- Comme tous nos ordinateurs *actuels* sont équivalents à des machines de Turing, ils sont donc également des modèles d'êtres humains en train d'effectuer des calculs de manière disciplinée mais non intelligente.
- Le programmeur est lui-même une machine de Turing s'il a écrit son programme (et prouvé qu'il fonctionne...).
- Un langage est une machine de Turing.

- On pourrait généraliser en définissant les fonctions récursives.
- Alors les fonctions MT-calculables sont les fonctions récursives.
- « un être humain muni d'un papier, d'un crayon, d'un ruban et soumis à une stricte discipline est en fait une machine universelle ».
- Comme tous nos ordinateurs *actuels* sont équivalents à des machines de Turing, ils sont donc également des modèles d'êtres humains en train d'effectuer des calculs de manière disciplinée mais non intelligente.
- Le programmeur est lui-même une machine de Turing s'il a écrit son programme (et prouvé qu'il fonctionne...).
- Un langage est une machine de Turing.

- On pourrait généraliser en définissant les fonctions récursives.
- Alors les fonctions MT-calculables sont les fonctions récursives.
- « un être humain muni d'un papier, d'un crayon, d'un ruban et soumis à une stricte discipline est en fait une machine universelle ».
- Comme tous nos ordinateurs *actuels* sont équivalents à des machines de Turing, ils sont donc également des modèles d'êtres humains en train d'effectuer des calculs de manière disciplinée mais non intelligente.
- Le programmeur est lui-même une machine de Turing s'il a écrit son programme (et prouvé qu'il fonctionne...).
- Un langage est une machine de Turing.

- On pourrait généraliser en définissant les fonctions récursives.
- Alors les fonctions MT-calculables sont les fonctions récursives.
- « un être humain muni d'un papier, d'un crayon, d'un ruban et soumis à une stricte discipline est en fait une machine universelle ».
- Comme tous nos ordinateurs *actuels* sont équivalents à des machines de Turing, ils sont donc également des modèles d'êtres humains en train d'effectuer des calculs de manière disciplinée mais non intelligente.
- Le programmeur est lui-même une machine de Turing s'il a écrit son programme (et prouvé qu'il fonctionne...).
- Un langage est une machine de Turing.

- Cependant, les machines de Turing sont des modèles théoriques (ruban infini) et donc éloignés du quotidien du programmeur.
- Savoir qu'il existe un algorithme qui résout notre problème est une chose, connaître sa complexité en est une autre...

- Cependant, les machines de Turing sont des modèles théoriques (ruban infini) et donc éloignés du quotidien du programmeur.
- Savoir qu'il existe un algorithme qui résout notre problème est une chose, connaître sa complexité en est une autre...

- Est-ce qu'une machine sera un jour capable de raisonner comme un cerveau humain ?
- *Science is what we understand well enough to explain to a computer. Art is everything else we do.*

- Est-ce qu'une machine sera un jour capable de raisonner comme un cerveau humain ?
- *Science is what we understand well enough to explain to a computer. Art is everything else we do.*

Sommaire

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;
- les bases de données, les moteurs de recherche ;
- etc.

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;
- les bases de données, les moteurs de recherche ;
- etc.

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;
- les bases de données, les moteurs de recherche ;
- etc.

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;
- les bases de données, les moteurs de recherche ;
- etc.

Sommaire

LA JEUNE FILLE ÉTUDIE LE BEAU COURS.

L'*alphabet des symboles* est :

$$A = \{\text{LA, JEUNE, FILLE, ÉTUDIE, LE, BEAU, COURS, .}\}$$

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > →

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

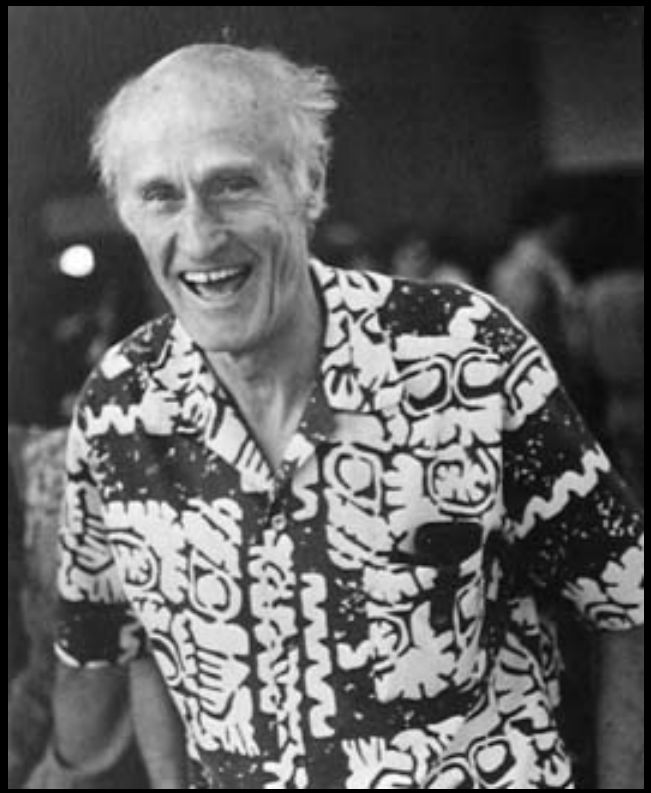
1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

1. <phrase > → <sujet ><verbe ><complément ><ponctuation >
2. <sujet > → <groupe nominal >
3. <complément > → <groupe nominal >
4. <groupe nominal > → <article féminin ><adjectif féminin ><nom féminin >
5. <groupe nominal > → <article féminin ><nom féminin ><adjectif féminin >
6. <groupe nominal > → <article féminin ><nom féminin >
7. <groupe nominal > → <article masculin ><adjectif masculin ><nom masculin >
8. <groupe nominal > → <article masculin ><nom masculin ><adjectif masculin >
9. <groupe nominal > → <article masculin ><nom masculin >
10. <article masculin > → LE
11. <article féminin > → LA
12. <adjectif féminin > → JEUNE
13. <adjectif masculin > → JEUNE
14. <adjectif masculin > → BEAU
15. <nom féminin > → FILLE
16. <nom masculin > → COURS
17. <verbe > → ÉTUDIE
18. <ponctuation > → .

LE JEUNE COURS ÉTUDIE LA FILLE.
LA FILLE ÉTUDIE LE JEUNE COURS.
LA FILLE ÉTUDIE LE COURS.


Amnesty
International



Sommaire

Définition 12

Un *alphabet* est un ensemble fini non vide. Les éléments sont appelés des *symboles* ou des *caractères*.

Définition 13

Soit A un alphabet. Soit λ_A la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

1. $\lambda_A \in A^*$;
2. si $a \in A$ et $c \in A^*$, alors la chaîne construite à partir de c en concaténant a à droite est une chaîne de A^* ;
3. μ est un élément de A^* seulement s'il peut être obtenu à partir de λ_A par application de l'étape 2 un nombre fini de fois.

Définition 13

Soit A un alphabet. Soit λ_A la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

1. $\lambda_A \in A^*$;
2. si $a \in A$ et $c \in A^*$, alors la chaîne construite à partir de c en concaténant a à droite est une chaîne de A^* ;
3. μ est un élément de A^* seulement s'il peut être obtenu à partir de λ_A par application de l'étape 2 un nombre fini de fois.

Définition 13

Soit A un alphabet. Soit λ_A la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

1. $\lambda_A \in A^*$;
2. si $a \in A$ et $c \in A^*$, alors la chaîne construite à partir de c en concaténant a à droite est une chaîne de A^* ;
3. μ est un élément de A^* seulement s'il peut être obtenu à partir de λ_A par application de l'étape 2 un nombre fini de fois.

Définition 13

Soit A un alphabet. Soit λ_A la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

1. $\lambda_A \in A^*$;
2. si $a \in A$ et $c \in A^*$, alors la chaîne construite à partir de c en concaténant a à droite est une chaîne de A^* ;
3. μ est un élément de A^* seulement s'il peut être obtenu à partir de λ_A par application de l'étape 2 un nombre fini de fois.

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Définition 14

- La *longueur* d'une chaîne est le nombre de symboles qui la composent.
- On note souvent $|\mu|$ la longueur de la chaîne μ. En particulier $|\lambda_A| = 0$.
- On note $|\mu|_a$ le nombre d'occurrences du symbole a dans la chaîne μ.
- On note A^n l'ensemble de toutes les chaînes de longueur n.
- Par convention, $A^0 = \{\lambda_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geqslant 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n > 0} A^n$: c'est l'ensemble de toutes les chaînes non vides.
- La concaténation des chaînes μ et ν sera notée $\mu\nu$.

Conventions

Monoïde libre

$$a^n = \underbrace{aa\ldots a}_{n \text{ fois}};\ a^n b^k = \underbrace{aa\ldots a}_{n \text{ fois}}\underbrace{bb\ldots b}_{k \text{ fois}};\ (ab)^n = \underbrace{abab\ldots ab}_{n \text{ fois } ab}$$

Définition 15

- On dit que $\mu \in A^*$ est un *préfixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \mu\xi$. Si $\xi \neq \lambda$, on dit que μ est un *préfixe propre* de ν.
- On dit que $\mu \in A^*$ est un *suffixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \xi\mu$. Si $\xi \neq \lambda$, on dit que μ est un *suffixe propre* de ν.
- On dit que $\mu \in A^*$ est un *facteur* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ et $xi' \in A^*$ telles que $\nu = \xi\mu\xi'$. Si $(\xi, \xi') \neq (\lambda, \lambda)$, on dit que μ est un *facteur propre* de ν.
- Pour toutes chaînes μ et ν de A^*, le quotient à gauche de ν par μ est noté $\mu^{-1}\nu$ et défini par :

$$\mu^{-1}\nu = \begin{cases} \xi \text{ si } \nu = \mu\xi \\ \text{n'a pas de sens si } \mu \text{ n'est pas un préfixe de } \nu \end{cases}$$

- On définit de même le quotient à droite.

Définition 15

- On dit que $\mu \in A^*$ est un *préfixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \mu\xi$. Si $\xi \neq \lambda$, on dit que μ est un *préfixe propre* de ν.
- On dit que $\mu \in A^*$ est un *suffixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \xi\mu$. Si $\xi \neq \lambda$, on dit que μ est un *suffixe propre* de ν.
- On dit que $\mu \in A^*$ est un *facteur* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ et $xi' \in A^*$ telles que $\nu = \xi\mu\xi'$. Si $(\xi, \xi') \neq (\lambda, \lambda)$, on dit que μ est un *facteur propre* de ν.
- Pour toutes chaînes μ et ν de A^*, le quotient à gauche de ν par μ est noté $\mu^{-1}\nu$ et défini par :

$$\mu^{-1}\nu = \begin{cases} \xi \text{ si } \nu = \mu\xi \\ \text{n'a pas de sens si } \mu \text{ n'est pas un préfixe de } \nu \end{cases}$$

On définit de même le quotient à droite.

Définition 15

- On dit que $\mu \in A^*$ est un *préfixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \mu\xi$. Si $\xi \neq \lambda$, on dit que μ est un *préfixe propre* de ν.
- On dit que $\mu \in A^*$ est un *suffixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \xi\mu$. Si $\xi \neq \lambda$, on dit que μ est un *suffixe propre* de ν.
- On dit que $\mu \in A^*$ est un *facteur* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ et $xi' \in A^*$ telles que $\nu = \xi\mu\xi'$. Si $(\xi, \xi') \neq (\lambda, \lambda)$, on dit que μ est un *facteur propre* de ν.
- Pour toutes chaînes μ et ν de A^*, le quotient à gauche de ν par μ est noté $\mu^{-1}\nu$ et défini par :

$$\mu^{-1}\nu = \begin{cases} \xi \text{ si } \nu = \mu\xi \\ \text{n'a pas de sens si } \mu \text{ n'est pas un préfixe de } \nu \end{cases}$$

On définit de même le quotient à droite.

Définition 15

- On dit que $\mu \in A^*$ est un *préfixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \mu\xi$. Si $\xi \neq \lambda$, on dit que μ est un *préfixe propre* de ν.
- On dit que $\mu \in A^*$ est un *suffixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \xi\mu$. Si $\xi \neq \lambda$, on dit que μ est un *suffixe propre* de ν.
- On dit que $\mu \in A^*$ est un *facteur* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ et $xi' \in A^*$ telles que $\nu = \xi\mu\xi'$. Si $(\xi, \xi') \neq (\lambda, \lambda)$, on dit que μ est un *facteur propre* de ν.
- Pour toutes chaînes μ et ν de A^*, le quotient à gauche de ν par μ est noté $\mu^{-1}\nu$ et défini par :

$$\mu^{-1}\nu = \begin{cases} \xi \text{ si } \nu = \mu\xi \\ \text{n'a pas de sens si } \mu \text{ n'est pas un préfixe de } \nu \end{cases}$$

On définit de même le quotient à droite.

Définition 15

- On dit que $\mu \in A^*$ est un *préfixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \mu\xi$. Si $\xi \neq \lambda$, on dit que μ est un *préfixe propre* de ν.
- On dit que $\mu \in A^*$ est un *suffixe* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ telle que $\nu = \xi\mu$. Si $\xi \neq \lambda$, on dit que μ est un *suffixe propre* de ν.
- On dit que $\mu \in A^*$ est un *facteur* de $\nu \in A^*$ si, et seulement si, il existe $\xi \in A^*$ et $xi' \in A^*$ telles que $\nu = \xi\mu\xi'$. Si $(\xi, \xi') \neq (\lambda, \lambda)$, on dit que μ est un *facteur propre* de ν.
- Pour toutes chaînes μ et ν de A^*, le quotient à gauche de ν par μ est noté $\mu^{-1}\nu$ et défini par :

$$\mu^{-1}\nu = \begin{cases} \xi \text{ si } \nu = \mu\xi \\ \text{n'a pas de sens si } \mu \text{ n'est pas un préfixe de } \nu \end{cases}$$

On définit de même le quotient à droite.

