

Machines de Turing, langages, automates, grammaires

INFO1 - Semaines 43 à 2

Guillaume CONNAN

IUT de Nantes - Dpt d'informatique

Dernière mise à jour : 3 décembre 2012

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière



Kurt Gödel (1906-1978)



Alonzo Church (1903-1995)



Stephen Kleene (1909-1994)



Alan Turing (1912-1954)



Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

Sommaire

1 Approche historique

2 Les machines de Turing

● Présentation sommaire

- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

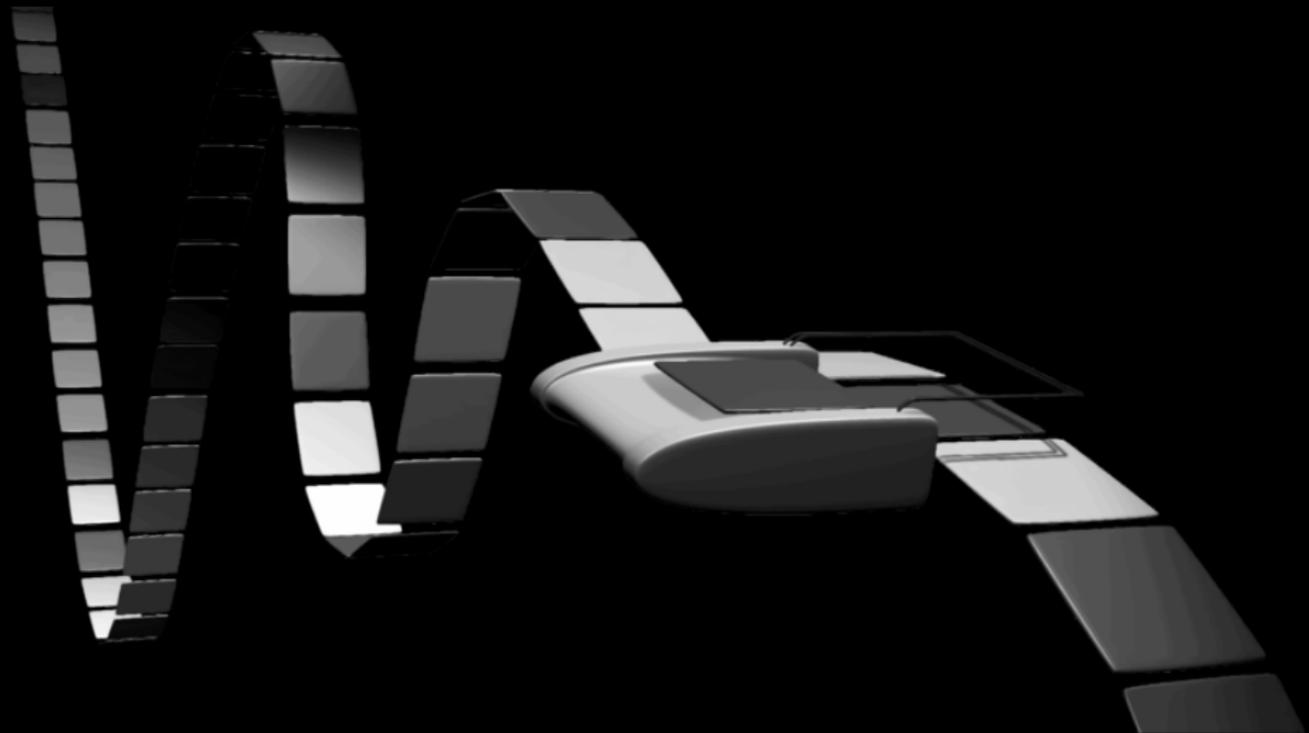
● Automates standards

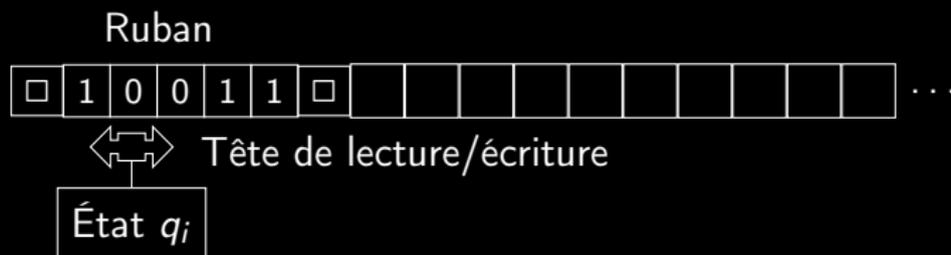
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile

5 Grammaires

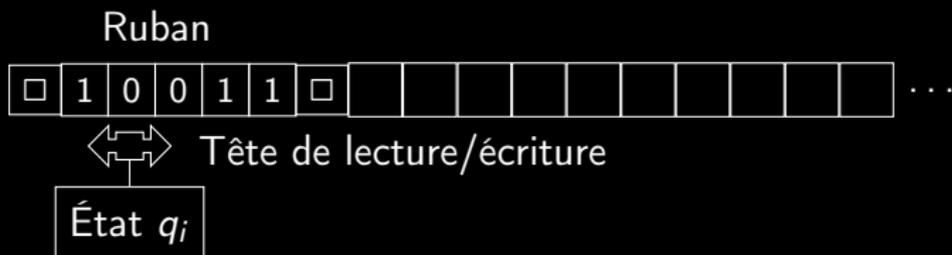
- Syntaxe et sémantique
- Grammaires algébriques (ou hors contexte)
- Grammaire régulière
- Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière



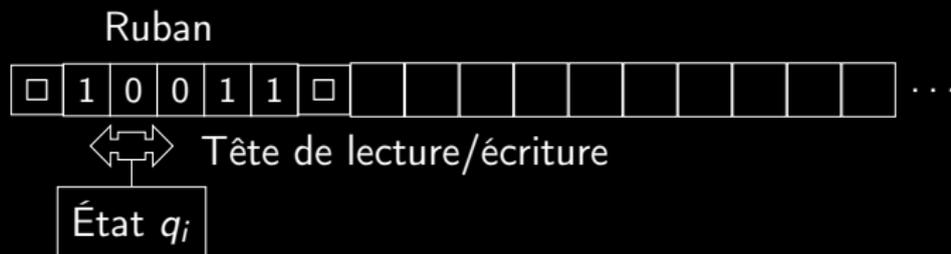


L'ensemble des symboles $\{0, 1, \square\}$ forme un alphabet.



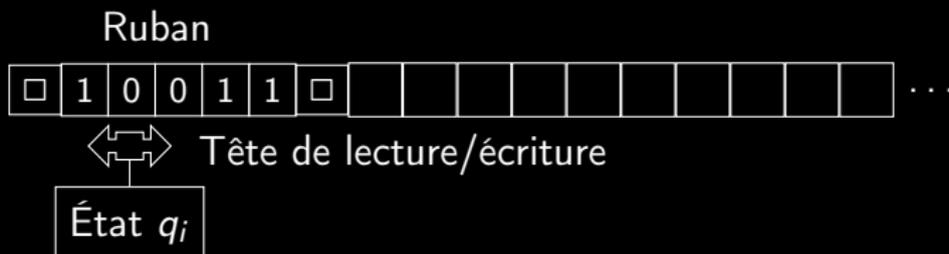
L'ensemble des symboles $\{0, 1, \square\}$ forme un **alphabet**.

- changer l'état du pointeur



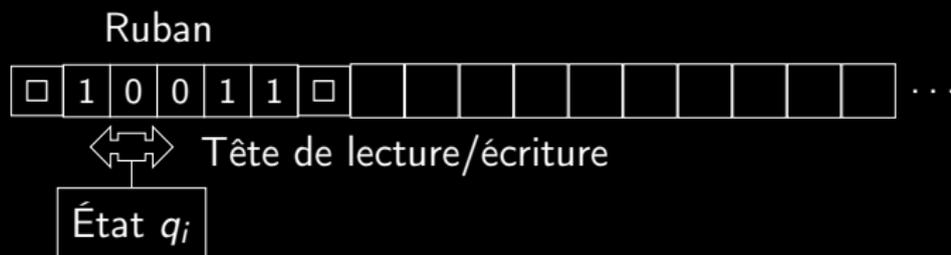
L'ensemble des symboles $\{0, 1, \square\}$ forme un **alphabet**.

- 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 $\{0, 1, \square\}$ 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 $\{0, 1, \square\}$ 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

1 Approche historique

2 Les machines de Turing

- Présentation sommaire

- **Premier exemple**

- Définitions
- Fonctions MT-calculables
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

- Automates standards

- Automates déterministes

- Automates émondés

- Opérations rationnelles sur les automates

- Théorème de Kleene

- Lemme de la pompe

- Automate associé à un langage défini par une expression rationnelle

- Automate minimal

- Automates à états finis avec sortie

- Automates à pile

5 Grammaires

- Syntaxe et sémantique

- Grammaires algébriques (ou hors contexte)

- Grammaire régulière

- Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

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

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

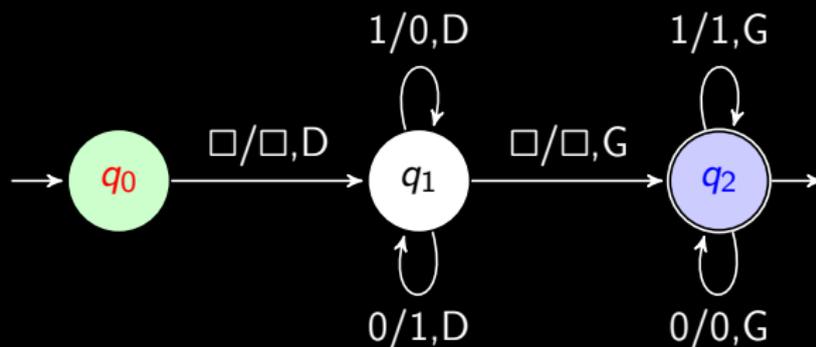
On entre un « mot » écrit à partir de l'alphabet $\{0,1,\square\}$.
Nous voudrions changer les 0 en 1 et vice-versa puis remettre le pointeur dans sa position initiale.

- 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.

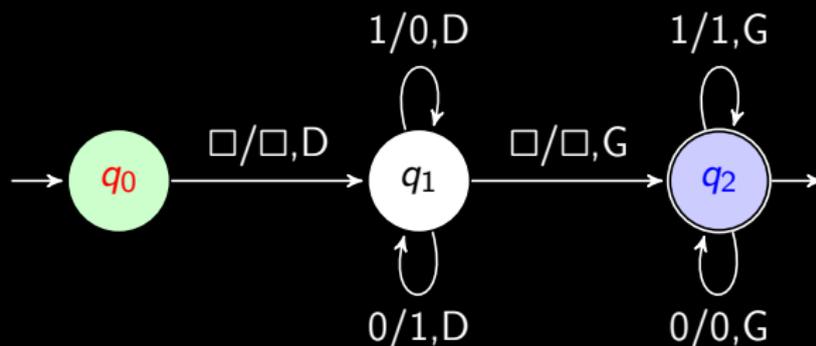
automate



Exemple :

Testez cette machine sur la chaîne 1100110

automate



Exemple

Testez cette machine sur la chaîne $\square 10011 \square$.