Définition 16

Une chaîne μ étant une suite (finie) de symboles, on appelle *sous-chaîne* de μ toute sous-suite de la suite μ. Tout facteur de μ est une sous-chaîne de μ mais la réciproque est fausse, un sous mot de μ contient des lettres de μ dans le bon ordre mais pas forcément de façon consécutive.

Définition 17

Si $\mu = a_{i_1} a_{i_2} \dots a_{i_n}$ est une chaîne de A^*, sa *transposée* (ou son image miroir) est la chaîne

$$^t\mu = a_{i_n} a_{i_{n-1}} \dots a_{i_1}$$

Un mot qui est égal à son transposé est appelé *palindrome*.

$$^t(\mu\nu) = {}^t\nu\,{}^t\mu$$

Sommaire

Définition 18

Un *langage* sur A est un sous-ensemble de A^*.

Théorème 19

Si L et L' sont deux langages sur l'alphabet A (rappel, L et L' sont deux parties de A^*) alors $L \cup L'$, $L \cap L'$, $L - L'$ et $\complement_{A^*} L = \overline{L}$ sont des langages sur A.

Définition 20

Si L et L' sont deux langages sur A, on définit le produit du langage L par le langage L' (attention à l'ordre) le langage noté $L.L'$ ou LL'défini par

$$LL' = \{\mu.\mu' \mid \mu \in L \text{ et } \mu' \in L'\}$$

Définition 21

On définit récursivement le langage L^n :

$$\begin{cases} L^0 = \{\lambda\} \text{ par convention (même si } L = \varnothing) \\ L^{n+1} = L^n L \end{cases}$$

Pratiquement L^n n'est autre que l'ensemble des chaînes de longueur n construites avec l'« alphabet » L. (on accepte cette vision même si L est infini).

Définition 21

On définit récursivement le langage L^n :

$$\begin{cases} L^0 = \{\lambda\} \text{ par convention (même si } L = \varnothing) \\ L^{n+1} = L^n L \end{cases}$$

Pratiquement L^n n'est autre que l'ensemble des chaînes de longueur n construites avec l'« alphabet » L. (on accepte cette vision même si L est infini).

Définition 21

On définit récursivement le langage L^n :

$$\begin{cases} L^0 = \{\lambda\} \text{ par convention (même si } L = \varnothing) \\ L^{n+1} = L^n L \end{cases}$$

Pratiquement L^n n'est autre que l'ensemble des chaînes de longueur n construites avec l'« alphabet » L. (on accepte cette vision même si L est infini).

Définition 22

Si $n \in \mathbb{N}^*$, $L^{<n}$ désigne le langage

$$L^{<n} = \bigcup_{i=0}^{n-1} L^i = L^0 \cup L^1 \cup \cdots \cup L^{n-1}$$

Pratiquement $L^{<n}$ est l'ensemble des mots de longueur $< n$ que l'on peut construire avec l'alphabet L.

Définition 22

Si $n \in \mathbb{N}^*$, $L^{<n}$ désigne le langage

$$L^{<n} = \bigcup_{i=0}^{n-1} L^i = L^0 \cup L^1 \cup \cdots \cup L^{n-1}$$

Pratiquement $L^{<n}$ est l'ensemble des mots de longueur $< n$ que l'on peut construire avec l'alphabet L.

Définition 22

Si $n \in \mathbb{N}^*$, $L^{<n}$ désigne le langage

$$L^{<n} = \bigcup_{i=0}^{n-1} L^i = L^0 \cup L^1 \cup \cdots \cup L^{n-1}$$

Pratiquement $L^{<n}$ est l'ensemble des mots de longueur $< n$ que l'on peut construire avec l'alphabet L.

Définition 23

On appelle **étoile de Kleene** ou **itération** du langage L le langage

$$L^* = \bigcup_{k=0}^{+\infty} L^k = \bigcup_{k \in \mathbb{N}} L^k$$

L'union de toutes les puissances strictement positives de L étant notée L^+ :

$$L^+ = \bigcup_{k \in \mathbb{N}^*} L^k \text{ et } L^* = L^+ \cup \{\lambda\}$$

Sommaire

Sommaire

Définition 24

Soit L un langage sur l'alphabet A. Il est dit rationnel s'il peut être obtenu récursivement uniquement à l'aide des opérations :

- réunion (somme),
- concaténation (produit),
- étoile de Kleene

en partant :

- du langage vide $\emptyset$,
- des langages du type $\{a\}$ avec $a \in A \cup \{\lambda_A\}$

Définition 24

Soit L un langage sur l'alphabet A. Il est dit rationnel s'il peut être obtenu récursivement uniquement à l'aide des opérations :

- réunion (somme),
- concaténation (produit),
- étoile de Kleene

en partant :

- du langage vide $\emptyset$,
- des langages du type $\{a\}$ avec $a \in A \cup \{\lambda_A\}$

Définition 24

Soit L un langage sur l'alphabet A. Il est dit rationnel s'il peut être obtenu récursivement uniquement à l'aide des opérations :

- réunion (somme),
- concaténation (produit),
- étoile de Kleene

en partant :

- du langage vide $\emptyset$,
- des langages du type $\{a\}$ avec $a \in A \cup \{\lambda_A\}$.

Définition 24

Soit L un langage sur l'alphabet A. Il est dit rationnel s'il peut être obtenu récursivement uniquement à l'aide des opérations :

- réunion (somme),
- concaténation (produit),
- étoile de Kleene

en partant :

- du langage vide $\emptyset$,
- des langages du type $\{a\}$ avec $a \in A \cup \{\lambda_A\}$.

Définition 24

Soit L un langage sur l'alphabet A. Il est dit rationnel s'il peut être obtenu récursivement uniquement à l'aide des opérations :

- réunion (somme),
- concaténation (produit),
- étoile de Kleene

en partant :

- du langage vide $\emptyset$,
- des langages du type $\{a\}$ avec $a \in A \cup \{\lambda_A\}$.

Définition 24

Soit L un langage sur l'alphabet A. Il est dit rationnel s'il peut être obtenu récursivement uniquement à l'aide des opérations :

- réunion (somme),
- concaténation (produit),
- étoile de Kleene

en partant :

- du langage vide $\emptyset$,
- des langages du type $\{a\}$ avec $a \in A \cup \{\lambda_A\}$.

Théorème 25

- *Tout langage* ***fini*** *est rationnel.*
- *Si L est rationnel, alors L^* est rationnel.*
- *Si L_1 et L_2 sont rationnels, alors $L_1 + L_2$ et $L_1 L_2$ sont rationnels.*

Théorème 25

- *Tout langage* ***fini*** *est rationnel.*
- *Si L est rationnel, alors $L^\star$ est rationnel.*
- *Si L_1 et L_2 sont rationnels, alors $L_1 + L_2$ et $L_1 L_2$ sont rationnels.*

Théorème 25

- *Tout langage* ***fini*** *est rationnel.*
- *Si L est rationnel, alors $L^\star$ est rationnel.*
- *Si L_1 et L_2 sont rationnels, alors $L_1 + L_2$ et $L_1 L_2$ sont rationnels.*

Théorème 25

- *Tout langage* ***fini*** *est rationnel.*
- *Si L est rationnel, alors $L^\star$ est rationnel.*
- *Si L_1 et L_2 sont rationnels, alors $L_1 + L_2$ et $L_1 L_2$ sont rationnels.*

Sommaire

Définition 26

Soit A un alphabet. Une **expression rationnelle** définie sur A est :

- $\emptyset$,
- λ_A,
- un élément quelconque de A,
- des « formules *bien formées* » à partir des éléments de A en utilisant les opérateurs rationnels. Ainsi, si R_1 et R_2 sont des expressions rationnelles, alors $(R_1 + R_2)$, $R_1 \cdot R_2$ et R_1^* le sont aussi.

Définition 26

Soit A un alphabet. Une **expression rationnelle** définie sur A est :

- $\emptyset$,
- λ_A,
- un élément quelconque de A,
- des « formules *bien formées* » à partir des éléments de A en utilisant les opérateurs rationnels. Ainsi, si R_1 et R_2 sont des expressions rationnelles, alors $(R_1 + R_2)$, $R_1 \cdot R_2$ et R_1^* le sont aussi.

Définition 26

Soit A un alphabet. Une **expression rationnelle** définie sur A est :

- $\emptyset$,
- λ_A,
- un élément quelconque de A,
- des « formules *bien formées* » à partir des éléments de A en utilisant les opérateurs rationnels. Ainsi, si R_1 et R_2 sont des expressions rationnelles, alors $(R_1 + R_2)$, $R_1 \cdot R_2$ et $R_1^\star$ le sont aussi.

Définition 26

Soit A un alphabet. Une **expression rationnelle** définie sur A est :

- $\emptyset$,
- λ_A,
- un élément quelconque de A,
- des « formules *bien formées* » à partir des éléments de A en utilisant les opérateurs rationnels. Ainsi, si R_1 et R_2 sont des expressions rationnelles, alors $(R_1 + R_2)$, $R_1 \cdot R_2$ et $R_1^\star$ le sont aussi.

Définition 26

Soit A un alphabet. Une **expression rationnelle** définie sur A est :

- $\emptyset$,
- λ_A,
- un élément quelconque de A,
- des « formules *bien formées* » à partir des éléments de A en utilisant les opérateurs rationnels. Ainsi, si R_1 et R_2 sont des expressions rationnelles, alors $(R_1 + R_2)$, $R_1 \cdot R_2$ et $R_1^\star$ le sont aussi.

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^\star) = \mathcal{L}(E)^*$

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A,\ \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A,\ \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A,\ \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^\star) = \mathcal{L}(E)^\star$

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A$, $\mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^\star) = \mathcal{L}(E)^\star$

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A$, $\mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^\star) = \mathcal{L}(E)^\star$

Théorème 27

À chaque expression rationnelle E on fait correspondre le langage $\mathcal{L}(E)$ de $A^\star$ défini par :

- $\mathcal{L}(\varnothing) = \varnothing$
- $\mathcal{L}(\lambda_A) = \{\lambda_A\}$
- $a \in A$, $\mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E + E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) + \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E)\,\mathcal{L}(E')$
- $\mathcal{L}(E^\star) = \mathcal{L}(E)^\star$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT,\quad (S+T)R=SR+TR$.
- $R^*=R^*R^*=(R^*)^*=(\lambda+R)^*$
- $(R+S)^*=(R^*S^*)^*=(R^*+S^*)^*=(R^*S)^*R^*$
- $R^+=RR^*=R^*R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT,\quad (S+T)R=SR+TR$.
- $R^*=R^*R^*=(R^*)^*=(\lambda+R)^*$
- $(R+S)^*=(R^*S^*)^*=(R^*+S^*)^*=(R^*S)^*R^*$
- $R^+=RR^*=R^*R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR$
- $R^*=R^*R^*=(R^*)^*=(\lambda+R)^*$
- $(R+S)^*=(R^*S^*)^*=(R^*+S^*)^*=(R^*S)^*R^*$
- $R^+=RR^*=R^*R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR$
- $R^*=R^*R^*=(R^*)^*=(\lambda+R)^*$
- $(R+S)^*=(R^*S^*)^*=(R^*+S^*)^*=(R^*S)^*R^*$
- $R^+=RR^*=R^*R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^*=R^*R^*=(R^*)^*=(\lambda+R)^*$
- $(R+S)^*=(R^*S^*)^*=(R^*+S^*)^*=(R^*S)^*R^*$
- $R^+=RR^*=R^*R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^\star=R^\star R^\star=(R^\star)^\star=(\lambda+R)^\star$
- $(R+S)^\star=(R^\star S^\star)^\star=(R^\star+S^\star)^\star=(R^\star S)^\star R^\star$
- $R^+=RR^\star=R^\star R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^\star=R^\star R^\star=(R^\star)^\star=(\lambda+R)^\star$
- $(R+S)^\star=(R^\star S^\star)^\star=(R^\star+S^\star)^\star=(R^\star S)^\star R^\star$
- $R^+=RR^\star=R^\star R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^\star=R^\star R^\star=(R^\star)^\star=(\lambda+R)^\star$
- $(R+S)^\star=(R^\star S^\star)^\star=(R^\star+S^\star)^\star=(R^\star S)^\star R^\star$
- $R^+=RR^\star=R^\star R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^\star=R^\star R^\star=(R^\star)^\star=(\lambda+R)^\star$
- $(R+S)^\star=(R^\star S^\star)^\star=(R^\star+S^\star)^\star=(R^\star S)^\star R^\star$
- $R^+=RR^\star=R^\star R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^\star=R^\star R^\star=(R^\star)^\star=(\lambda+R)^\star$
- $(R+S)^\star=(R^\star S^\star)^\star=(R^\star+S^\star)^\star=(R^\star S)^\star R^\star$
- $R^+=RR^\star=R^\star R$

Théorème 28

- $R+S=S+R, R+R=R$
- $R+\emptyset=R$
- $(R+S)+T=R+(S+T)$
- $R(ST)=(RS)T$
- $R\lambda=\lambda R=R$
- $R\emptyset=\emptyset R=\emptyset$
- $R(S+T)=RS+RT, \quad (S+T)R=SR+TR.$
- $R^\star=R^\star R^\star=(R^\star)^\star=(\lambda+R)^\star$
- $(R+S)^\star=(R^\star S^\star)^\star=(R^\star+S^\star)^\star=(R^\star S)^\star R^\star$
- $R^+=RR^\star=R^\star R$

Théorème 29

Un langage de $A^\star$ est rationnel si, et seulement si, il est dénoté par une expression rationnelle.

Sommaire

Sommaire

Définition 30

Un automate $\mathcal{A}$ est caractérisé par un quintuplet $< A, Q, I, T, \varphi$:

- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\lambda\})$ vers Q, appelée *relation de transition*.

Définition 30

Un automate $\mathcal{A}$ est caractérisé par un quintuplet $< A, Q, I, T, \varphi >$:

- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\lambda\})$ vers Q, appelée *relation de transition*.

Définition 30

Un automate $\mathcal{A}$ est caractérisé par un quintuplet $< A, Q, I, T, \varphi$:

- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\lambda\})$ vers Q, appelée *relation de transition*.

Définition 30

Un automate $\mathcal{A}$ est caractérisé par un quintuplet $< A, Q, I, T, \varphi$:

- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\lambda\})$ vers Q, appelée *relation de transition*.

Définition 30

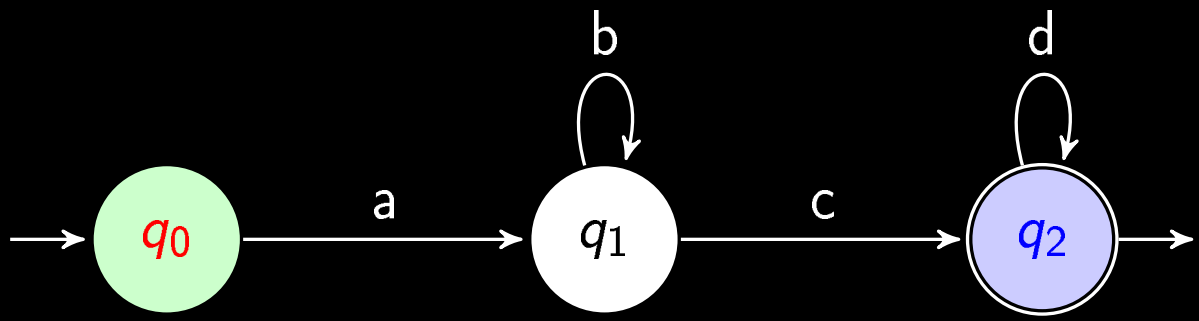
Un automate $\mathcal{A}$ est caractérisé par un quintuplet $< A, Q, I, T, \varphi$:

- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\lambda\})$ vers Q, appelée *relation de transition*.

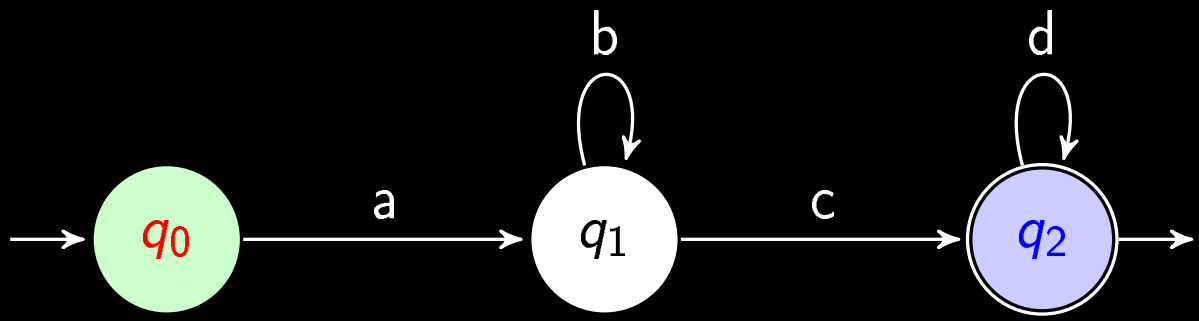
Définition 30

Un automate $\mathcal{A}$ est caractérisé par un quintuplet $< A, Q, I, T, \varphi$:

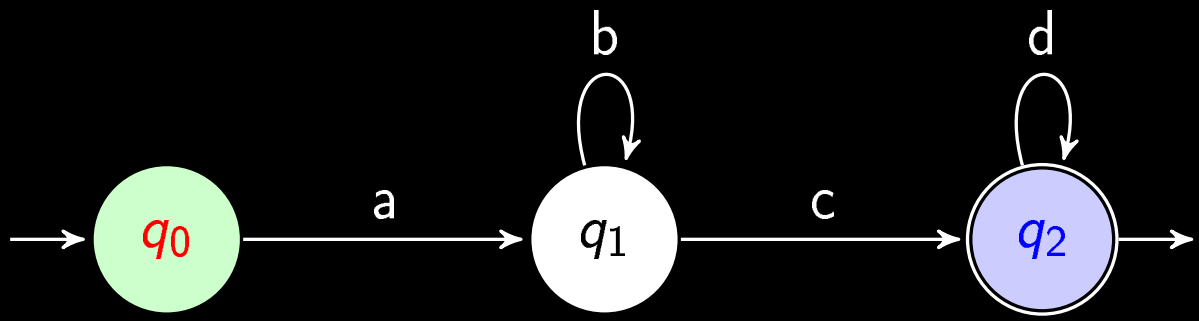
- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\lambda\})$ vers Q, appelée *relation de transition*.



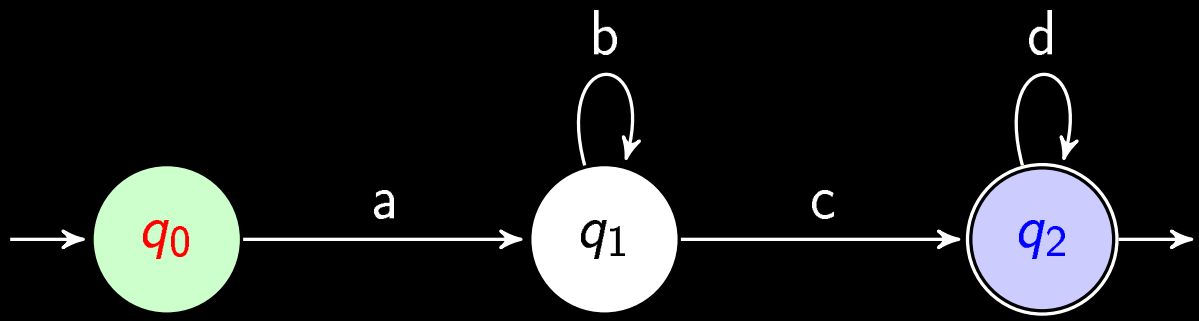
- transition
- étiquette
- origine
- extrémité terminale



- transition
- étiquette
- origine
- extrémité terminale



- transition
- étiquette
- origine
- extrémité terminale



- transition
- étiquette
- origine
- extrémité terminale

Table de transition

μ	q_0	q_1	q_2
a	q_1		
b		q_1	
c		q_2	
d			q_2

Sommaire

Un **calcul** c de l'automate $\mathcal{A}$ (ou une exécution ou une trace de l'automate $\mathcal{A}$) est une suite de transitions (des relations de $Q \times A$ dans Q) telles que l'origine de chacune (sauf la première) coïncide avec l'extrémité terminale de la précédente :

$$c = q_{i_0} \xrightarrow{\alpha_1} q_{i_1} \xrightarrow{\alpha_2} q_{i_2} \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_n} q_{i_n}$$

que l'on note aussi

$$c = q_{i_0} \xrightarrow{\alpha_1 \alpha_2 \cdots \alpha_n} q_{i_n}$$

Un calcul de $\mathcal{A}$ est **réussi** si son origine est un état initial et son extrémité terminale un état final.

Un **calcul** c de l'automate $\mathcal{A}$ (ou une exécution ou une trace de l'automate $\mathcal{A}$) est une suite de transitions (des relations de $Q \times A$ dans Q) telles que l'origine de chacune (sauf la première) coïncide avec l'extrémité terminale de la précédente :

$$c = q_{i_0} \xrightarrow{\alpha_1} q_{i_1} \xrightarrow{\alpha_2} q_{i_2} \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_n} q_{i_n}$$

que l'on note aussi

$$c = q_{i_0} \xrightarrow{\alpha_1 \alpha_2 \cdots \alpha_n} q_{i_n}$$

Un calcul de $\mathcal{A}$ est **réussi** si son origine est un état initial et son extrémité terminale un état final.

Langage reconnaissable par un automate

Définition 31

On note $\mathcal{L}(\mathcal{A})$ l'ensemble des mots reconnus par l'automate :

$$\mathcal{L}(\mathcal{A}) = \left\{ \mu \in A^\star \mid \exists q_i \in I, \exists q_t \in T, q_i \xrightarrow{\mu} q_t \right\}$$

et on dit qu'un langage L **est reconnaissable par l'automate** $\mathcal{A}$ si, et seulement si, $\mathcal{L}(\mathcal{A}) = L$.
Un langage est **reconnaissable** si, et seulement si,il existe un automate qui le reconnaît.
L'ensemble des langages reconnaissables de $A^\star$ est noté $\mathrm{Rec}(A^\star)$.

Sommaire

Méthode « manuelle »

Technique d'élimination d'états.

- On commence par créer deux nouveaux états q_d et q_t qui seront les uniques états initiaux et terminaux.
  On relie q_d à tous les états initiaux par une transition étiquetée par λ et on relie tous les états terminaux à q_t par le même type de transitions.
- Notons Q l'ensemble tel que l'alphabet d'états soit $\{q_d\} \cup Q \cup \{q_t\}$.
  Tant que Q n'est pas vide, on applique l'opération suivante : pour tout couple (p, r) d'états tels qu'il existe une transition directe de p à q et de q à r étiquetée respectivement par L_{pq} et L_{qr}, on crée une transition (p, L_{pqr}, r) avec $L_{pqr} = L_{pr} \cup L_{pq} L_{qq}^{\star} L_{qr}$.

Technique d'élimination d'états.

- On commence par créer deux nouveaux états q_d et q_t qui seront les uniques états initiaux et terminaux.
  On relie q_d à tous les états initiaux par une transition étiquetée par λ et on relie tous les états terminaux à q_t par le même type de transitions.
- Notons Q l'ensemble tel que l'alphabet d'états soit $\{q_d\} \cup Q \cup \{q_t\}$.
  Tant que Q n'est pas vide, on applique l'opération suivante : pour tout couple (p, r) d'états tels qu'il existe une transition directe de p à q et de q à r étiquetée respectivement par L_{pq} et L_{qr}, on crée une transition (p, L_{pqr}, r) avec $L_{pqr} = L_{pr} \cup L_{pq} L_{qq}^{\star} L_{qr}$.

Sommaire

Théorème 32

Automates équivalents On définit une relation $\equiv$ sur l'ensemble des automates finis.
Deux automates $\mathcal{A}$ et $\mathcal{A}'$ sont en relation si, et seulement si, ils reconnaissent le même langage.
Cette relation est une relation d'équivalence. On note $\mathcal{A} \equiv \mathcal{A}'$ pour signifier que deux automates sont équivalents.

Sommaire

Définition 33

Automate standard Un automate $< A, Q, D, T, \tau >$ est standard si, et seulement si, D est réduit à un singleton $\{d\}$ et aucune transition n'aboutit en d.

L'algorithme suivant va nous permettre de rendre standard tout automate fini. Notons $\mathcal{A}'$ l'automate standardisé, alors $\mathcal{A}' =< A, Q \cup \{d\}, \{d\}, T', \tau' >$ avec :

- $d \notin Q$;
- si $D \cap T = \emptyset$ alors $T' = T$ sinon $T' = T \cup \{d\}$;
- $\mathcal{G}_{\tau'} = \mathcal{G}_\tau \cup \{(d, a, q) \mid \exists q_d \in D, (q_d, a, q) \in \mathcal{G}_\tau\}$

L'algorithme suivant va nous permettre de rendre standard tout automate fini. Notons $\mathcal{A}'$ l'automate standardisé, alors $\mathcal{A}' =< A, Q \cup \{d\}, \{d\}, T', \tau' >$ avec :

- $d \notin Q$;
- si $D \cap T = \emptyset$ alors $T' = T$ sinon $T' = T \cup \{d\}$;
- $\mathcal{G}_{\tau'} = \mathcal{G}_\tau \cup \{(d, a, q) \mid \exists q_d \in D, (q_d, a, q) \in \mathcal{G}_\tau\}$

L'algorithme suivant va nous permettre de rendre standard tout automate fini. Notons $\mathcal{A}'$ l'automate standardisé, alors $\mathcal{A}' =< A, Q \cup \{d\}, \{d\}, T', \tau' >$ avec :

- $d \notin Q$;
- si $D \cap T = \emptyset$ alors $T' = T$ sinon $T' = T \cup \{d\}$;
- $\mathcal{G}_{\tau'} = \mathcal{G}_\tau \cup \{(d, a, q) \mid \exists q_d \in D, (q_d, a, q) \in \mathcal{G}_\tau\}$

Théorème 34

Tout automate fini est équivalent à un automate standard.

Sommaire

Définition 35 (États accessibles et co-accessibles)

Soit $\mathcal{A} =< A, Q, D, T, \tau >$ un automate fini.

- On dit que l'état q_j ($q_j \in Q$) est **accessible à partir d'un état** q_i ($q_i \in Q$) s'il existe (au moins) un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état q_j est **accessible** s'il est accessible à partir d'un état initial.
- Un état q_i est **co-accessible à un état** q_j s'il existe un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état e_i est **co-accessible** s'il est co-accessible d'un état final.
- On dit qu'un état q_i est **utile** s'il est à la fois accessible et co-accessible.

Définition 35 (États accessibles et co-accessibles)

Soit $\mathcal{A} =< A, Q, D, T, \tau >$ un automate fini.

- On dit que l'état q_j ($q_j \in Q$) est **accessible à partir d'un état** q_i ($q_i \in Q$) s'il existe (au moins) un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état q_j est **accessible** s'il est accessible à partir d'un état initial.
- Un état q_i est **co-accessible à un état** q_j s'il existe un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état e_i est **co-accessible** s'il est co-accessible d'un état final.
- On dit qu'un état q_i est **utile** s'il est à la fois accessible et co-accessible.

Définition 35 (États accessibles et co-accessibles)

Soit $\mathcal{A} =< A, Q, D, T, \tau >$ un automate fini.