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- **Définitions**
- Fonctions MT-calculables
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

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 :

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 ;
- $\emptyset \subseteq Q$ est l'ensemble des états finaux ;
- $\emptyset \subseteq X$ est l'ensemble des symboles de fin de ruban.

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* ;

• Σ est l'*alphabet de l'entrée*.

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* ;
- $\square \in X$ 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 :

- $q_0 \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\square \in X$ 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 :

- $q_0 \in Q$ est l'*état initial* ;
- $Q_f \subseteq Q$ est l'*ensemble des états finaux* ;
- $\square \in X$ 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 :

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

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

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

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

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

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

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

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

$\varphi(q_i, x)$	q_0	q_1	q_2
\square	(q_1, \square, D)	(q_2, \square, 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 $\mathbb{1}$ le mot ne comportant aucun caractère.

L'ensemble X^* des mots construits avec l'alphabet X est défini par :

- 1 $\mathbb{1} \in X^*$;
- 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 $\mathbb{1}$ par application de l'étape 2 un nombre fini de fois.

Définition 2

Soit X un alphabet. Soit $\mathbb{1}$ le mot ne comportant aucun caractère.
L'ensemble X^* des mots construits avec l'alphabet X est défini par :

- 1 $\mathbb{1} \in X^*$;
- 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 $\mathbb{1}$ par application de l'étape 2 un nombre fini de fois.

Définition 2

Soit X un alphabet. Soit $\mathbb{1}$ le mot ne comportant aucun caractère.

L'ensemble X^* des mots construits avec l'alphabet X est défini par :

- 1 $\mathbb{1} \in X^*$;
- 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 $\mathbb{1}$ par application de l'étape 2 un nombre fini de fois.

Définition 2

Soit X un alphabet. Soit $\mathbb{1}$ le mot ne comportant aucun caractère.

L'ensemble X^* des mots construits avec l'alphabet X est défini par :

- 1 $\mathbb{1} \in X^*$;
- 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 $\mathbb{1}$ 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 $m_1m_2 \in X^*$ obtenu en écrivant m_2 à la suite de m_1 .

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- **Fonctions MT-calculables**
- La théorie en pratique
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

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 \bar{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 \bar{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 \bar{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 \bar{n} la représentation de l'entier n .

Décrivons une machine de Turing qui calcule cette fonction :

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

Décrivons une machine de Turing qui calcule cette fonction :

- la fonction successeur s est définie sur $D = \{1, \square 1 \square, \square 1 1 \square, \square 1 1 1 \square, \dots\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole \square , toutes les autres cases étant occupées par des \square ;
- l'alphabet de ruban est $X = \{1, \square\}$;
- 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 :

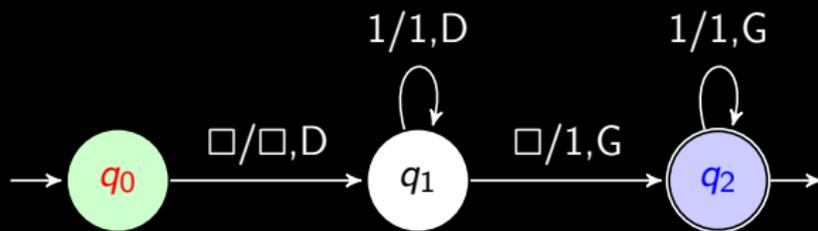
- la fonction successeur s est définie sur $D = \{1, \square 1 \square, \square 11 \square, \square 111 \square, \dots\}$: l'entrée sur le ruban est donc la représentation unaire du nombre n précédée du symbole \square , toutes les autres cases étant occupées par des \square ;
- l'alphabet de ruban est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

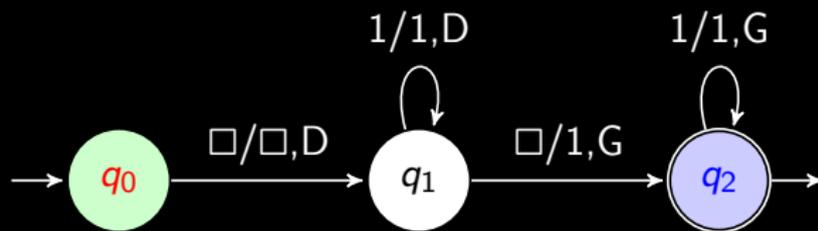
Décrivons une machine de Turing qui calcule cette fonction :

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



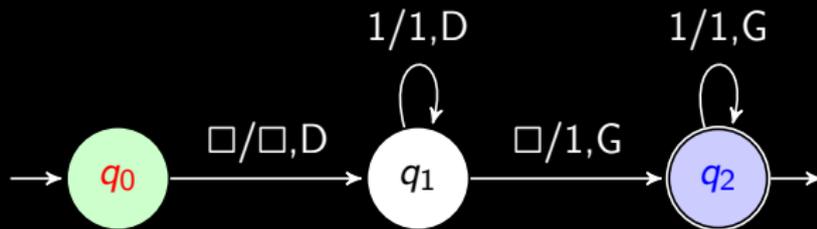
La configuration initiale est le mot $q_0 \square \bar{n} \square$.

La configuration finale est $q_2 \square \bar{n} 1 \square$.



La configuration initiale est le mot $q_0 \square \bar{n} \square$.

La configuration finale est $q_2 \square \bar{n} 1 \square$.



La configuration initiale est le mot $q_0 \square \bar{n} \square$.

La configuration finale est $q_2 \square \bar{n} 1 \square$.

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 $\varphi(q_0, \square) = (q_i, \square, D)$ avec $q_i \neq q_0$ (il faut quitter l'état initial à partir de \square) ;
- 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 $\varphi(q_f, \square)$ 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 $x \in D$, notons $y = f(x)$. Le calcul effectué par la machine (à partir de la configuration initiale) doit s'arrêter après un nombre fini d'étapes dans la configuration finale q, \square, y (la machine doit calculer $f(x)$) ;
- le calcul ne s'arrête jamais si $x \notin D$ (la machine ne calcule rien que son 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 $\varphi(q_0, \square) = (q_i, \square, D)$ avec $q_i \neq q_0$ (il faut quitter l'état initial à partir de \square) ;
- 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 $\varphi(q_f, \square)$ 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 $q_0 \square \mu \square$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \square \nu \square$ (la machine doit calculer $f(\mu)$) ;
- le calcul ne s'arrête jamais si $\mu \notin D$ (la machine ne calcule rien qui ne soit dans D) ;

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 $\varphi(q_0, \square) = (q_i, \square, D)$ avec $q_i \neq q_0$ (il faut quitter l'état initial à partir de \square) ;
- 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 $\varphi(q_f, \square)$ 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 $v = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \square \mu \square$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \square v \square$ (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 $\varphi(q_0, \square) = (q_i, \square, D)$ avec $q_i \neq q_0$ (il faut quitter l'état initial à partir de \square) ;
- 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 $\varphi(q_f, \square)$ 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 $v = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \square \mu \square$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \square v \square$ (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 $\varphi(q_0, \square) = (q_i, \square, D)$ avec $q_i \neq q_0$ (il faut quitter l'état initial à partir de \square) ;
- 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 $\varphi(q_f, \square)$ 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 $v = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \square \mu \square$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \square v \square$ (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 $\varphi(q_0, \square) = (q_i, \square, D)$ avec $q_i \neq q_0$ (il faut quitter l'état initial à partir de \square) ;
- 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 $\varphi(q_f, \square)$ 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 $v = f(\mu)$. Le calcul effectué par la machine M en partant de la configuration initiale $q_0 \square \mu \square$ s'arrête après un nombre fini d'étapes dans la configuration finale $q_f \square v \square$ (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 :

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

Nous définirons en TD :

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

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

Nous définirons en TD :

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

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

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- **La théorie en pratique**
- Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

```
let en_place vec pos valeur =  
  vec.(pos) <- valeur;  
  vec;  
;;
```

```
exception Pas_lu;;
let rec phi q pos vec =
  match (q,vec.(pos)) with
  |("q0","b") -> phi "q1" (pos+1) vec
  |("q0", _) -> raise Pas_lu
  |("q1","b") -> phi "q2" (pos-1) (en_place vec pos "b")
  |("q1","0") -> phi "q1" (pos+1) (en_place vec pos "1")
  |("q1","1") -> phi "q1" (pos+1) (en_place vec pos "0")
  |("q2","b") -> (en_place vec pos (vec.(pos)^^)) (* sortie *)
  |("q2","1") -> phi "q2" (pos-1) (en_place vec pos "1")
  |("q2","0") -> phi "q2" (pos-1) (en_place vec pos "0")
  |_ -> raise Pas_lu;;

let turing phi vec = phi "q0" 0 vec;;
```

```
# turing phi [|"b";"1";"0";"0";"1";"1";"b"|];;
- : string array = [|"b^"; "0"; "1"; "1"; "0"; "0"; "b"|]
```

```
exception Pas_lu;;  
let rec phi q pos vec =  
  match (q,vec.(pos)) with  
  |("q0","b") -> phi "q1" (pos+1) vec  
  |("q0", _) -> raise Pas_lu  
  |("q1","b") -> phi "q2" (pos-1) (en_place vec pos "b")  
  |("q1","0") -> phi "q1" (pos+1) (en_place vec pos "1")  
  |("q1","1") -> phi "q1" (pos+1) (en_place vec pos "0")  
  |("q2","b") -> (en_place vec pos (vec.(pos)^^)) (* sortie *)  
  |("q2","1") -> phi "q2" (pos-1) (en_place vec pos "1")  
  |("q2","0") -> phi "q2" (pos-1) (en_place vec pos "0")  
  |_ -> raise Pas_lu;;  
  
let turing phi vec = phi "q0" 0 vec;;
```

```
# turing phi [|"b";"1";"0";"0";"1";"1";"b"|];;  
- : string array = [|"b^"; "0"; "1"; "1"; "0"; "0"; "b"|]
```

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique

● Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

Sommaire

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique

● Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

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, \dots, g_k k fonctions arithmétiques de k variables et h une fonction arithmétique de k variables. Soit f la fonction définie par :

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

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

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

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 :

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

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 :

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

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 :

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

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 $h = s \circ \pi_3^{(3)}$:

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

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 $h = s \circ \pi_3^{(3)}$:

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

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 $h = s \circ \pi_3^{(3)}$:

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

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

1 Approche historique

2 Les machines de Turing

- Présentation sommaire
- Premier exemple
- Définitions
- Fonctions MT-calculables
- La théorie en pratique

● Fonctions récursives

3 Langages

- Un exemple pour découvrir
- Définitions et notations
- Langages
- Langages et expressions rationnels (ou réguliers)

4 Automates

- Définitions
- Langage reconnaissable
- Langage reconnu
- Automates équivalents

● Automates standards

● Automates déterministes

● Automates émondés

● Opérations rationnelles sur les automates

● Théorème de Kleene

● Lemme de la pompe

● Automate associé à un langage défini par une expression rationnelle

● Automate minimal

● Automates à états finis avec sortie

● Automates à pile

5 Grammaires

● Syntaxe et sémantique

● Grammaires algébriques (ou hors contexte)

● Grammaire régulière

● Analyse syntaxique (« parsing »)

6 Correspondance automate fini / grammaire régulière

- 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 prouve 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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

- 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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 **Langages**
 - **Un exemple pour découvrir**
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

LA JEUNE FILLE ÉTUDIE LE BEAU COURS.