- On dit que l'état q_j ($q_j \in Q$) est **accessible à partir d'un état** q_i ($q_i \in Q$) s'il existe (au moins) un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état q_j est **accessible** s'il est accessible à partir d'un état initial.
- Un état q_i est **co-accessible à un état** q_j s'il existe un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état e_i est **co-accessible** s'il est co-accessible d'un état final.
- On dit qu'un état q_i est **utile** s'il est à la fois accessible et co-accessible.

Définition 35 (États accessibles et co-accessibles)

Soit $\mathcal{A} =< A, Q, D, T, \tau >$ un automate fini.

- On dit que l'état q_j ($q_j \in Q$) est **accessible à partir d'un état** q_i ($q_i \in Q$) s'il existe (au moins) un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état q_j est **accessible** s'il est accessible à partir d'un état initial.
- Un état q_i est **co-accessible à un état** q_j s'il existe un calcul **c** dans $\mathcal{A}$ dont l'origine est q_i et l'extrémité est q_j. On dit que l'état e_i est **co-accessible** s'il est co-accessible d'un état final.
- On dit qu'un état q_i est **utile** s'il est à la fois accessible et co-accessible.

Définition 36

Automate émondé Un automate émondé est un automate dont tous les états sont utiles.

Fonction émondage aller($\mathcal{A}$: *automate fini*) : *automate fini*

Début

$E \leftarrow D$

$E' \leftarrow \emptyset$

TantQue $E' \neq E$ **Faire**

$E' \leftarrow E$

$E \leftarrow E \cup \{q \in Q \mid \exists e \in E, \exists a \in A : (e, a, q) \in \mathcal{G}_\tau\}$

FinTantQue

Retourner $< A, E, D, T, \tau' >$

Fin

Fonction émondage retour($\mathcal{A}$: *automate fini*) : *automate fini*

Début

$C \leftarrow T \cap E$

$C' \leftarrow \emptyset$

TantQue $C' \neq C$ **Faire**

$C' \leftarrow C$

$C \leftarrow C \cup \{q \in Q \mid \exists c \in C, \exists a \in A : (q, a, c) \in \mathcal{G}_\tau\}$

FinTantQue

Retourner $< A, C, D, T, \tau'' >$

Fin

Théorème 37

Tout automate fini est équivalent à un automate émondé

Sommaire

Définition 38

Un automate *déterministe* :

- a un unique état initial ;
- la relation de transition τ est une fonction : il y a, à partir de chaque état, au plus une transition d'étiquette donnée.

Si un automate n'est pas déterministe, il est...non-déterministe.

Définition 38

Un automate *déterministe* :

- a un unique état initial ;
- la relation de transition τ est une fonction : il y a, à partir de chaque état, au plus une transition d'étiquette donnée.

Si un automate n'est pas déterministe, il est...non-déterministe.

Définition 38

Un automate *déterministe* :

- a un unique état initial ;
- la relation de transition τ est une fonction : il y a, à partir de chaque état, au plus une transition d'étiquette donnée.

Si un automate n'est pas déterministe, il est...non-déterministe.

Définition 39

Un automate *déterministe complet* :

- est un automate déterministe ;
- la relation de transition τ est une application : il y a, à partir de chaque état, une et une seule transition d'étiquette donnée.

Définition 39

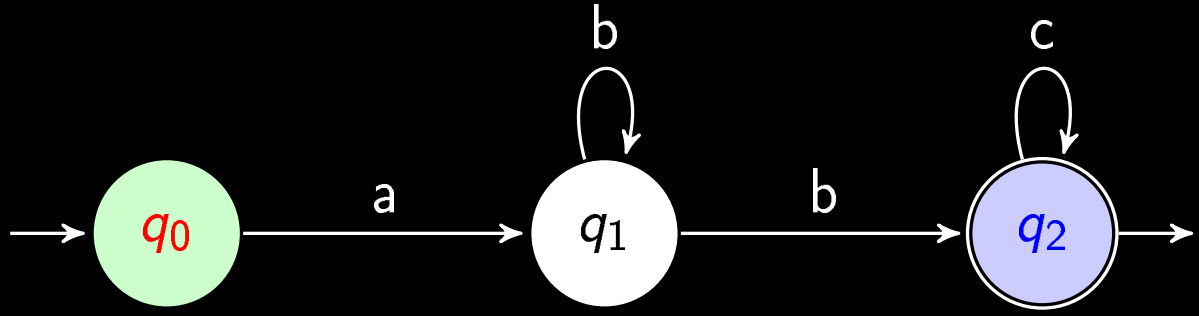
Un automate *déterministe complet* :

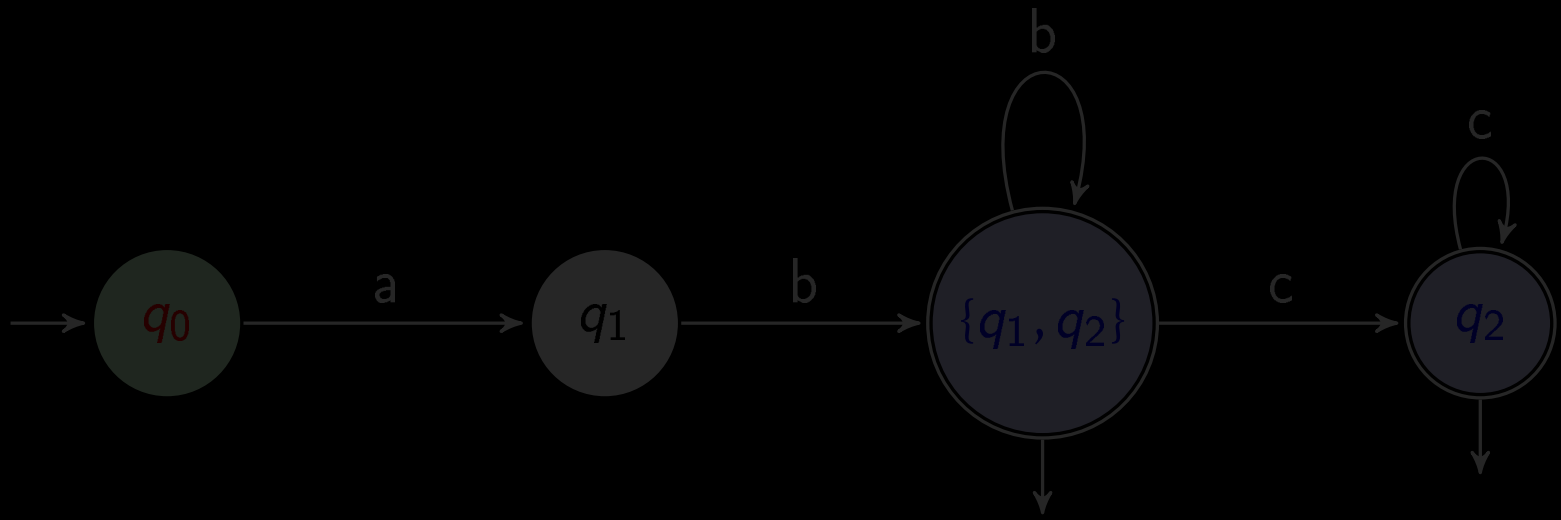
- est un automate déterministe ;
- la relation de transition τ est une application : il y a, à partir de chaque état, une et une seule transition d'étiquette donnée.

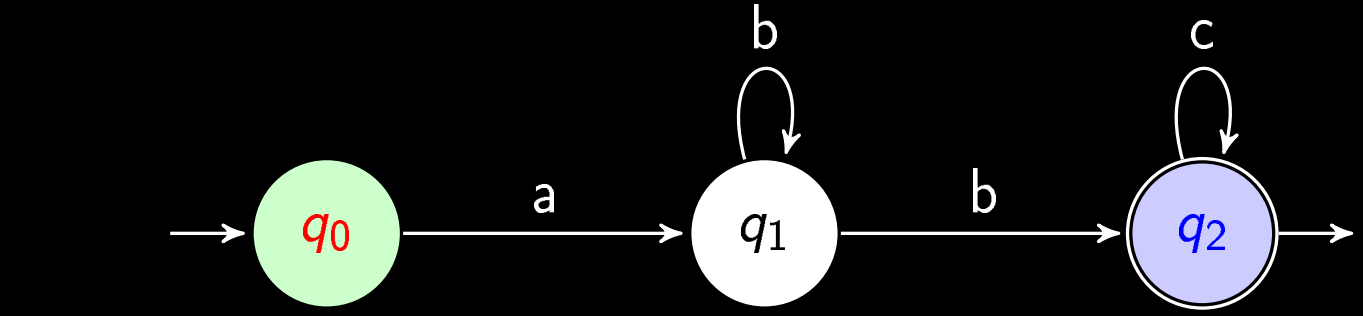
Définition 39

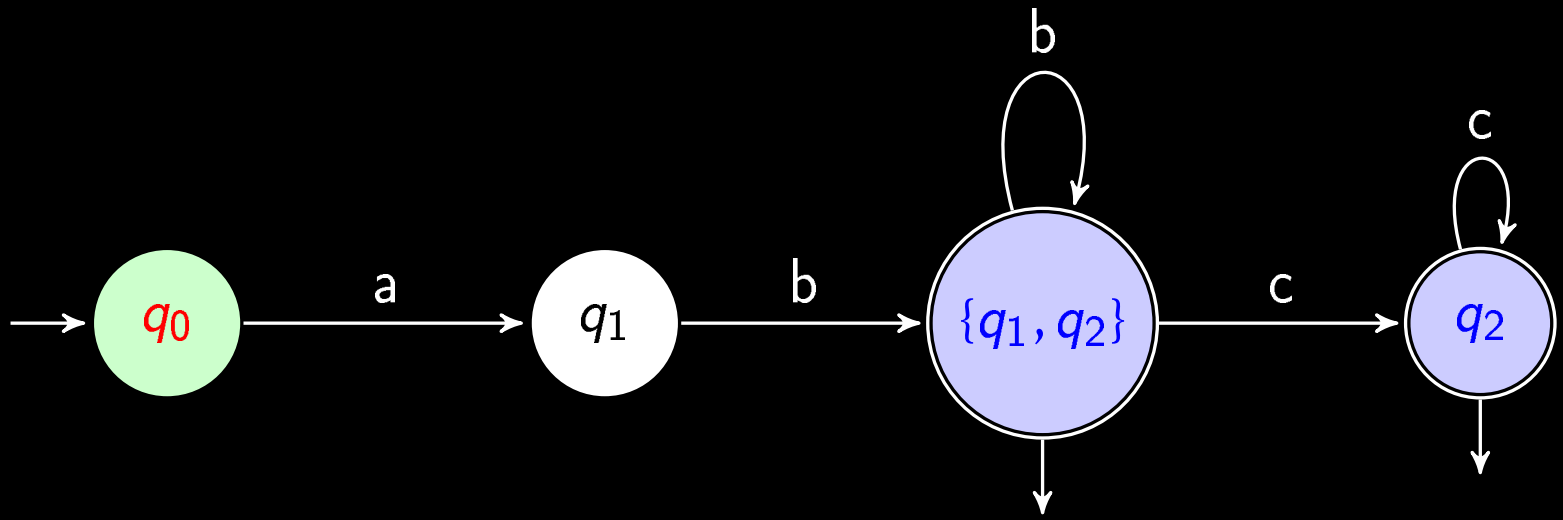
Un automate *déterministe complet* :

- est un automate déterministe ;
- la relation de transition τ est une application : il y a, à partir de chaque état, une et une seule transition d'étiquette donnée.


b
c
a
b
q_0
q_1
q_2


b
c
a
b
c
q_0
q_1
$\{q_1, q_2\}$
q_2


b
c
a
b
q_0
q_1
q_2


b
c
a
b
c
q_0
q_1
$\{q_1, q_2\}$
q_2

Théorème 40

Tout automate peut être transformé en un automate déterministe reconnaissant le même langage.

Sommaire

Sommaire

Soit $\mathcal{A}_1 =< A, Q_1, D_1, T_1, \tau_1 >$ et $\mathcal{A}_1 =< A, Q_2, D_2, T_2, \tau_2 >$ deux automates finis reconnaissant deux langages L_1 et L_2. On peut supposer que $Q_1 \cap Q_2 = \emptyset$ quitte à renuméroter les états (en effet les noms des états n'ont aucune importance). Alors on définit un nouvel automate

$$\mathcal{A}_1 + \mathcal{A}_2 =< A, Q_1 \cup Q_2, D_1 \cup D_2, T_1 \cup T_2, \tau_1 \cup \tau_2 >$$

en notant abusivement $\tau_1 \cup \tau_2$ la relation de graphe $\mathcal{G}_{\tau_1} \cup \mathcal{G}_{\tau_2}$.

Théorème 41

L'automate $\mathcal{A}_1 + \mathcal{A}_2$ reconnaît le langage $L_1 + L_2$.

Sommaire

Soit $\mathcal{A}_1 =< A, Q_1, D_1, T_1, \tau_1 >$ et $\mathcal{A}_1 =< A, Q_2, D_2, T_2, \tau_2 >$ deux automates finis reconnaissant deux langages L_1 et L_2. On peut supposer que $Q_1 \cap Q_2 = \emptyset$. Quitte à standardiser $\mathcal{A}_2$, on peut supposer que $D_2 = \{d_2\}$. On définit alors un nouvel automate :

$$\mathcal{A}_1 \cdot \mathcal{A}_2 =< A, (Q_1 \cup Q_2) \setminus \{d_2\}, D_1, T', \tau_1 \cup \tau_2' \cup \tau_2'' >$$

avec :

- $T' = (T_1 \cup T_2) \setminus \{d_2\}$ si $d_2 \in T_2$ et T_2 sinon ;
- $\mathcal{G}_{\tau_2'} = \{(q, a, q') \in GR_{\tau_2} \mid q \neq d_2\}$;
- $\mathcal{G}_{\tau_2''} = \{(q_{t_1}, a, q') \mid q_{t_1} \in T_1, (d_2, a, q') \in \mathcal{G}_{\tau_2}\}$.