L'*alphabet des symboles* est :

$$A = \{LA, JEUNE, FILLE, \acute{E}TUDIE, LE, BEAU, COURS, .\}$$

- 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 > → ETUDE
- 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 > → ETUDE
- 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 > → JEUNE
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILS
- 17 < verbe > → ETUDIE
- 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 > → JEUNE
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILS
- 17 < verbe > → ETUDE
- 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 > → BELLE
- 13 < adjectif masculin > → BELLE
- 14 < adjectif masculin > → BELLE
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILLE
- 17 < verbe > → ETUDE
- 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 > → BELLE
- 13 < adjectif masculin > → BEL
- 14 < adjectif masculin > → BEAU
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILS
- 17 < groupe > → PHRASE
- 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 > → JEUNE
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILS
- 17 < article > → LA
- 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 > → BELLE
- 13 < adjectif masculin > → BELLE
- 14 < adjectif masculin > → BELLE
- 15 < nom féminin > → LA BELLE
- 16 < nom masculin > → LA BELLE
- 17 < nom masculin > → LA BELLE
- 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 > → JEUNE
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILLE
- 17 < nom masculin > → FILLE
- 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 > → JEUNE
- 15 < nom féminin > → FILLE
- 16 < nom masculin > → FILS
- 17 < nom féminin > → FILLE
- 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 > → GARÇON
- 17 < nom masculin > → GARÇON
- 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 > → FILLE
- 17 < nom féminin > → FILLE
- 18 < nom masculin >

- 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 < nom masculin > → PAYSAN
- 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 > → ETUDIE
- 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.



Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 **Langages**
 - Un exemple pour découvrir
 - **Définitions et notations**
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

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 $\mathbb{1}_A$ la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

- 1 $\mathbb{1}_A \in A^*$;
- 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 $\mathbb{1}_A$ par application de l'étape 2 un nombre fini de fois.

Définition 13

Soit A un alphabet. Soit $\mathbb{1}_A$ la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

- 1 $\mathbb{1}_A \in A^*$;
- 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 $\mathbb{1}_A$ par application de l'étape 2 un nombre fini de fois.

Définition 13

Soit A un alphabet. Soit $\mathbb{1}_A$ la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

- 1 $\mathbb{1}_A \in A^*$;
- 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 $\mathbb{1}_A$ par application de l'étape 2 un nombre fini de fois.

Définition 13

Soit A un alphabet. Soit $\mathbb{1}_A$ la *chaîne vide* (ne comportant aucun symbole). L'ensemble A^* des chaînes construites avec l'alphabet A est défini par :

- 1 $\mathbb{1}_A \in A^*$;
- 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 $\mathbb{1}_A$ 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$ est l'ensemble de toutes les chaînes (éventuellement vides).
- $A^+ = \bigcup_{n \geq 1} A^n$ 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$ est l'ensemble de toutes les chaînes (éventuellement vides).
- $A^+ = \bigcup_{n \geq 1} A^n$ 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$ est l'ensemble de toutes les chaînes (éventuellement vides).
- $A^+ = \bigcup_{n \geq 1} A^n$ 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$: c'est l'ensemble de toutes les chaînes, éventuellement vides.
- $A^+ = \bigcup_{n \geq 1} A^n$: 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$: 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$: 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$: 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$: 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 $|\mathbb{1}_A| = 0$.
- 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, $A^0 = \{\mathbb{1}_A\}$ qui n'est donc pas vide.
- $A^* = \bigcup_{n \geq 0} A^n$: 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

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

Définition 15

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

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

On définit de même le quotient à droite :

Définition 15

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

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

On définit de même le quotient à droite.

Définition 15

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

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

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 $\xi \neq \mathbb{1}$, 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 $\xi \neq \mathbb{1}$, 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 $(\xi, \xi') \neq (\mathbb{1}, \mathbb{1})$, 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 :

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

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 $\xi \neq \mathbb{1}$, 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 $\xi \neq \mathbb{1}$, 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 $(\xi, \xi') \neq (\mathbb{1}, \mathbb{1})$, 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 :

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

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 fautive, 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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 **Langages**
 - Un exemple pour découvrir
 - Définitions et notations
 - **Langages**
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

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 $\mathcal{C}_{A^*} L = \bar{L}$ 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 LL' 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 :

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

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 :

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

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 :

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

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

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

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

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

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

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

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^+ :

$$L^+ = \bigcup_{k \in \mathbb{N}^*} L^k \text{ et } L^* = L^+ \cup \{\mathbf{1}\}$$

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 **Langages**
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - **Langages et expressions rationnels (ou réguliers)**
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 **Langages**
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - **Langages et expressions rationnels (ou réguliers)**
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

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 $\{\epsilon\}$,
- des langages du type $\{a, a^2, a^3, \dots\}$

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 $a \in A$.

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 $a \in A \cup \{\Lambda\}$.

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 $a \in A \cup \{\mathbb{1}_A\}$.

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 $a \in A \cup \{\mathbb{1}_A\}$.

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 $a \in A \cup \{\mathbb{1}_A\}$.

Théorème 25

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

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_1L_2 sont rationnels.*

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_1L_2 sont rationnels.*

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_1L_2 sont rationnels.*

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 **Langages**
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - **Langages et expressions rationnels (ou réguliers)**
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 26

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

- \emptyset ,
- $\mathbb{1}_A$,
- 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 $(R_1 R_2)$, $R_1 R_2$ et R_1^* le sont aussi.

Définition 26

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

- \emptyset ,
- $\mathbb{1}_A$,
- 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 ,
- $\mathbb{1}_A$,
- 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 ,
- $\mathbb{1}_A$,
- 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 ,
- $\mathbb{1}_A$,
- 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.

a

à/L'D'Q'Qj

a/| |--

ab

abaca/S*()

abacule/S*()

abaissable/S*()

abaissante/F*()

abaissée/F*()

abaissée/S*()

abaisse-langue/L'D'Q'

abaissement/S*()

abaisser/a4a+()

abaisseur/S*()

abajoue/S*()

abalober

abalone/S*()

```
$ egrep a[^/]*e[^/]*i[^/]*o[^/]*u ./Dictionnaire.txt
```

```
garde-chiourme  
gardes-chiourme  
quatre-vingt-douze
```

```
$ egrep a[^/]*e[^/]*i[^/]*o[^/]*u ./Dictionnaire.txt
```

```
garde-chiourme  
gardes-chiourme  
quatre-vingt-douze
```

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E_1 E_2) = \mathcal{L}(E_1) \mathcal{L}(E_2) = \mathcal{L}(E_2) \mathcal{L}(E_1)$
- $\mathcal{L}(E_1 + E_2) = \mathcal{L}(E_1) \cup \mathcal{L}(E_2) = \mathcal{L}(E_2) \cup \mathcal{L}(E_1)$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E|E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) \setminus \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E|E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) | \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E|E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E|E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) | \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E|E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) | \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 27

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

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(1_A) = \{1_A\}$
- $a \in A, \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E|E') = \mathcal{L}(E) \cup \mathcal{L}(E') = \mathcal{L}(E) | \mathcal{L}(E')$
- $\mathcal{L}(E \cdot E') = \mathcal{L}(E) \cdot \mathcal{L}(E') = \mathcal{L}(E) \mathcal{L}(E')$
- $\mathcal{L}(E^*) = \mathcal{L}(E)^*$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R|(ST) = (RS)|T$
- $R|1 = R = R$
- $R|\emptyset = \emptyset = \emptyset$
- $R|(S|T) = RS|T = (S|T)R = SR|T$
- $R^* = R^*R^* = (R^*)^* = (1|R)^*$
- $(RS)^* = (R^*S^*)^* = (R^*S^*)^*R^*S^* = (R^*S^*)^*$
- $R^* = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R| = |R = R$
- $R|\emptyset = \emptyset$
- $R|(S|T) = RS|T = (S|T)|R = S|TR$
- $R^* = R^+R^0 = (R^+)|$
- $(R|S)^* = (R^*|S^*) = (R^*|S^*)^+R^0|S^0 = (R^*|S^*)^*$
- $R^* = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R1 = 1R = R$
- $R\epsilon = \epsilon R = \emptyset$
- $R|(S|T) = RS|RT, (S|T)R = SR|TR$
- $R^* = R^+R = (R)^+(R)$
- $(RS)^* = (R^*S)^* = (RS^*)^* = (R^*S^*)^*$
- $R^* = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR$
- $R^* = R^*R^* = R^*$
- $(R|S)^* = R^*|S^* = (R^*|S^*)^* = (R^*|S^*)^*$
- $R^* = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR$
- $R^* = R^*R^* = (R^*)^* = \mathbb{1}R^*$
- $(R|S)^* = R^*|S^* = (R^*|S^*)^* = (R^*|S^*)^*R^*|S^*$
- $R^* = R^*R^* = R^*R^*$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR.$
- $R^* = R^*R^* = (R^*)^* = (\mathbb{1}|R)^*$
- $(R|S)^* = (R^*|S^*) = (R^*S^*)^* = (R^*S^*)^*R^*$
- $R^* = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR.$
- $R^* = R^*R^* = (R^*)^* = (\mathbb{1}|R)^*$
- $(R|S)^* = (R^*S^*)^* = (R^*|S^*)^* = (R^*S)^*R^*$
- $R^* = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR.$
- $R^* = R^*R^* = (R^*)^* = (\mathbb{1}|R)^*$
- $(R|S)^* = (R^*S^*)^* = (R^*|S^*)^* = (R^*S)^*R^*$
- $R^+ = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR.$
- $R^* = R^*R^* = (R^*)^* = (\mathbb{1}|R)^*$
- $(R|S)^* = (R^*S^*)^* = (R^*|S^*)^* = (R^*S)^* R^*$
- $R^+ = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR.$
- $R^* = R^*R^* = (R^*)^* = (\mathbb{1}|R)^*$
- $(R|S)^* = (R^*S^*)^* = (R^*|S^*)^* = (R^*S)^* R^*$
- $R^+ = RR^* = R^*R$

Théorème 28

- $R|S = S|R, R|R = R$
- $R|\emptyset = R$
- $(R|S)|T = R|(S|T)$
- $R(ST) = (RS)T$
- $R\mathbb{1} = \mathbb{1}R = R$
- $R\emptyset = \emptyset R = \emptyset$
- $R(S|T) = RS|RT, (S|T)R = SR|TR.$
- $R^* = R^*R^* = (R^*)^* = (\mathbb{1}|R)^*$
- $(R|S)^* = (R^*S^*)^* = (R^*|S^*)^* = (R^*S)^* R^*$
- $R^+ = RR^* = R^*R$

Théorème 29

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
 - Automate minimal
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 30

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

- 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 φ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\epsilon\})$ vers Q appelée *relation de transition*.