Soit $\mathcal{A}_1 =< A, Q_1, D_1, T_1, \tau_1 >$ et $\mathcal{A}_1 =< A, Q_2, D_2, T_2, \tau_2 >$ deux automates finis reconnaissant deux langages L_1 et L_2. On peut supposer que $Q_1 \cap Q_2 = \emptyset$. Quitte à standardiser $\mathcal{A}_2$, on peut supposer que $D_2 = \{d_2\}$. On définit alors un nouvel automate :

$$\mathcal{A}_1 \cdot \mathcal{A}_2 =< A, (Q_1 \cup Q_2) \setminus \{d_2\}, D_1, T', \tau_1 \cup \tau_2' \cup \tau_2'' >$$

avec :

- $T' = (T_1 \cup T_2) \setminus \{d_2\}$ si $d_2 \in T_2$ et T_2 sinon ;
- $\mathcal{G}_{\tau_2'} = \{(q, a, q') \in GR_{\tau_2} \mid q \neq d_2\}$;
- $\mathcal{G}_{\tau_2''} = \{(q_{t_1}, a, q') \mid q_{t_1} \in T_1, (d_2, a, q') \in \mathcal{G}_{\tau_2}\}$.

Soit $\mathcal{A}_1 =< A, Q_1, D_1, T_1, \tau_1 >$ et $\mathcal{A}_1 =< A, Q_2, D_2, T_2, \tau_2 >$ deux automates finis reconnaissant deux langages L_1 et L_2. On peut supposer que $Q_1 \cap Q_2 = \emptyset$. Quitte à standardiser $\mathcal{A}_2$, on peut supposer que $D_2 = \{d_2\}$. On définit alors un nouvel automate :

$$\mathcal{A}_1 \cdot \mathcal{A}_2 =< A, (Q_1 \cup Q_2) \setminus \{d_2\}, D_1, T', \tau_1 \cup \tau_2' \cup \tau_2'' >$$

avec :

- $T' = (T_1 \cup T_2) \setminus \{d_2\}$ si $d_2 \in T_2$ et T_2 sinon ;
- $\mathcal{G}_{\tau_2'} = \{(q, a, q') \in GR_{\tau_2} \mid q \neq d_2\}$;
- $\mathcal{G}_{\tau_2''} = \{(q_{t_1}, a, q') \mid q_{t_1} \in T_1, (d_2, a, q') \in \mathcal{G}_{\tau_2}\}$.

Théorème 42

L'automate $\mathcal{A}_1 \cdot \mathcal{A}_2$ reconnaît le langage $L_1 \cdot L_2$.

Sommaire

Soit $\mathcal{A} =< A, Q, \{d\}, T, \tau >$ un automate fini *standard* reconnaissant un langage L.
On définit alors un nouvel automate

$$\mathcal{A}^\star =< A, Q, \{d\}, T \cup \{d\}, \tau' >$$

avec $GR_{\tau'} = \mathcal{G}_\tau \cup \{(q_t, a, q) \mid q_t \in T, (d, a, q) \in \mathcal{G}_\tau\}$

Sommaire

Théorème 43

Théorème de Kleene *(1956) Pour tout alphabet A,* $Rat(A^\star) = Rec(A^\star)$.

Sommaire

Sommaire

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite.
Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.
On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.
On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.
On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.
On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.
Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite. Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.

On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.

On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.

On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.

On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.

Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite.
Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.
On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.
On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.
On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.
On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.
Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite. Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.

On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.

On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.

On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.

On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.

Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite. Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.

On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.

On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.

On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.

On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.

Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite. Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.
On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.
On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.
On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.
On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.
Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

On *linéarise* l'expression rationnelle, i.e., on remplace tous les caractères par des symboles distincts, numérotés à partir de 1 de la gauche vers la droite. Par exemple, $ab^\star + a^2b^+$ devient $x_1x_2^\star + x_3x_4x_5^+$.
On crée ensuite un état par variable ainsi qu'un état q_0 qui représente l'état initial.
On crée l'ensemble posliedny (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si λ appartient au langage : il constitue l'ensemble des états terminaux de l'automate.
On crée l'ensemble piervy (premier en russe) des caractères pouvant commencer un mot : on relie l'état q_0 à chaque q_i de cet ensemble en l'étiquetant par x_i.
On crée une transition étiquetée par x_j de l'état q_i vers l'état q_j si le facteur x_ix_j peut apparaître dans au moins une chaîne du langage.
Finalement, on remplace les x_i par les caractères qu'ils remplaçaient.

Sommaire

Définition 44

Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.

Définition 44

Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.

Définition 44

Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.

Définition 44

Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.

Définition 44

Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.

Définition 44

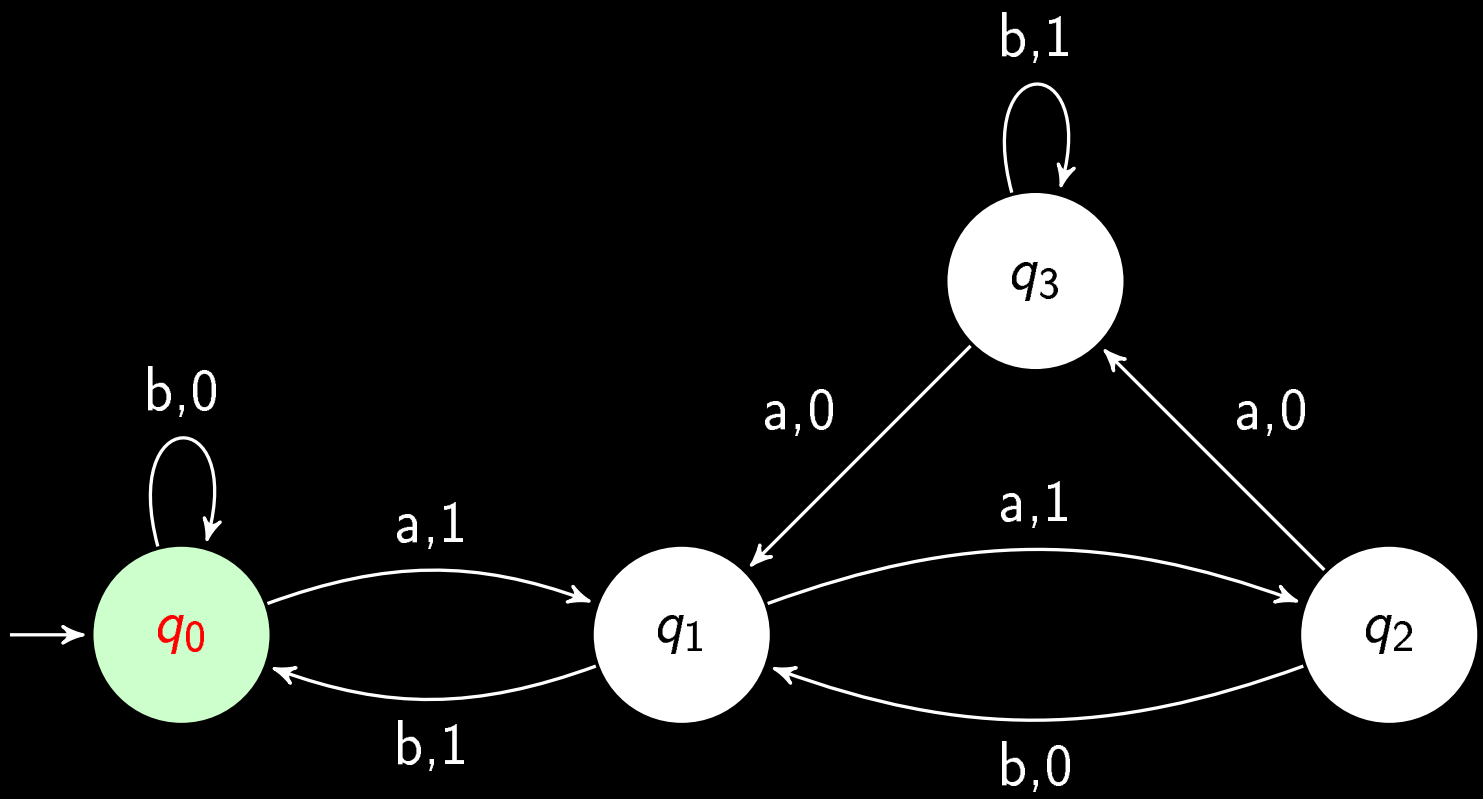
Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.

Définition 44

Un automate à états finis avec sorties est caractérisé par un sextuplet $< Q, q_0, A_I, A_S, f_t, f_s >$ où :

- Q est un ensemble fini d'états ;
- q_0 est l'état initial ;
- A_I est l'alphabet d'entrée ;
- A_S est l'alphabet de sortie ;
- f_t est la fonction de transition de $Q \times A_I$ vers Q ;
- f_s est la fonction de sortie de $Q \times A_I$ vers A_S.


b,1
q_3
b,0
a,0
a,0
a,1
a,1
q_0
q_1
q_2
b,1
b,0

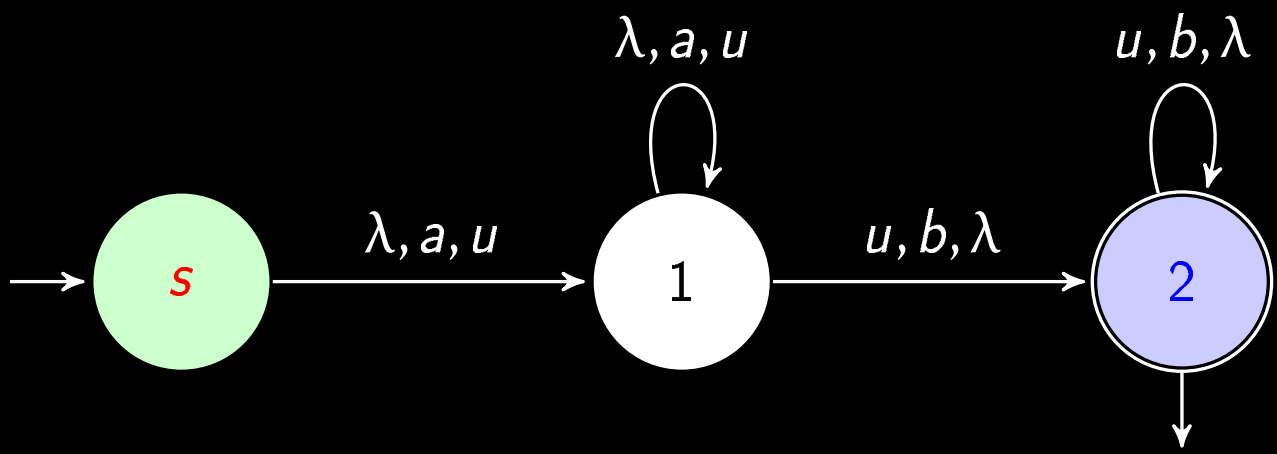
Sommaire

Considérons l'automate défini par

$$A = \{a, b\}, E = \{s, 1, 2\}, T = \{2\}, P = \{u\}$$

et les transitions

$$\begin{aligned} T_1 &: ((s, \lambda), a, (1, u)) \\ T_2 &: ((1, \lambda), a, (1, u)) \\ T_3 &: ((1, u), b, (2, \lambda)) \\ T_4 &: ((2, u), b, (2, \lambda)) \end{aligned}$$


λ, a, u
u, b, λ
λ, a, u
u, b, λ
s
1
2

Sommaire

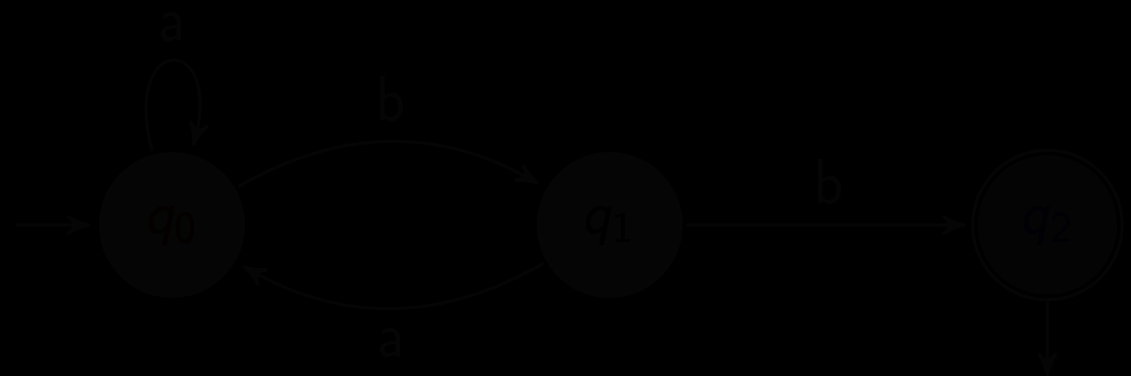
Sommaire

Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.

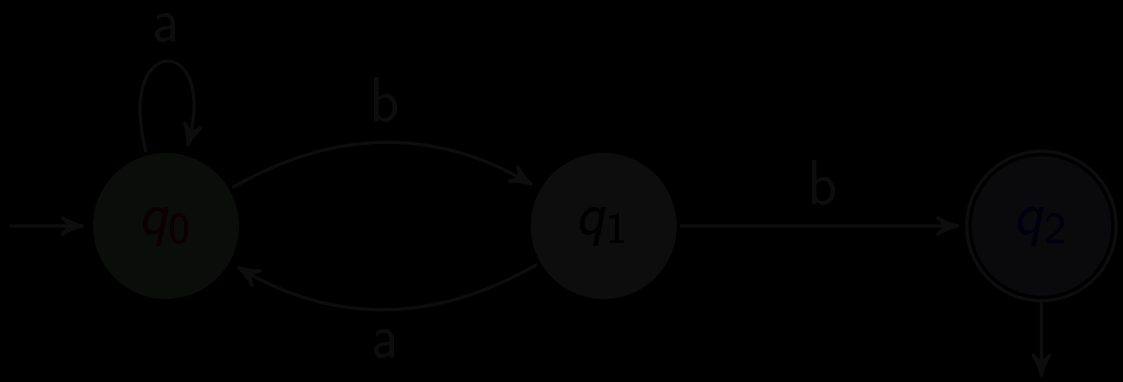


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.

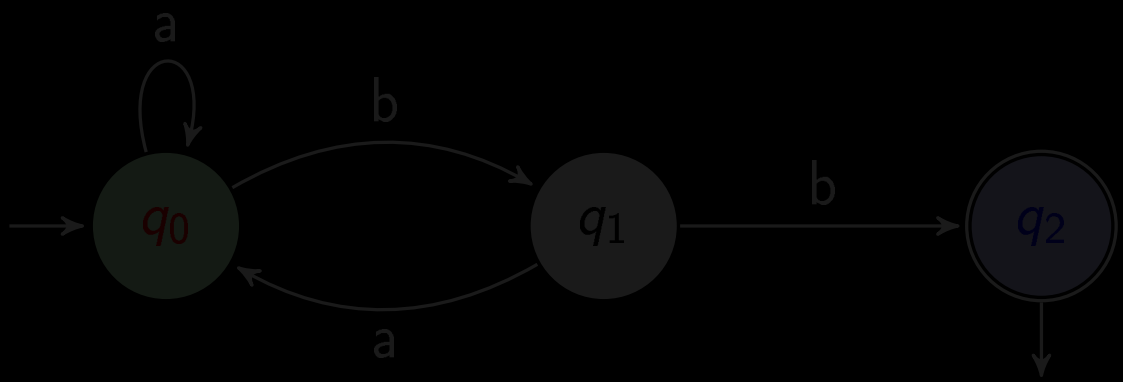


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.

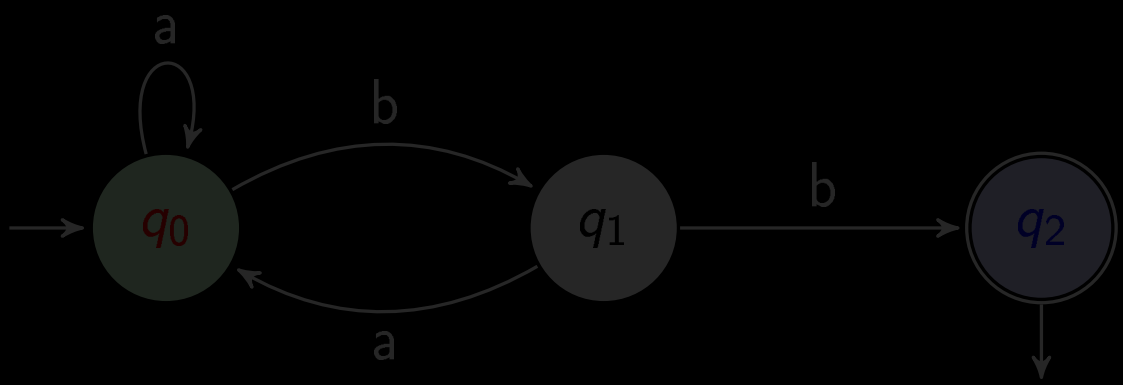


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.



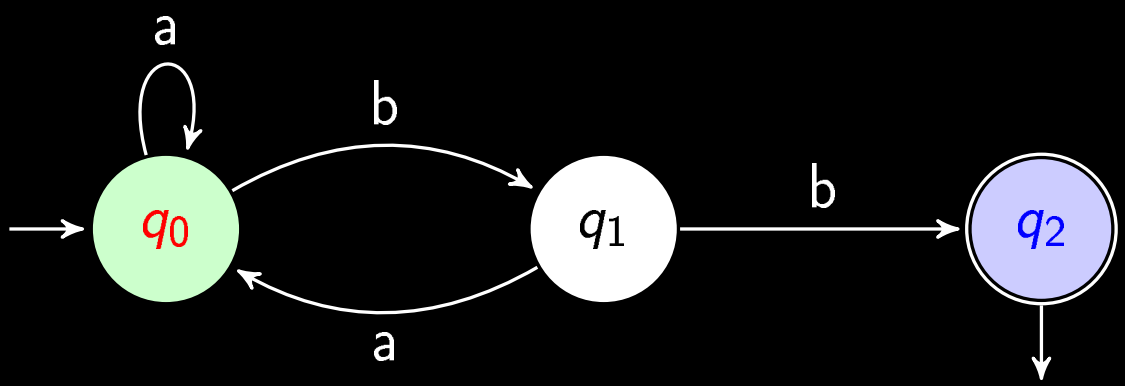
q_0 et q_1 sont-ils distinguables par a ? Par b ?

Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.



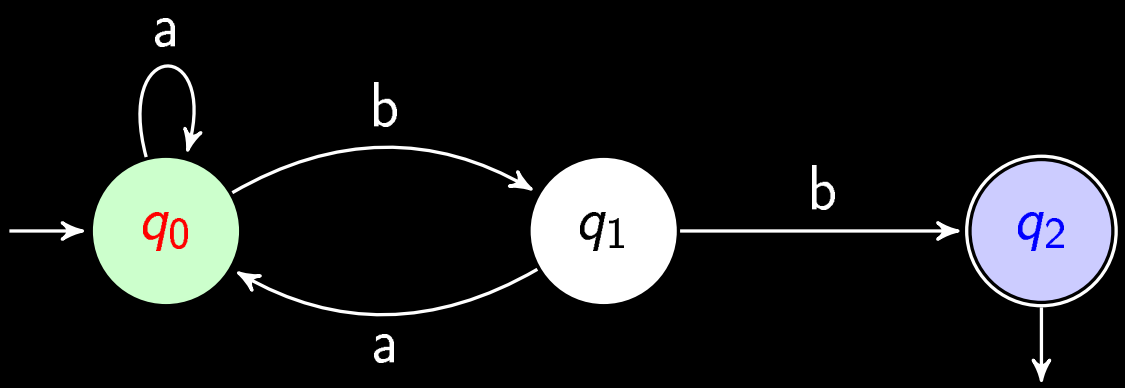
- q_0 et q_1 sont-ils distinguables par a ? Par b ?
- Par quelle chaîne q_1 et q_2 sont-ils distinguables ?

Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.



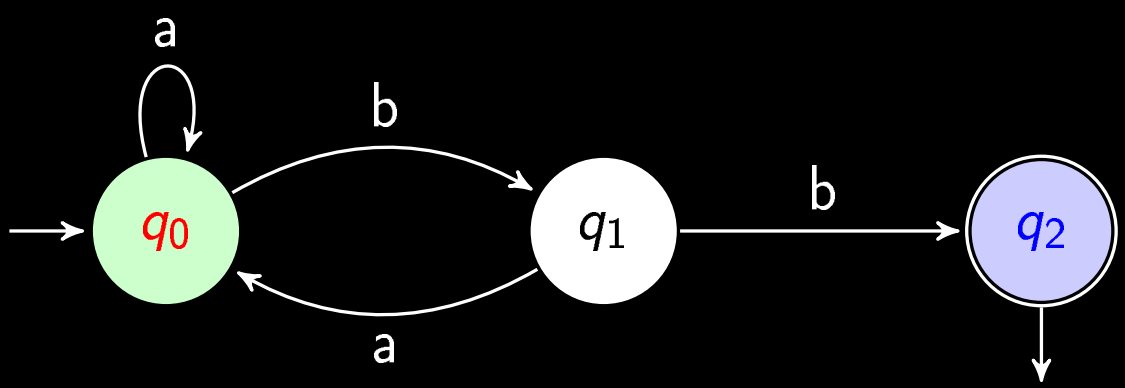
- q_0 et q_1 sont-ils distinguables par a ? Par b ?
- Par quelle chaîne q_1 et q_2 sont-ils distinguables ?

Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^\star$ si :

- $\tau(q_i, \mu) \in T$ et $\tau(q_j, \mu) \notin T$
- $\tau(q_i, \mu) \notin T$ et $\tau(q_j, \mu) \in T$

Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.



- q_0 et q_1 sont-ils distinguables par a ? Par b ?
- Par quelle chaîne q_1 et q_2 sont-ils distinguables ?

Théorème 46 (Équivalence de Nerode)

On définit la relation :

$$q_i \equiv_N q_j \Leftrightarrow q_i \text{ et } q_j \text{ sont indistinguables}$$

C'est une classe d'équivalence.

Théorème 47 (Théorème de Myhill-Nerode de l'unicité)

Soit L un langage reconnaissable. Parmi tous les automates déterministes finis qui reconnaissent L, il en existe un et un seul (au nom des états près) qui a un nombre minimum d'états. C'est l'automate minimal de L.

Théorème 46 (Équivalence de Nerode)

On définit la relation :

$$q_i \equiv_N q_j \Leftrightarrow q_i \text{ et } q_j \text{ sont indistinguables}$$

C'est une classe d'équivalence.

Théorème 47 (Théorème de Myhill-Nerode (1958))

Soit L un langage reconnaissable. Parmi tous les automates déterministes finis qui reconnaissent L, il en existe un et un seul (au nom des états près) qui a un nombre minimum d'états. C'est l'automate minimal de L.

Théorème 46 (Équivalence de Nerode)

On définit la relation :

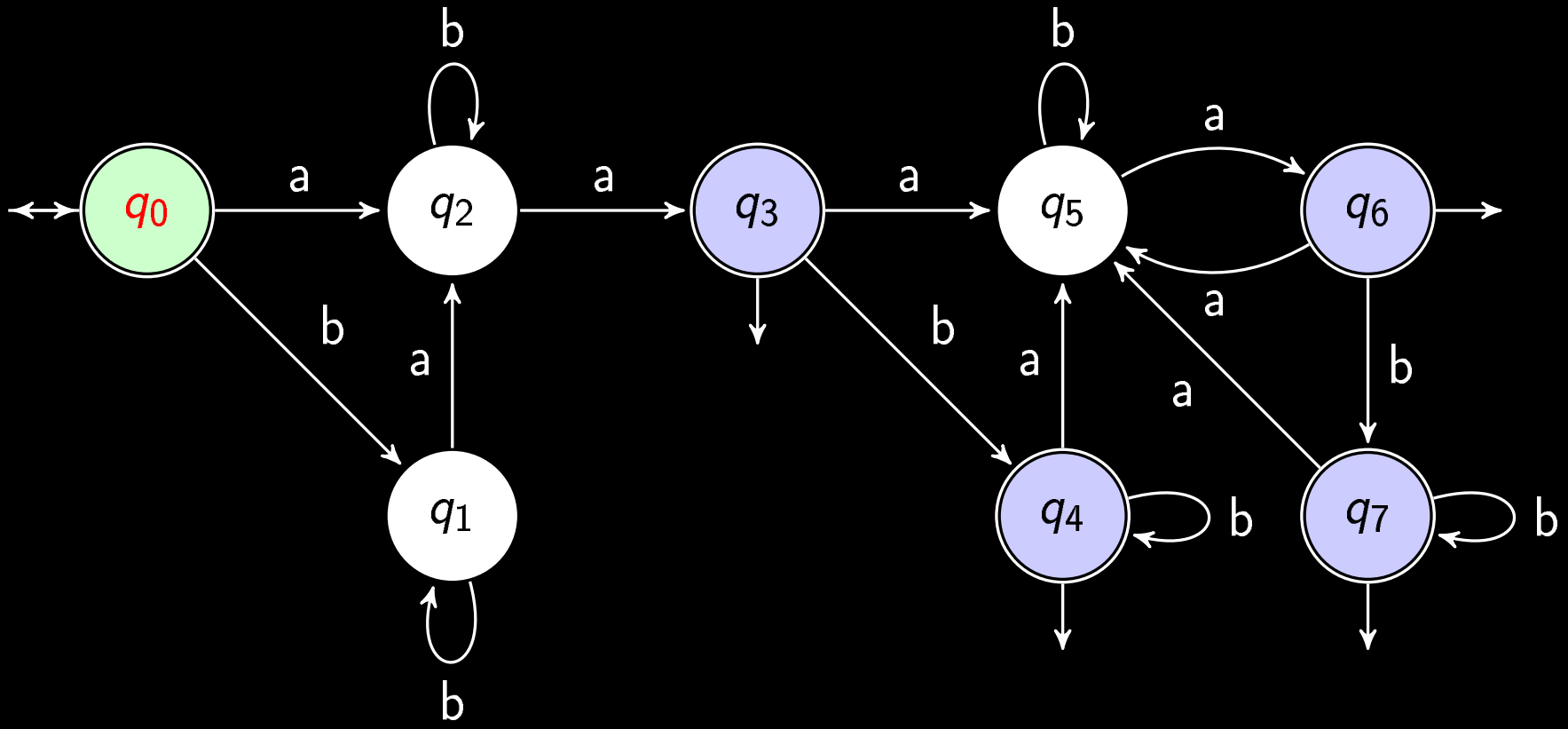
$$q_i \equiv_N q_j \Leftrightarrow q_i \text{ et } q_j \text{ sont indistinguables}$$

C'est une classe d'équivalence.