Définition 30

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

- 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 φ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\epsilon\})$ vers Q appelée *relation de transition*.

Définition 30

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

- 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 $Q \times (A \cup \{\epsilon\})$ vers Q , appelée *relation de transition*.

Définition 30

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

- 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 $Q \times (A \cup \{\epsilon\})$ vers Q , appelée *relation de transition*.

Définition 30

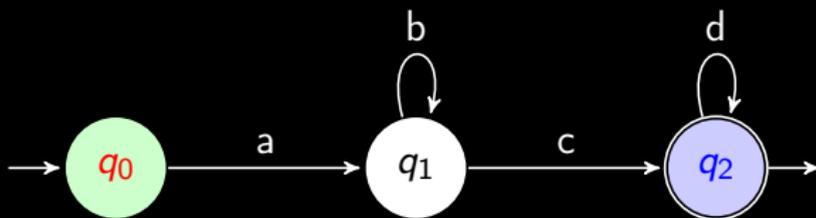
Un automate \mathcal{A} est caractérisé par un quintuplet $\langle A, Q, I, T, \varphi \rangle$:

- 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 $Q \times (A \cup \{\epsilon\})$ vers Q , appelée *relation de transition*.

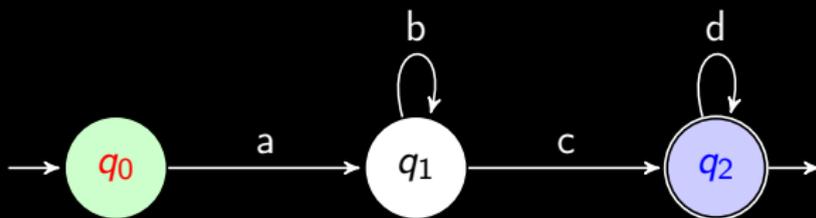
Définition 30

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

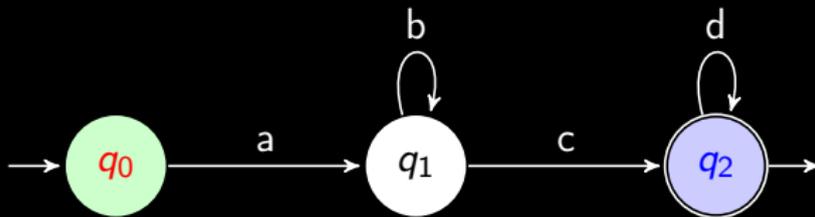
- un alphabet A appelé l'*alphabet d'entrée* ;
- un alphabet Q appelé l'*alphabet d'état* ;
- une partie I non vide de Q appelée *ensemble des états initiaux* ;
- une partie T non vide de Q appelée *ensemble des états terminaux* ;
- une relation τ de $Q \times A$ vers Q ou de $Q \times (A \cup \{\epsilon\})$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



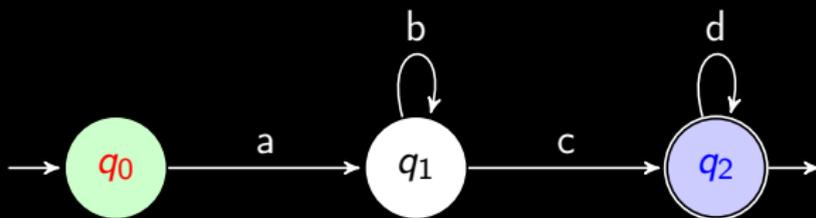
- transition
- étiquette
- origine
- extrémité terminale



- 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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - **Langage reconnaissable**
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

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 :

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

que l'on note aussi

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

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

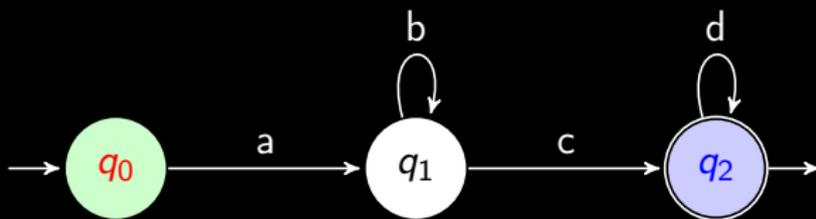
Un **calcul** c de l'automate \mathcal{A} (ou une exécution ou une trace de l'automate \mathcal{A}) est une suite de transitions (des relations de $Q \times A$ dans Q) telles que l'origine de chacune (sauf la première) coïncide avec l'extrémité terminale de la précédente :

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

que l'on note aussi

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

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 :

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

et on dit qu'un langage L est **reconnaisable par l'automate** \mathcal{A} si, et seulement si, $\mathcal{L}(\mathcal{A}) = L$.

Un langage est **reconnaisable** si, et seulement si, il existe un automate qui le reconnaît.

L'ensemble des langages reconnaissables de A^* est noté $\text{Rec}(A^*)$.

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - **Langage reconnu**
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

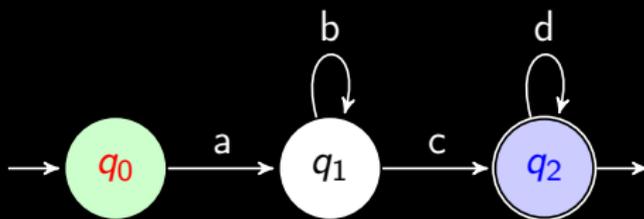
Méthode « manuelle »

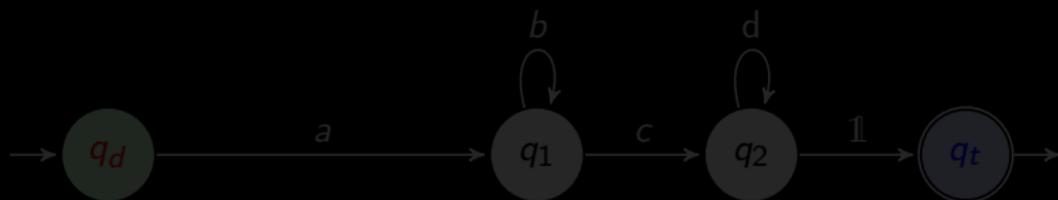
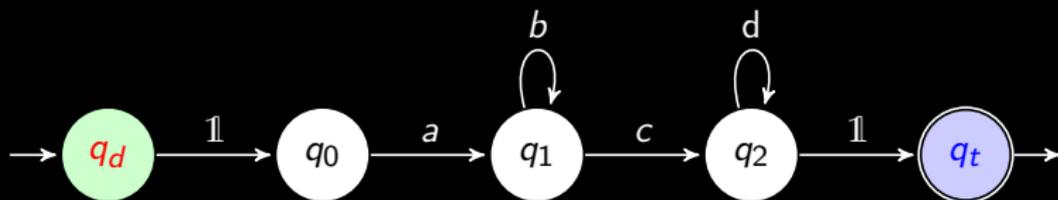
Technique d'élimination d'états.

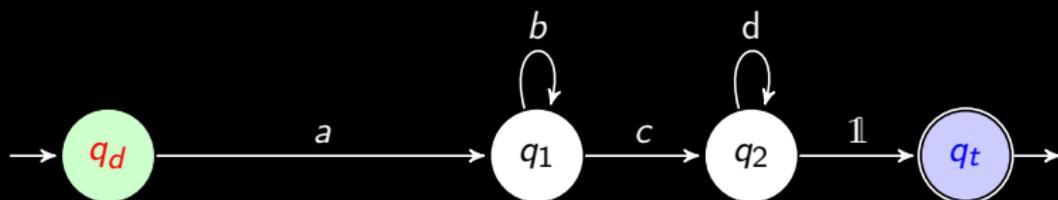
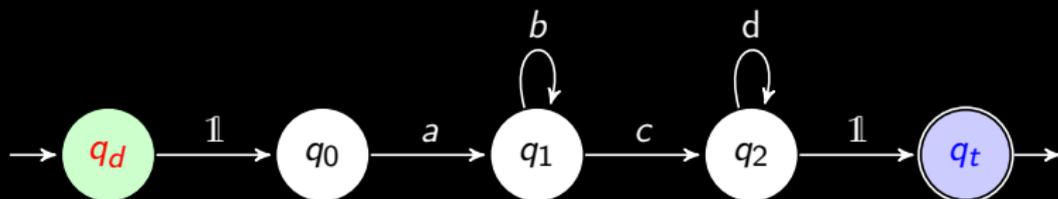
- On commence par créer deux nouveaux états q_d et q_t qui seront les uniques états initiaux et terminaux.
On relie q_d à tous les états initiaux par une transition étiquetée par $\mathbb{1}$ et on relie tous les états terminaux à q_t par le même type de transitions.
- Notons Q l'ensemble tel que l'alphabet 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 $L_{pqr} = L_{pr} \cup L_{pq}L_{qr}^*$.

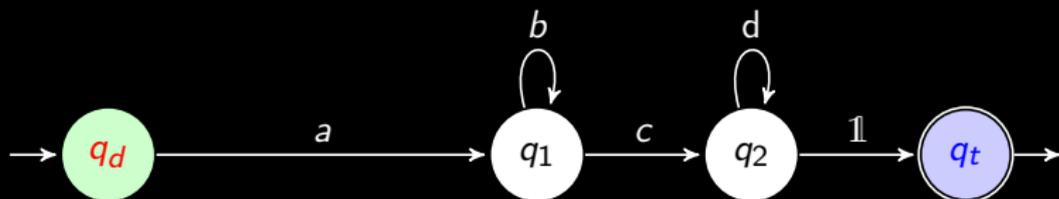
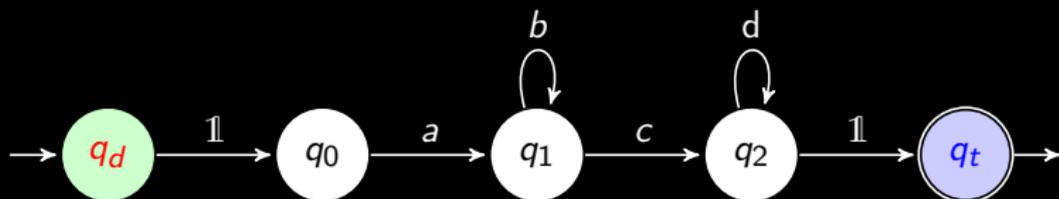
Technique d'élimination d'états.

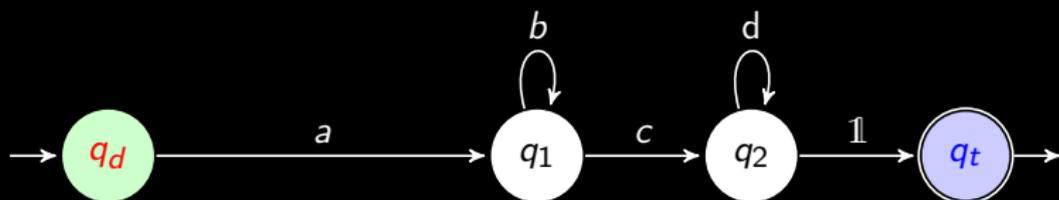
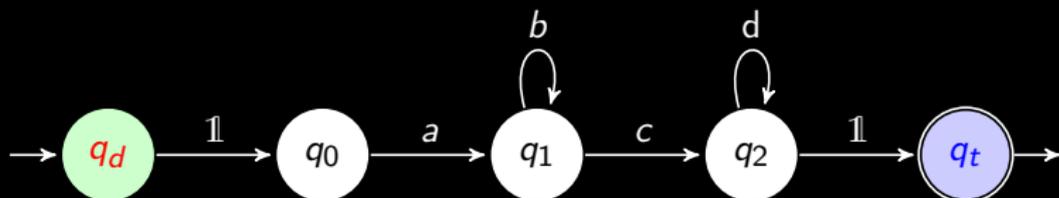
- On commence par créer deux nouveaux états q_d et q_t qui seront les uniques états initiaux et terminaux.
On relie q_d à tous les états initiaux par une transition étiquetée par $\mathbb{1}$ et on relie tous les états terminaux à q_t par le même type de transitions.
- Notons Q l'ensemble tel que l'alphabet 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 $L_{pqr} = L_{pr} \cup L_{pq}L_{qr}^*$.











Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - **Automates équivalents**
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
 - Automate minimal
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- **Automates standards**
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
 - Automate minimal
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 33

Automate standard Un automate $\langle A, Q, D, T, \tau \rangle$ 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 $\mathcal{A}' = \langle A, Q \cup \{d\}, \{d\}, T', \tau' \rangle$ avec :

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

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

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

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

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

Exercice 1

Standardisez l'automate défini sur l'alphabet $\{a, b\}$ par la table de transition suivante :

τ	a	b
q_0	q_1	
q_1	q_1	q_0, q_2
q_2	q_0	q_2

avec $D = Q$ et $T = \{q_0\}$.

Théorème 34

Tout automate fini est équivalent à un automate standard.

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 35

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 35

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 35

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 35

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 35

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 36

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 36

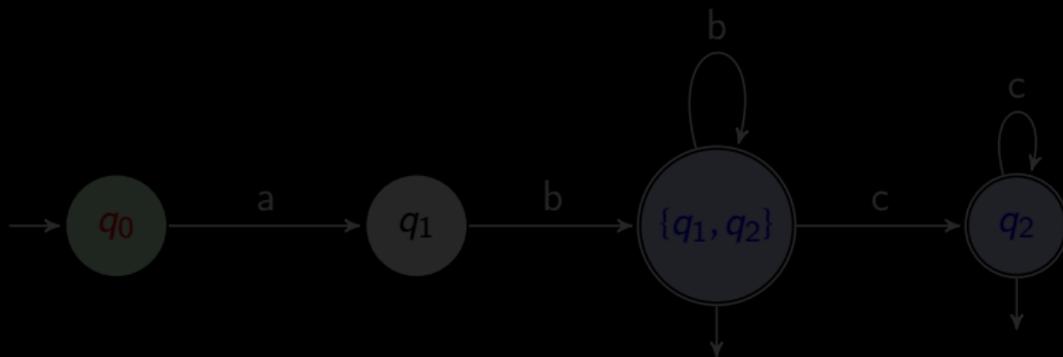
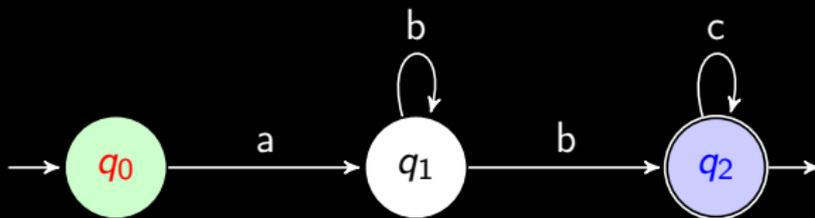
Un automate *déterministe complet* :

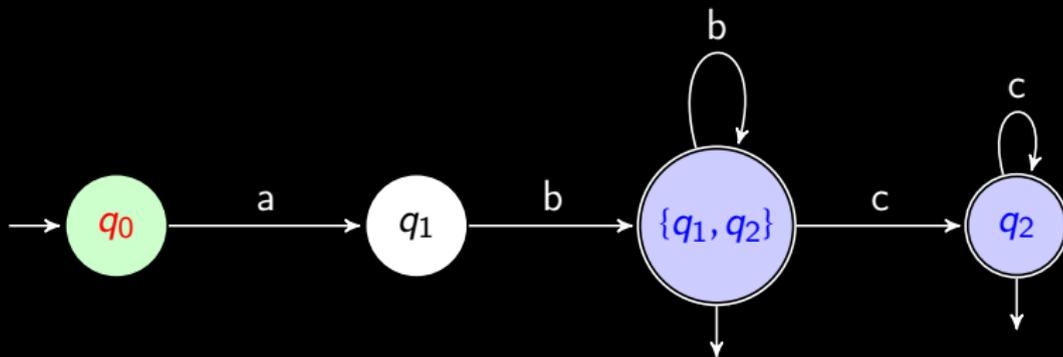
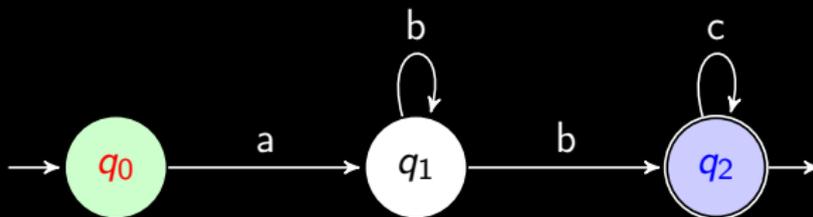
- est un automate déterministe ;
- la relation de transition τ est une application : il y a, à partir de chaque état, une et une seule transition d'étiquette donnée.

Définition 36

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.



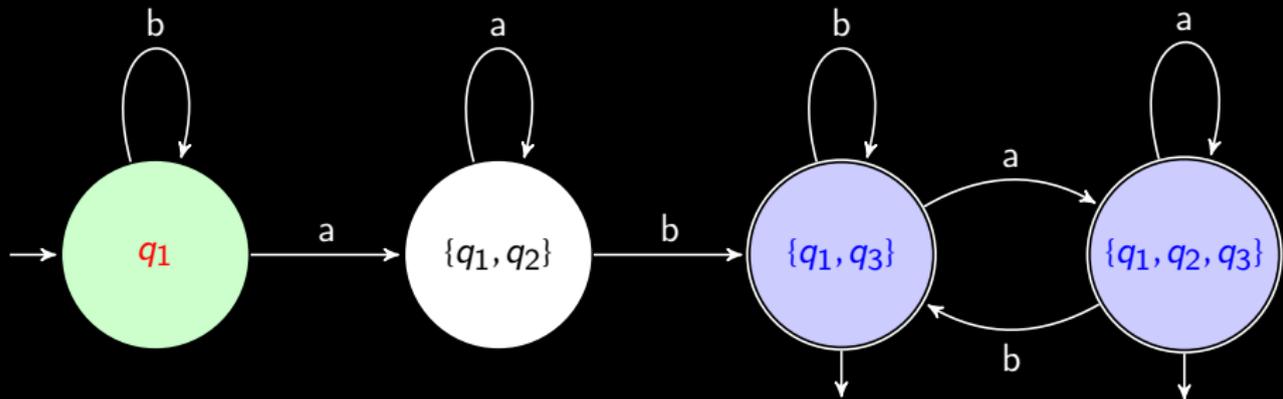


Exercice 2

Déterminez l'automate déterministe équivalent à l'automate défini par la table de transition :

τ	a	b
q_1	q_1, q_2	q_1
q_2		q_3
q_3	q_3	q_3

avec $I = \{q_1\}$ et $T = \{q_3\}$



Théorème 37

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - **Automates émondés**
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
 - Automate minimal
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 38 (États accessibles et co-accessibles)

Soit $\mathcal{A} = \langle A, Q, D, T, \tau \rangle$ 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 38 (États accessibles et co-accessibles)

Soit $\mathcal{A} = \langle A, Q, D, T, \tau \rangle$ 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_j 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 38 (États accessibles et co-accessibles)

Soit $\mathcal{A} = \langle A, Q, D, T, \tau \rangle$ 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 38 (États accessibles et co-accessibles)

Soit $\mathcal{A} = \langle A, Q, D, T, \tau \rangle$ 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 39

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

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

Début

$E \leftarrow D$

$E' \leftarrow \emptyset$

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

$E' \leftarrow E$

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

FinTantQue

Retourner $\langle A, E, D, T, \tau' \rangle$

Fin

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

Début

$C \leftarrow T \cap E$

$C' \leftarrow \emptyset$

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

$C' \leftarrow C$

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

FinTantQue

Retourner $\langle A, C, D, T, \tau'' \rangle$

Fin

Théorème 40

Tout automate fini est équivalent à un automate émondé

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
- **Opérations rationnelles sur les automates**
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile

 - 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
 - 6 Correspondance automate fini / grammaire régulière

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- **Opérations rationnelles sur les automates**
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Soit $\mathcal{A}_1 = \langle A, Q_1, D_1, T_1, \tau_1 \rangle$ et $\mathcal{A}_2 = \langle A, Q_2, D_2, T_2, \tau_2 \rangle$ 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

$$\mathcal{A}_1 | \mathcal{A}_2 = \langle A, Q_1 \cup Q_2, D_1 \cup D_2, T_1 \cup T_2, \tau_1 \cup \tau_2 \rangle$$

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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- **Opérations rationnelles sur les automates**
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Soit $\mathcal{A}_1 = \langle A, Q_1, D_1, T_1, \tau_1 \rangle$ et $\mathcal{A}_2 = \langle A, Q_2, D_2, T_2, \tau_2 \rangle$ 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 :

$$\mathcal{A}_1 \cdot \mathcal{A}_2 = \langle A, (Q_1 \cup Q_2), D_1, T', \tau_1 \cup \tau_2 \cup \tau_2'' \rangle$$

avec :

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

Soit $\mathcal{A}_1 = \langle A, Q_1, D_1, T_1, \tau_1 \rangle$ et $\mathcal{A}_2 = \langle A, Q_2, D_2, T_2, \tau_2 \rangle$ 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 :

$$\mathcal{A}_1 \cdot \mathcal{A}_2 = \langle A, (Q_1 \cup Q_2), D_1, T', \tau_1 \cup \tau_2 \cup \tau_2'' \rangle$$

avec :

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

Théorème 42

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- **Opérations rationnelles sur les automates**
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Soit $\mathcal{A} = \langle A, Q, \{d\}, T, \tau \rangle$ un automate fini *standard* reconnaissant un langage L.