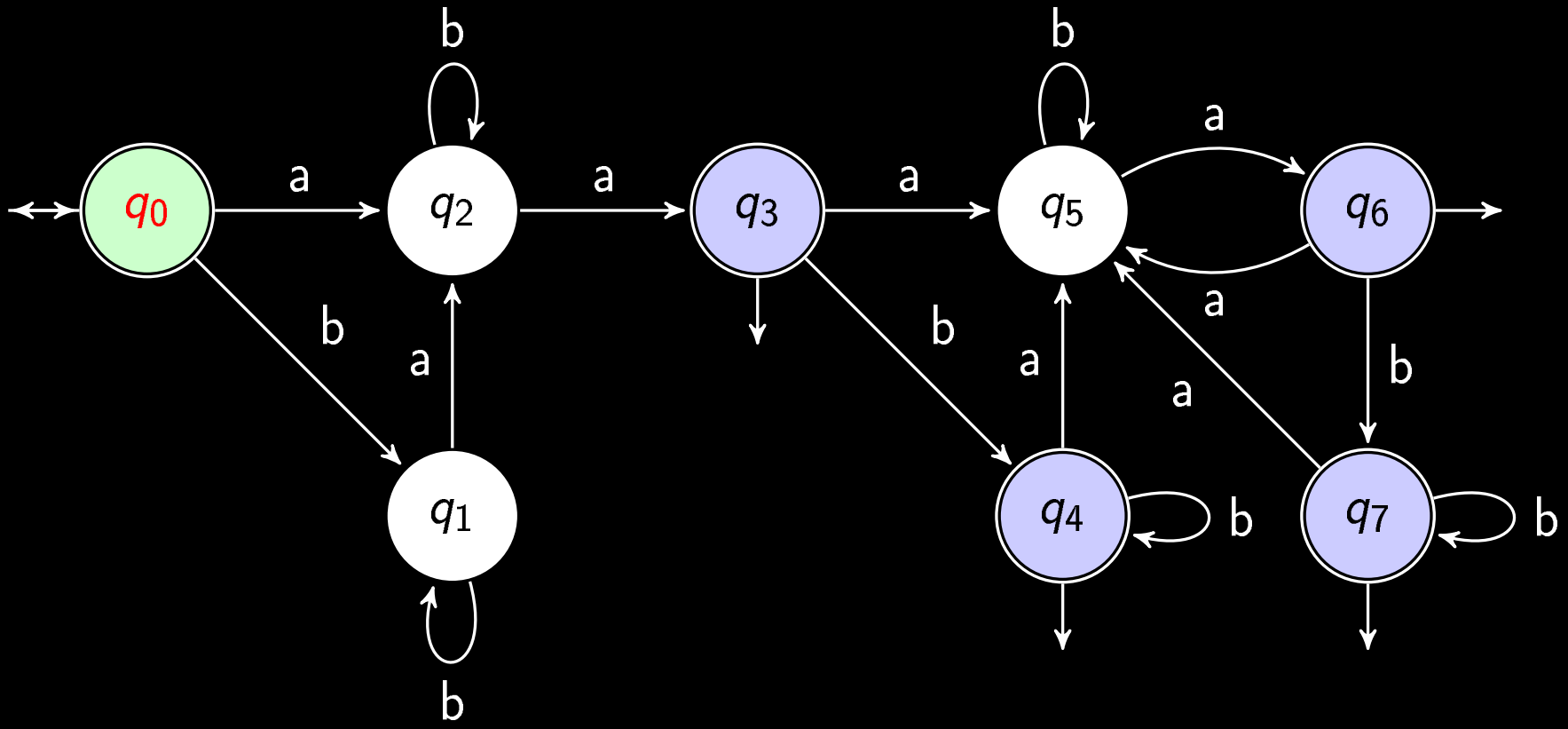
Théorème 47 (Théorème de Myhill-Nerode (1958))

Soit L un langage reconnaissable. Parmi tous les automates déterministes finis qui reconnaissent L, il en existe un et un seul (au nom des états près) qui a un nombre minimum d'états. C'est l'automate minimal de L.

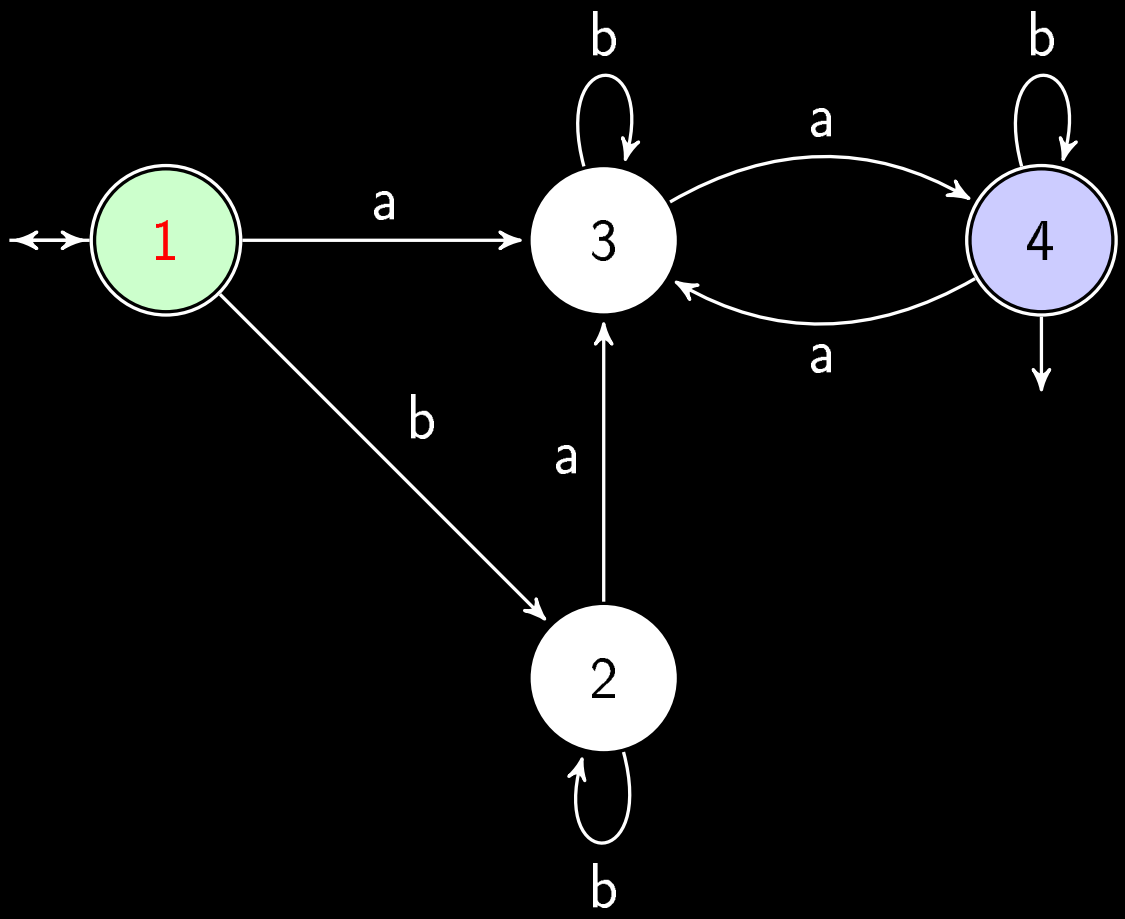
Sommaire



	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
λ	1	2	2	1	1	2	1	1



	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
λ	1	2	2	1	1	2	1	1


b
b
a
a
1
3
4
a
b
a
2
b

Sommaire

Sommaire

Mon père est marinier
Dans cette péniche
Ma mère dit : « La paix niche
Dans ce mari niais »
Ma mère est habile
Mais ma bile est amère
Car mon père et ses verres
Ont les pieds fragiles

- « le vendredi matin, l'étudiant sent la rose »
- « Tous les matins, je mange un avocat »
- « Tous les avocats, je mange la Pologne »
- « Je regarde cette jolie fille qui se promène avec des jumelles »

- « le vendredi matin, l'étudiant sent la rose »
- « Tous les matins, je mange un avocat »
- « Tous les avocats, je mange la Pologne »
- « Je regarde cette jolie fille qui se promène avec des jumelles »

- « le vendredi matin, l'étudiant sent la rose »
- « Tous les matins, je mange un avocat »
- « Tous les avocats, je mange la Pologne »
- « Je regarde cette jolie fille qui se promène avec des jumelles »

- « le vendredi matin, l'étudiant sent la rose »
- « Tous les matins, je mange un avocat »
- « Tous les avocats, je mange la Pologne »
- « Je regarde cette jolie fille qui se promène avec des jumelles »

```
>>> 3 + 1.2
4.2
```

```
# 3 + 1.2 ;;
"Error: This expression has type float but an expression was
    expected of type int"
# 3. +. 1.2 ;;
- : float = 4.2
# 3 + 1 ;;
- : int = 4
```

```
>>> 3 + 1.2
4.2
```

```
# 3 + 1.2 ;;
"Error: This expression has type float but an expression was
    expected of type int"
# 3. +. 1.2 ;;
- : float = 4.2
# 3 + 1 ;;
- : int = 4
```

Sommaire

Sommaire

Définition 48

Une grammaire algébrique est définie par la donnée de :

- Σ : un ensemble fini de *symboles terminaux* ;
- V : un ensemble fini de *variables* (symboles non terminaux) ;
- $S \in V$: une variable particulière appelée axiome (pensez à « Start ») ;
- P : un ensemble fini de *règles de production* (ou de réécriture) qui sont de la forme $A \to w$ avec $A \in V$ et $w \in (\Sigma \cup V)^*$.

Définition 48

Une grammaire algébrique est définie par la donnée de :

- Σ : un ensemble fini de *symboles terminaux* ;
- V : un ensemble fini de *variables* (symboles non terminaux) ;
- $S \in V$: une variable particulière appelée axiome (pensez à « Start ») ;
- P : un ensemble fini de *règles de production* (ou de réécriture) qui sont de la forme $A \to w$ avec $A \in V$ et $w \in (\Sigma \cup V)^*$.

Définition 48

Une grammaire algébrique est définie par la donnée de :

- Σ : un ensemble fini de *symboles terminaux* ;
- V : un ensemble fini de *variables* (symboles non terminaux) ;
- $S \in V$: une variable particulière appelée axiome (pensez à « Start ») ;
- P : un ensemble fini de *règles de production* (ou de réécriture) qui sont de la forme $A \to w$ avec $A \in V$ et $w \in (\Sigma \cup V)^*$.

Définition 48

Une grammaire algébrique est définie par la donnée de :

- Σ : un ensemble fini de *symboles terminaux* ;
- V : un ensemble fini de *variables* (symboles non terminaux) ;
- $S \in V$: une variable particulière appelée axiome (pensez à « Start ») ;
- P : un ensemble fini de *règles de production* (ou de réécriture) qui sont de la forme $A \to w$ avec $A \in V$ et $w \in (\Sigma \cup V)^*$.

Définition 48

Une grammaire algébrique est définie par la donnée de :

- Σ : un ensemble fini de *symboles terminaux* ;
- V : un ensemble fini de *variables* (symboles non terminaux) ;
- $S \in V$: une variable particulière appelée axiome (pensez à « Start ») ;
- P : un ensemble fini de *règles de production* (ou de réécriture) qui sont de la forme $A \rightarrow w$ avec $A \in V$ et $w \in (\Sigma \cup V)^\star$.

Définition 49

Une chaîne μ *dérive* d'une chaîne ν, si, à partir de ν, on peut arriver à μ par un nombre fini d'applications des règles de G.
On note $\nu \Rightarrow \mu$ ou on peut même préciser le nombre n d'applications avec $\nu \overset{n}{\Rightarrow} \mu$

Le *langage engendré* par une grammaire G est l'ensemble de toutes les chaînes terminales (appartenant donc à $\Sigma^\star$) qui dérivent de l'axiome S. On le note $\mathcal{L}(G)$.

Définition 49

Une chaîne μ *dérive* d'une chaîne ν, si, à partir de ν, on peut arriver à μ par un nombre fini d'applications des règles de G.
On note $\nu \Rightarrow \mu$ ou on peut même préciser le nombre n d'applications avec $\nu \overset{n}{\Rightarrow} \mu$

Le *langage engendré* par une grammaire G est l'ensemble de toutes les chaînes terminales (appartenant donc à $\Sigma^\star$) qui dérivent de l'axiome S. On le note $\mathcal{L}(G)$.

Définition 49

Une chaîne μ *dérive* d'une chaîne ν, si, à partir de ν, on peut arriver à μ par un nombre fini d'applications des règles de G.
On note $\nu \Rightarrow \mu$ ou on peut même préciser le nombre n d'applications avec $\nu \overset{n}{\Rightarrow} \mu$

Le *langage engendré* par une grammaire G est l'ensemble de toutes les chaînes terminales (appartenant donc à $\Sigma^\star$) qui dérivent de l'axiome S. On le note $\mathcal{L}(G)$.

Définition 50

Une *dérivation à gauche* est une dérivation qui s'applique à la première variable rencontrée dans la chaîne (à gauche, donc...)

Sommaire

- Pour toute règle de la forme $X \rightarrow ABCD...$, on construit un arbre de ne nœud X et de feuilles A, B, C, D,...
- Pour toute règle du type $X \rightarrow A|B|C|...$, on construit un des arbres de nœud X et n'ayant qu'une seule feuille, celle-ci étant A ou B ou C ou...
- On applique récursivement cette règle tant que les feuilles provisoires sont des symboles non terminaux.

- Pour toute règle de la forme $X \rightarrow ABCD...$, on construit un arbre de ne nœud X et de feuilles A, B, C, D,...
- Pour toute règle du type $X \rightarrow A|B|C|...$, on construit un des arbres de nœud X et n'ayant qu'une seule feuille, celle-ci étant A ou B ou C ou...
- On applique récursivement cette règle tant que les feuilles provisoires sont des symboles non terminaux.

- Pour toute règle de la forme $X \rightarrow ABCD...$, on construit un arbre de ne nœud X et de feuilles A, B, C, D,...
- Pour toute règle du type $X \rightarrow A|B|C|...$, on construit un des arbres de nœud X et n'ayant qu'une seule feuille, celle-ci étant A ou B ou C ou...
- On applique récursivement cette règle tant que les feuilles provisoires sont des symboles non terminaux.