On définit alors un nouvel automate

$$\mathcal{A}^* = \langle A, Q, \{d\}, T \cup \{d\}, \tau' \rangle$$

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- **Théorème de Kleene**
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Théorème 43 (Théorème de Kleene (1956))

Pour tout alphabet A , $\text{Rat}(A^) = \text{Rec}(A^*)$.*

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - **Lemme de la pompe**
 - Automate associé à un langage défini par une expression rationnelle
 - Automate minimal
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes x , y et z telles que

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes ν , ξ et ρ telles que

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes ν , ξ et ρ telles que

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes ν , ξ et ρ telles que

- μ puisse s'écrire sous la forme $\nu\xi\rho$.

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes ν , ξ et ρ telles que

- μ puisse s'écrire sous la forme $\nu\xi\rho$,
- ξ étant non vide,
- pour tout entier naturel n , chaque chaîne $\nu\xi^n\rho$ est acceptée par \mathcal{A} .

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes ν , ξ et ρ telles que

- μ puisse s'écrire sous la forme $\nu\xi\rho$,
- ξ étant non vide,
- pour tout entier naturel n , chaque chaîne $\nu\xi^n\rho$ est acceptée par \mathcal{A} .

Théorème 44 (Lemme de la Pompe : version automate)

Soit \mathcal{A} un automate tel que :

- \mathcal{A} a k états ;
- \mathcal{A} accepte une chaîne μ de longueur $|\mu| > k$;

Alors il existe trois chaînes ν , ξ et ρ telles que

- μ puisse s'écrire sous la forme $\nu\xi\rho$,
- ξ étant non vide,
- pour tout entier naturel n , chaque chaîne $\nu\xi^n\rho$ est acceptée par \mathcal{A} .

Exercice 3 (Application du lemme de la pompe)

Le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas rationnel.

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - **Automate associé à un langage défini par une expression rationnelle**
 - Automate minimal
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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

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

On crée l'ensemble pervy (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_i de l'état q_i vers l'état q_j si le facteur $x_i x_j$ peut apparaître dans au moins une chaîne du langage.

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

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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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

On crée l'ensemble *posledny* (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si $\mathbb{1}$ 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_i de l'état q_i vers l'état q_j si le facteur $x_i x_j$ peut apparaître dans au moins une chaîne du langage.

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

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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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

On crée l'ensemble *posledny* (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si $\mathbb{1}$ 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_j .

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

Finalement, on remplace les x_j par les caractères qu'ils remplacent.

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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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 $\mathbb{1}$ 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_j .

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

Finalement, on remplace les x_j 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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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 $\mathbb{1}$ 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_j .

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

Finalement, on remplace les x_j 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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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

On crée l'ensemble *posledny* (dernier en russe) qui contient tous les caractères pouvant terminer une chaîne plus éventuellement l'état q_0 si $\mathbb{1}$ 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_j de cet ensemble en l'étiquetant par x_j .

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

Finalement, on remplace les x_j 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, $ab^*|a^2b^+$ devient $x_1x_2^*|x_3x_4x_5^+$.

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 $\mathbb{1}$ 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_j de cet ensemble en l'étiquetant par x_j .

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

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
- 5 **Automate minimal**
 - Automates à états finis avec sortie
 - Automates à pile
- 6 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Sommaire

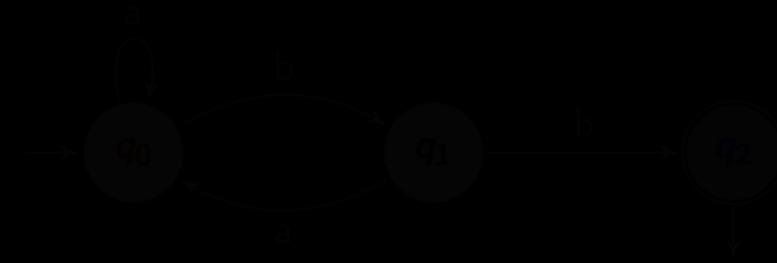
- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
- 5 **Automate minimal**
 - Automates à états finis avec sortie
 - Automates à pile
- 6 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.

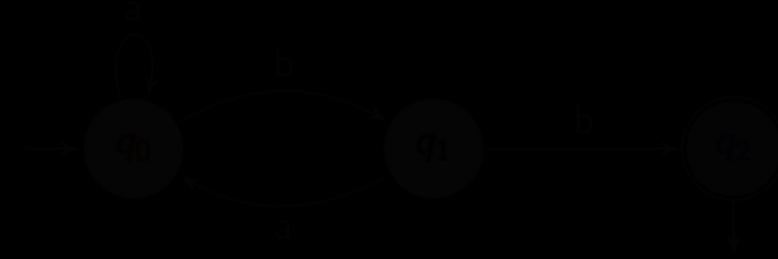


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.

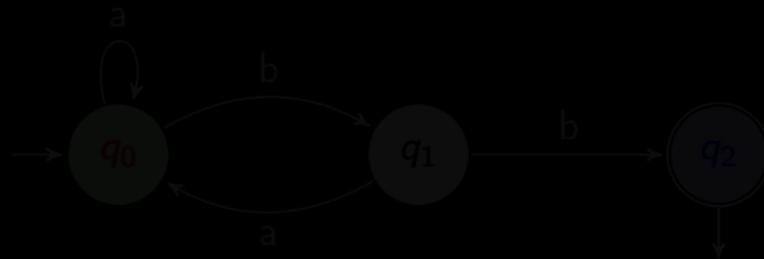


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.

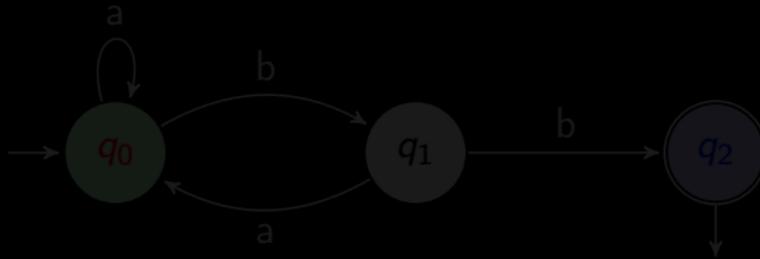


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.

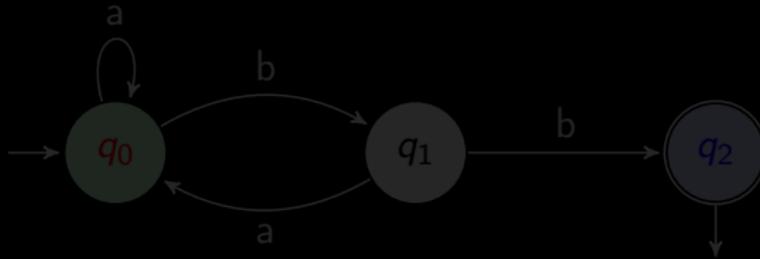


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.

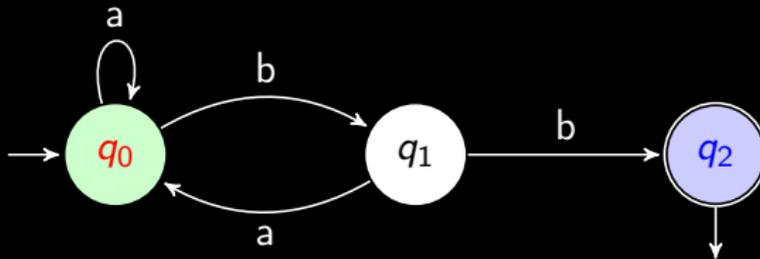


Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.



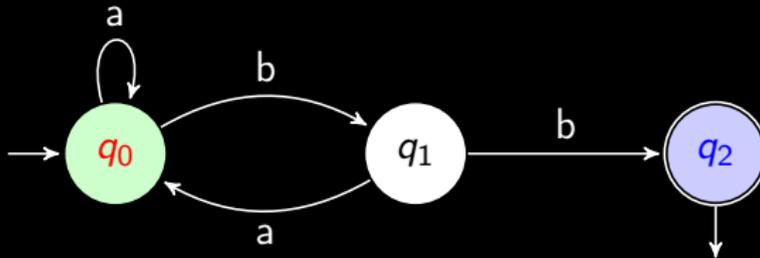
q_0 et q_1 sont-ils distinguables par a ? Par b ?

Définition 45 (États distinguables)

Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ 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.



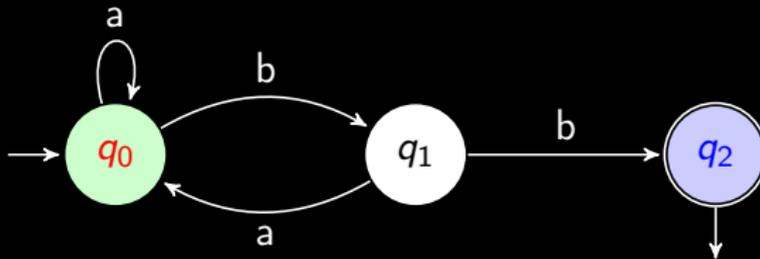
- q_0 et q_1 sont-ils distinguables par a ? Par b ?
- Par quelle chaîne q_1 et q_2 sont-ils distinguables?

Définition 45 (États distinguables)

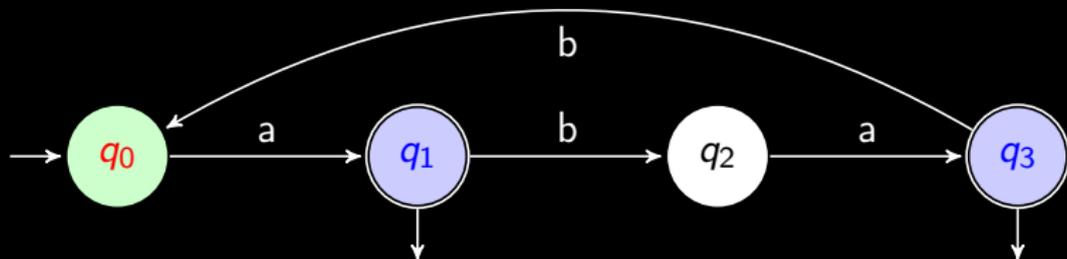
Deux états q_i et q_j dans Q sont distinguables par la chaînes $\mu \in A^*$ si :

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

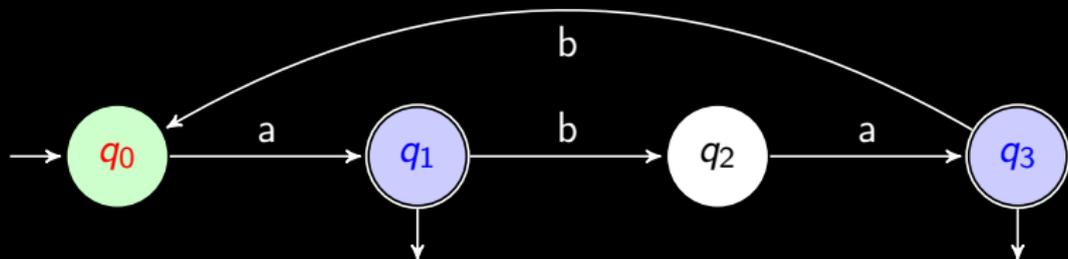
Deux états sont **indistinguables** s'ils ne sont distinguables par aucune chaîne.



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

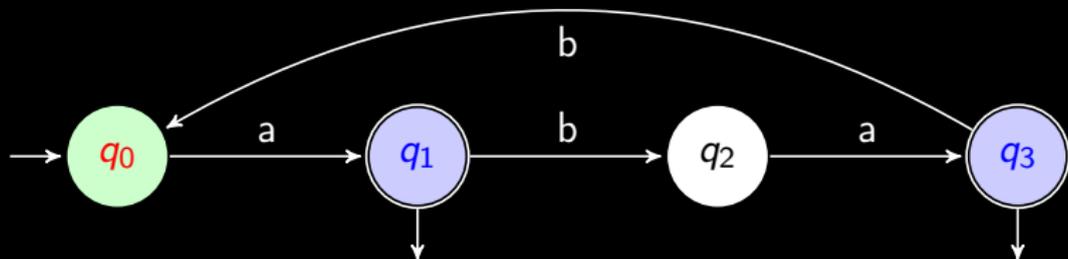


Notons $L(q)$ l'ensemble des chaînes qui sont l'étiquette d'un chemin allant de q à un état terminal : déterminez $L(q_0)$, $L(q_1)$, $L(q_2)$ et $L(q_3)$.
Quel est le langage reconnu par cet automate ?



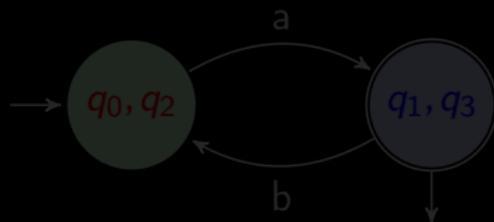
Notons $L(q)$ l'ensemble des chaînes qui sont l'étiquette d'un chemin allant de q à un état terminal : déterminez $L(q_0)$, $L(q_1)$, $L(q_2)$ et $L(q_3)$.

Quel est le langage reconnu par cet automate ?



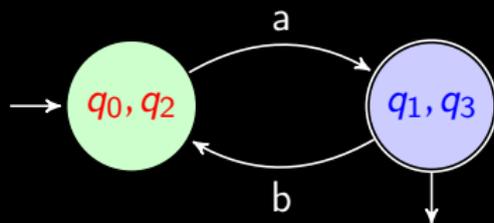
Notons $L(q)$ l'ensemble des chaînes qui sont l'étiquette d'un chemin allant de q à un état terminal : déterminez $L(q_0)$, $L(q_1)$, $L(q_2)$ et $L(q_3)$.
Quel est le langage reconnu par cet automate ?

$L(q_0) = L(q_2)$ et $L(q_1) = L(q_3)$.



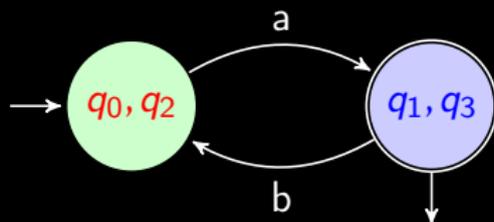
Langage reconnu ?

$L(q_0) = L(q_2)$ et $L(q_1) = L(q_3)$.



Langage reconnu ?

$L(q_0) = L(q_2)$ et $L(q_1) = L(q_3)$.



Langage reconnu ?

Théorème 46 (Équivalence de Nerode)

On définit la relation :

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

Si on note $L(q) = \{a \in A^* \mid \tau(q, a) \in T\}$, cela revient à dire que $L(q_i) = L(q_j)$.
On définit ainsi une relation d'équivalence.

Notons $[q]_N$ la classe d'un état q : la relation \equiv_N est compatible avec la fonction de transition :

$$q_i \equiv_N q_j \Rightarrow \tau(q_i, a) \equiv_N \tau(q_j, a)$$

Théorème 46 (Équivalence de Nerode)

On définit la relation :

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

Si on note $L(q) = \{a \in A^* \mid \tau(q, a) \in T\}$, cela revient à dire que $L(q_i) = L(q_j)$.
On définit ainsi une relation d'équivalence.

Notons $[q]_N$ la classe d'un état q : la relation \equiv_N est compatible avec la fonction de transition :

$$q_i \equiv_N q_j \rightarrow \tau(q_i, a) \equiv_N \tau(q_j, a)$$

Théorème 46 (Équivalence de Nerode)

On définit la relation :

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

Si on note $L(q) = \{a \in A^* \mid \tau(q, a) \in T\}$, cela revient à dire que $L(q_i) = L(q_j)$.
On définit ainsi une relation d'équivalence.

Notons $[q]_N$ la classe d'un état q : la relation \equiv_N est compatible avec la fonction de transition :

$$q_i \equiv_N q_j \rightarrow \tau(q_i, a) \equiv_N \tau(q_j, a)$$

Théorème 46 (Équivalence de Nerode)

On définit la relation :

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

Si on note $L(q) = \{a \in A^* \mid \tau(q, a) \in T\}$, cela revient à dire que $L(q_i) = L(q_j)$.
On définit ainsi une relation d'équivalence.

Notons $[q]_N$ la classe d'un état q : la relation \equiv_N est compatible avec la fonction de transition :

$$q_i \equiv_N q_j \rightarrow \tau(q_i, a) \equiv_N \tau(q_j, a)$$

Théorème 46 (Équivalence de Nerode)

On définit la relation :

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

Si on note $L(q) = \{a \in A^* \mid \tau(q, a) \in T\}$, cela revient à dire que $L(q_i) = L(q_j)$.
On définit ainsi une relation d'équivalence.

Notons $[q]_N$ la classe d'un état q : la relation \equiv_N est compatible avec la fonction de transition :

$$q_i \equiv_N q_j \rightarrow \tau(q_i, a) \equiv_N \tau(q_j, a)$$

On peut donc définir l'automate de Nérode \mathcal{A}_N associé à \mathcal{A} :

$$\mathcal{A}_N = (A, [Q]_N, [q_0]_N, [T]_N, \tau_N)$$

avec

- $[Q]_N = \{[q]_N \mid q \in Q\}$
- $[T]_N = \{[q]_N \mid q \in T\}$
- $\tau_N([q]_N, a) = [\tau(q, a)]$

On peut donc définir l'automate de Nérode \mathcal{A}_N associé à \mathcal{A} :

$$\mathcal{A}_N = (A, [Q]_N, [q_0]_N, [T]_N, \tau_N)$$

avec

- $[Q]_N = \{[q]_N \mid q \in Q\}$
- $[T]_N = \{[q]_N \mid q \in T\}$
- $\tau_N([q]_N, a) = [\tau(q, a)]$

On peut donc définir l'automate de Nérode \mathcal{A}_N associé à \mathcal{A} :

$$\mathcal{A}_N = (A, [Q]_N, [q_0]_N, [T]_N, \tau_N)$$

avec

- $[Q]_N = \{[q]_N \mid q \in Q\}$
- $[T]_N = \{[q]_N \mid q \in T\}$
- $\tau_N([q]_N, a) = [\tau(q, a)]$

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 de Nérode associé. On l'appellera désormais automate minimal de L .

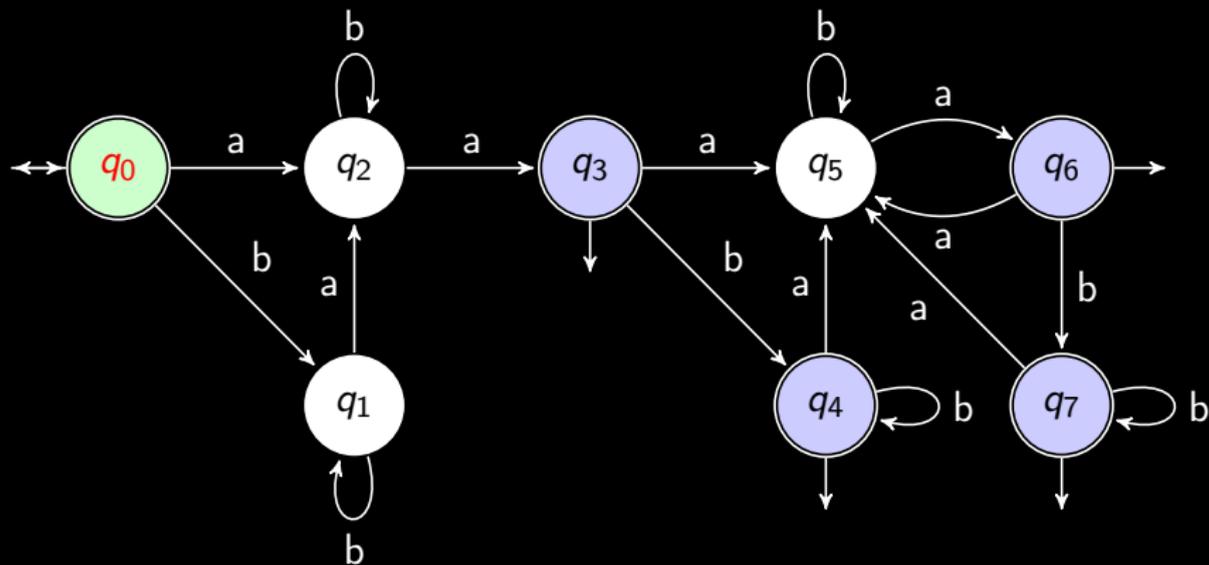
Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
- **Automate minimal**
 - Automates à états finis avec sortie
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