Définition 51

Une grammaire est ambiguë si pour au moins une chaîne du langage, il existe au moins deux arbres de dérivation.

Sommaire

Définition 52

Une grammaire algébrique est *régulière* si ses règles sont d'un des types suivant :

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \lambda$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 52

Une grammaire algébrique est *régulière* si ses règles sont d'un des types suivant :

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \lambda$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 52

Une grammaire algébrique est *régulière* si ses règles sont d'un des types suivant :

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \lambda$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 52

Une grammaire algébrique est *régulière* si ses règles sont d'un des types suivant :

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \lambda$, avec $A, B \in V$ et $a \in \Sigma$.

Théorème 53

Si G est une grammaire régulière, alors $L(G)$ est régulier.

Tout langage régulier peut être engendré par une grammaire régulière.

Définition 52

Une grammaire algébrique est *régulière* si ses règles sont d'un des types suivant :

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \lambda$, avec $A, B \in V$ et $a \in \Sigma$.

Théorème 53

Si G est une grammaire régulière, alors $\mathcal{L}(G)$ est régulier.

Tout langage régulier peut être engendré par une grammaire régulière.

Définition 52

Une grammaire algébrique est *régulière* si ses règles sont d'un des types suivant :

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \lambda$, avec $A, B \in V$ et $a \in \Sigma$.

Théorème 53

Si G est une grammaire régulière, alors $\mathcal{L}(G)$ est régulier.
Tout langage régulier ***peut*** *être engendré par une grammaire régulière.*

Sommaire

Théorème 54

Soit L un langage régulier. Il existe une constante k telle que toute chaîne μ de L dont la longueur est strictement supérieure à k peut s'écrire sous la forme $\mu = \nu\xi\rho$ avec ξ une chaîne non vide, la longueur de $\nu\xi$ étant inférieure ou égale à k et la chaîne $\nu\xi^n\rho$ appartient à L pour tout entier n.

Sommaire

Définition 55

Le *préfixe terminal* d'une chaîne u de $(\Sigma \cup V)^\star$ est la sous-chaîne des symboles terminaux qui précèdent la première variable de u.

Sommaire

Début

Q ←[S]; c← S

TantQue $c \neq \mu$ ou $Q \neq \emptyset$ **Faire**

Défiler Q

Appliquer successivement chacune des règles possibles à la première variable de c

Pour chaque chaîne obtenue **Faire**

Si préfixe terminal de chaîne est préfixe de μ et chaîne est non terminale **Alors**

l'enfiler dans Q

SinonSi La chaîne est égale à μ **Alors**

C'est fini

FinSi

c ← la tête de Q

FinPour

FinTantQue

Retourner Ce n'est pas un mot du langage

Sommaire

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \lambda$.
- À chaque variable de la grammaire on associe un état de l'automate.
- L'axiome est l'unique état initial.
- Les règles $A \rightarrow aB$ donnent les transitions $A \xrightarrow{a} B$.
- Une règle $A \rightarrow \lambda$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \lambda$.
- À chaque variable de la grammaire on associe un état de l'automate.
- L'axiome est l'unique état initial.
- Les règles $A \rightarrow aB$ donnent les transitions $A \xrightarrow{a} B$.
- Une règle $A \rightarrow \lambda$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \to aB$ ou $A \to \lambda$.
- À chaque variable de la grammaire on associe un état de l'automate.
- L'axiome est l'unique état initial.
- Les règles $A \to aB$ donnent les transitions $A \xrightarrow{a} B$.
- Une règle $A \to \lambda$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \to aB$ ou $A \to \lambda$.
- À chaque variable de la grammaire on associe un état de l'automate.
- L'axiome est l'unique état initial.
- Les règles $A \to aB$ donnent les transitions $A \xrightarrow{a} B$.
- Une règle $A \to \lambda$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \to aB$ ou $A \to \lambda$.
- À chaque variable de la grammaire on associe un état de l'automate.
- L'axiome est l'unique état initial.
- Les règles $A \to aB$ donnent les transitions $A \xrightarrow{a} B$.
- Une règle $A \to \lambda$ donne un état terminal.

Sommaire

On appellera **proposition** tout énoncé dont on peut décider s'il est vrai ou faux indépendamment de ses composantes.

On distinguera la logique des propositions de la logique des **prédicats** qui introduit des variables dans les assertions qui peut les rendre donc parfois vraies et parfois fausses.

« *Igor dort* »

« *x dort* »

On appellera **proposition** tout énoncé dont on peut décider s'il est vrai ou faux indépendamment de ses composantes.
On distinguera la logique des propositions de la logique des **prédicats** qui introduit des variables dans les assertions qui peut les rendre donc parfois vraies et parfois fausses.

« Igor dort »

« x dort »

On appellera **proposition** tout énoncé dont on peut décider s'il est vrai ou faux indépendamment de ses composantes.
On distinguera la logique des propositions de la logique des **prédicats** qui introduit des variables dans les assertions qui peut les rendre donc parfois vraies et parfois fausses.
« *Igor dort* »
« *x dort* »

On appellera **proposition** tout énoncé dont on peut décider s'il est vrai ou faux indépendamment de ses composantes.
On distinguera la logique des propositions de la logique des **prédicats** qui introduit des variables dans les assertions qui peut les rendre donc parfois vraies et parfois fausses.
« *Igor dort* »
« *x dort* »

- **syntaxe** : n. f. (bas latin syntaxis, du grec suntaksis, ordre) Partie de la grammaire qui décrit les règles par lesquelles les unités linguistiques se combinent en phrases.
- **sémantique** : n. f. (bas latin semanticus, du grec sêmantikos, qui signifie) **1.** Étude du sens des unités linguistiques et de leurs combinaisons. **2.** Étude des propositions d'une théorie déductive du point de vue de leur vérité ou de leur fausseté.

- **syntaxe** : n. f. (bas latin syntaxis, du grec suntaksis, ordre) Partie de la grammaire qui décrit les règles par lesquelles les unités linguistiques se combinent en phrases.
- **sémantique** : n. f. (bas latin semanticus, du grec sêmantikos, qui signifie) **1.** Étude du sens des unités linguistiques et de leurs combinaisons. **2.** Étude des propositions d'une théorie déductive du point de vue de leur vérité ou de leur fausseté.

Sommaire

L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini ou infini dénombrable de *variables propositionnelles* ;
- la constante logique $\bot$ qui se lit « FAUX » ;
- les *connecteurs* $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique) et $\leftrightarrow$ (équivalent) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.

L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini ou infini dénombrable de *variables propositionnelles* ;
- la constante logique $\bot$ qui se lit « FAUX » ;
- les *connecteurs* $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique) et $\leftrightarrow$ (équivalent) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.

L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini ou infini dénombrable de *variables propositionnelles* ;
- la constante logique $\bot$ qui se lit « FAUX » ;
- les *connecteurs* $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique) et $\leftrightarrow$ (équivalent) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.

L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini ou infini dénombrable de *variables propositionnelles* ;
- la constante logique $\bot$ qui se lit « FAUX » ;
- les *connecteurs* $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique) et $\leftrightarrow$ (équivalent) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.

L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini ou infini dénombrable de *variables propositionnelles* ;
- la constante logique $\bot$ qui se lit « FAUX » ;
- les *connecteurs* $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique) et $\leftrightarrow$ (équivalent) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.

L'alphabet du langage de la logique des propositions est constitué de :

- un ensemble fini ou infini dénombrable de *variables propositionnelles* ;
- la constante logique $\bot$ qui se lit « FAUX » ;
- les *connecteurs* $\neg$ (non), $\wedge$ (et), $\vee$ (ou), $\rightarrow$ (implique) et $\leftrightarrow$ (équivalent) ;
- les parenthèses « (» et «) ».

Le connecteur $\neg$ est un *connecteur unaire* et les autres sont des *connecteurs binaires*.

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{CF}$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$

$\neg \wedge (p) \vee$

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{C}F$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$

$\neg \wedge (p) \vee$

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{C}F$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$

$\neg \wedge (p) \vee$

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{C}F$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$

$\neg \wedge (p) \vee$

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{C}F$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$
$\neg \wedge (p) \vee$

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{C}F$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$

$\neg \wedge (p) \vee$

L'*ensemble des formules* de la logique propositionnelle est le plus petit ensemble $\mathcal{C}F$ tel que :

- toute variable propositionnelle est un élément de $\mathcal{F}$;
- $\bot$ est un élément de $\mathcal{F}$;
- si $p \in \mathcal{F}$, alors $(\neg p) \in \mathcal{F}$;
- si p et q sont dans $\mathcal{F}$, alors $(p \wedge q)$, $(p \vee q)$, $(p \rightarrow q)$, et $(p \leftrightarrow q)$ sont des éléments de $\mathcal{F}$.

$s \rightarrow ((p \vee (\neg q)) \vee (p \vee (\neg r)))$
$\neg \wedge (p) \vee$

- $\neg$ est prioritaire sur les autres opérateurs ;
- $\vee$ et $\wedge$ sont prioritaires sur $\rightarrow$ et $\leftrightarrow$.

- $\neg$ est prioritaire sur les autres opérateurs ;
- $\vee$ et $\wedge$ sont prioritaires sur $\rightarrow$ et $\leftrightarrow$.

Sommaire

Théorème 56

Si une propriété P portant sur les formules de $\mathcal{F}$ est telle que :

- *toute variable propositionnelle vérifie P ;*
- *$\bot$ vérifie P ;*
- *si la formule p vérifie P, alors $(\neg p)$ vérifie P ;*
- *si p et q vérifient P, alors $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ et $(p \leftrightarrow q)$ vérifient P ;*

alors toutes les formules de $\mathcal{F}$ vérifient P.

Théorème 56

Si une propriété P portant sur les formules de $\mathcal{F}$ est telle que :

- *toute variable propositionnelle vérifie P ;*
- *$\bot$ vérifie P ;*
- *si la formule p vérifie P, alors $(\neg p)$ vérifie P ;*
- *si p et q vérifient P, alors $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ et $(p \leftrightarrow q)$ vérifient P ;*

alors toutes les formules de $\mathcal{F}$ vérifient P.

Théorème 56

Si une propriété P portant sur les formules de $\mathcal{F}$ est telle que :

- *toute variable propositionnelle vérifie P ;*
- *$\bot$ vérifie P ;*
- *si la formule p vérifie P, alors $(\neg p)$ vérifie P ;*
- *si p et q vérifient P, alors $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ et $(p \leftrightarrow q)$ vérifient P ;*

alors toutes les formules de $\mathcal{F}$ vérifient P.

Théorème 56

Si une propriété P portant sur les formules de $\mathcal{F}$ est telle que :

- *toute variable propositionnelle vérifie P ;*
- *$\bot$ vérifie P ;*
- *si la formule p vérifie P, alors $(\neg p)$ vérifie P ;*
- *si p et q vérifient P, alors $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ et $(p \leftrightarrow q)$ vérifient P ;*

alors toutes les formules de $\mathcal{F}$ vérifient P.

Théorème 56

Si une propriété P portant sur les formules de $\mathcal{F}$ est telle que :

- *toute variable propositionnelle vérifie P ;*
- *$\bot$ vérifie P ;*
- *si la formule p vérifie P, alors $(\neg p)$ vérifie P ;*
- *si p et q vérifient P, alors $(p \vee q)$, $(p \wedge q)$, $(p \rightarrow q)$ et $(p \leftrightarrow q)$ vérifient P ;*

alors toutes les formules de $\mathcal{F}$ vérifient P.

Sous-formules

Il s'agit de toutes les formules apparaissant dans une formule donnée.
Par exemple, l'ensemble des sous-formules de $(p \to q) \vee \neg(q \leftrightarrow r)$ est

$$\{p \to q, \neg(q \leftrightarrow r), q \leftrightarrow r, p, q, r\}$$

Sous-formules

Il s'agit de toutes les formules apparaissant dans une formule donnée.
Par exemple, l'ensemble des sous-formules de $(p \to q) \vee \neg(q \leftrightarrow r)$ est

$$\{p \to q, \neg(q \leftrightarrow r), q \leftrightarrow r, p, q, r\}$$

Sous-formules

Il s'agit de toutes les formules apparaissant dans une formule donnée.
Par exemple, l'ensemble des sous-formules de $(p \to q) \vee \neg(q \leftrightarrow r)$ est

$$\{p \to q, \neg(q \leftrightarrow r), q \leftrightarrow r, p, q, r\}$$

Sommaire

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$.

avec p et q des éléments quelconques de $\mathcal{F}$.

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$.

avec p et q des éléments quelconques de $\mathcal{F}$.

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$.

avec p et q des éléments quelconques de $\mathcal{F}$.

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$,

avec p et q des éléments quelconques de $\mathcal{F}$.

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$,

avec p et q des éléments quelconques de $\mathcal{F}$.

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$,

avec p et q des éléments quelconques de $\mathcal{F}$.

Valeurs et tables de vérité

On considère un ensemble $\mathcal{B} = \{0,1\}$ ou $\{F,V\}$ ou $\{\text{Oui},\text{Non}\}$, etc.

Définition 57

Une *distribution des valeurs de vérité* est une application v de $\mathcal{F}$ dans $\mathcal{B}$ définie inductivement par :

- $v(\bot) = 0$,
- $v(\neg p) = 1$ si, et seulement si, $v(p) = 0$;
- $v(p \wedge q) = 1$ si, et seulement si, $(p) = v(q) = 1$;
- $v(p \vee q) = 0$ si, et seulement si, $v(p) = v(q) = 0$;
- $v(p \rightarrow q) = 0$ si, et seulement si, $v(p) = 1$ et $v(q) = 0$;
- $v(p \leftrightarrow) = 1$ si, et seulement si, $v(p) = v(q)$,

avec p et q des éléments quelconques de $\mathcal{F}$.