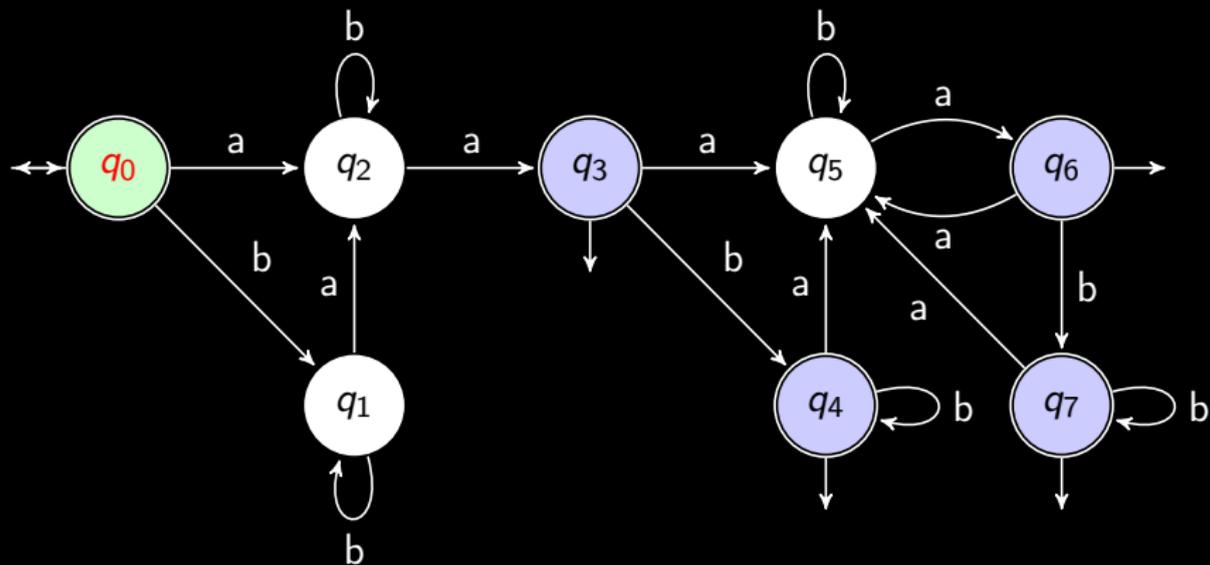
On va en fait travailler par approximation successives de $equiv_N$ par des relations de plus en plus fines, en considérant les relations \equiv_N^P :

$$q \equiv_N^P q' \rightarrow L(q) \cap A^{\leq P} = L(q') \cap A^{\leq P}$$

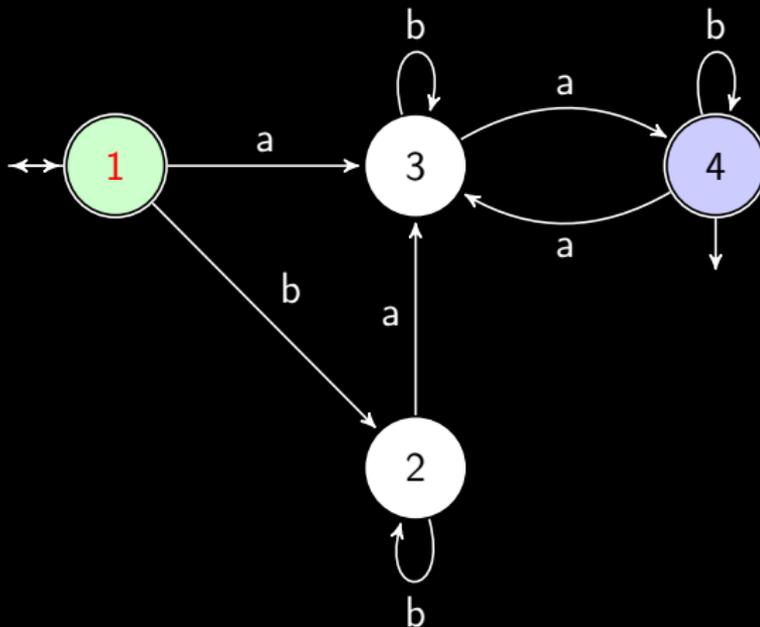
On peut montrer que la suite de ces relations d'équivalences stationne vers \equiv_N dès que deux relations consécutives sont égales ce qui va nous donner un algorithme pratique pour minimaliser un automate.



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



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



Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
 - Automates standards
 - Automates déterministes
 - Automates émondés
 - Opérations rationnelles sur les automates
 - Théorème de Kleene
 - Lemme de la pompe
 - Automate associé à un langage défini par une expression rationnelle
 - Automate minimal
 - **Automates à états finis avec sortie**
 - Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 48

Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ 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 48

Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ 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 48

Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ 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 48

Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ 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 48

Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ 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 48

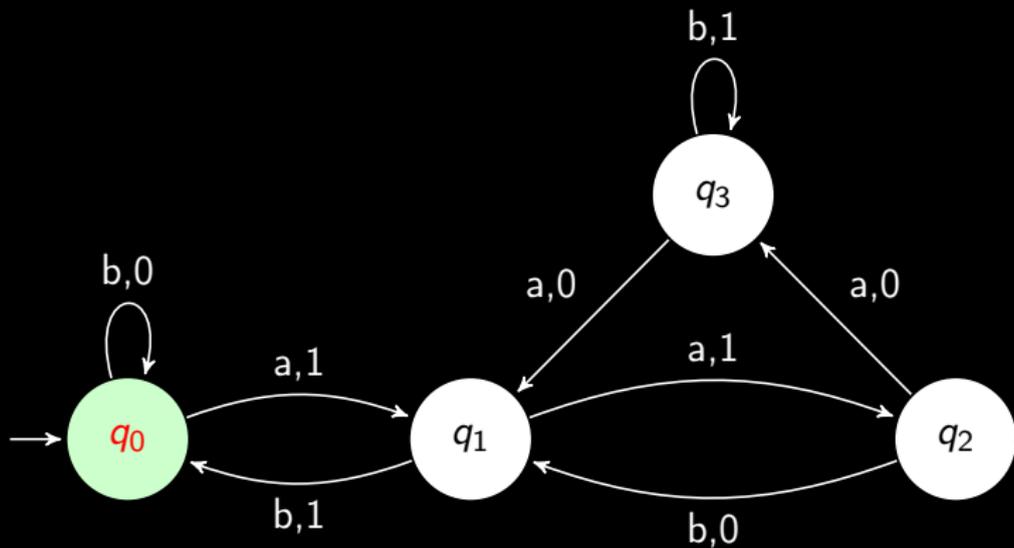
Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ où :

- 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 48

Un automate à états finis avec sorties est caractérisé par un sextuplet $\langle Q, q_0, A_I, A_S, f_t, f_s \rangle$ 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 .



Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 **Automates**
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- **Automates à pile**
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Considérons l'automate défini par

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

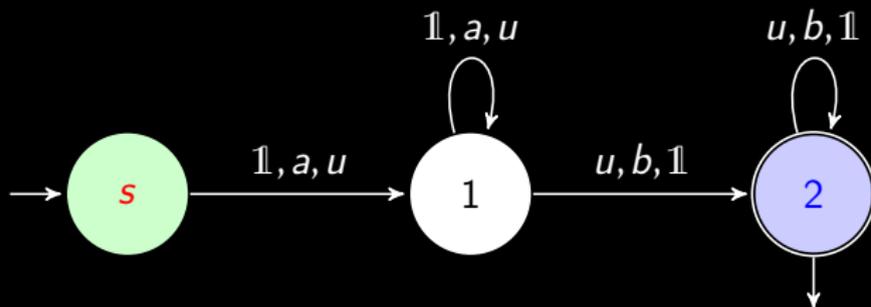
et les transitions

$$T_1 : ((s, 1), a, (1, u))$$

$$T_2 : ((1, 1), a, (1, u))$$

$$T_3 : ((1, u), b, (2, 1))$$

$$T_4 : ((2, u), b, (2, 1))$$



Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - **Syntaxe et sémantique**
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

*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

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - Syntaxe et sémantique
 - **Grammaires algébriques (ou hors contexte)**
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - Syntaxe et sémantique
 - **Grammaires algébriques (ou hors contexte)**
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 49

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 *regles de production* (ou de *réécriture*) qui sont de la forme $A \rightarrow w$ avec $A \in V$ et $w \in (\Sigma \cup V)^*$.

Définition 49

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 $w \in (\Sigma \cup V)^*$.

Définition 49

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 $w \in (\Sigma \cup V)^*$.

Définition 49

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 $w \in (\Sigma \cup V)^*$.

Définition 49

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 $w \in (\Sigma \cup V)^*$.

Définition 50

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 $\nu \xRightarrow{n} \mu$

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

Définition 50

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 $\nu \xRightarrow{n} \mu$

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

Définition 50

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 $\nu \xRightarrow{n} \mu$

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

Définition 51

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - Syntaxe et sémantique
 - **Grammaires algébriques (ou hors contexte)**
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

- Pour toute règle de la forme $X \rightarrow ABCD\dots$, on construit un arbre de noeud X et de feuilles A, B, C, D, \dots
- Pour toute règle du type $X \rightarrow A|B|C|\dots$, on construit un des arbres de noeud 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 $X \rightarrow ABCD\dots$, on construit un arbre de noeud X et de feuilles A, B, C, D, \dots
- Pour toute règle du type $X \rightarrow A|B|C|\dots$, on construit un des arbres de noeud 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 $X \rightarrow ABCD\dots$, on construit un arbre de nœud X et de feuilles A, B, C, D, \dots
- Pour toute règle du type $X \rightarrow A|B|C|\dots$, 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 52

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - **Grammaire régulière**
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

Définition 53

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

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \mathbb{1}$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 53

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

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \mathbb{1}$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 53

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

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \mathbb{1}$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 53

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

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \mathbb{1}$, avec $A, B \in V$ et $a \in \Sigma$.

Définition 53

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

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \mathbb{1}$, avec $A, B \in V$ et $a \in \Sigma$.

Si L est un langage régulier, alors $L(G)$ est régulière.

Tout langage régulier peut être engendré par une grammaire régulière.

Définition 53

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

- $A \rightarrow aB$;
- $A \rightarrow a$;
- $A \rightarrow \mathbb{1}$, avec $A, B \in V$ et $a \in \Sigma$.

Théorème 54

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

*Tout langage régulier **peut** être engendré par une grammaire régulière.*

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - **Analyse syntaxique (« parsing »)**
- 6 Correspondance automate fini / grammaire régulière

Définition 55

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

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 **Grammaires**
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - **Analyse syntaxique (« parsing »)**
- 6 Correspondance automate fini / grammaire régulière

Début

 $Q \leftarrow [S]; c \leftarrow S$
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 $c \leftarrow$ la tête de Q **FinPour****FinTantQue****Retourner** Ce n'est pas un mot du langage

Sommaire

- 1 Approche historique
- 2 Les machines de Turing
 - Présentation sommaire
 - Premier exemple
 - Définitions
 - Fonctions MT-calculables
 - La théorie en pratique
 - Fonctions récursives
- 3 Langages
 - Un exemple pour découvrir
 - Définitions et notations
 - Langages
 - Langages et expressions rationnels (ou réguliers)
- 4 Automates
 - Définitions
 - Langage reconnaissable
 - Langage reconnu
 - Automates équivalents
- Automates standards
- Automates déterministes
- Automates émondés
- Opérations rationnelles sur les automates
- Théorème de Kleene
- Lemme de la pompe
- Automate associé à un langage défini par une expression rationnelle
- Automate minimal
- Automates à états finis avec sortie
- Automates à pile
- 5 Grammaires
 - Syntaxe et sémantique
 - Grammaires algébriques (ou hors contexte)
 - Grammaire régulière
 - Analyse syntaxique (« parsing »)
- 6 Correspondance automate fini / grammaire régulière

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \mathbb{1}$.
- 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 $A \rightarrow \mathbb{1}$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \mathbb{1}$.
- 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 $A \rightarrow \mathbb{1}$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \mathbb{1}$.
- 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 $A \rightarrow \mathbb{1}$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \mathbb{1}$.
- 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 $A \rightarrow \mathbb{1}$ donne un état terminal.

- On s'arrange pour que la grammaire régulière ait des règles du type $A \rightarrow aB$ ou $A \rightarrow \mathbb{1}$.
- 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 $A \rightarrow \mathbb{1}$ donne un état terminal.