Introduction à la théorie des langages
Généralités

TELECOM Nancy 1A

Alphabets

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Exemples

Les ensembles suivants sont des alphabets

- $A_1 = \{0, 1\}$: alphabet binaire

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Exemples

Les ensembles suivants sont des alphabets

- $A_1 = \{0, 1\}$: alphabet binaire
- $A_2 = \{a, b, \ldots, z, A, B, \ldots Z\}$: alphabet latin

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Exemples

Les ensembles suivants sont des alphabets

- $A_1 = \{0, 1\}$: alphabet binaire
- $A_2 = \{a, b, \ldots, z, A, B, \ldots Z\}$: alphabet latin
- $A_3 = \{a, b, \ldots, z, A, B, \ldots Z, 0, 1, \ldots, 9\}$: alphabet ASCII

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Exemples

Les ensembles suivants sont des alphabets

- $A_1 = \{0, 1\}$: alphabet binaire
- $A_2 = \{a, b, \ldots, z, A, B, \ldots Z\}$: alphabet latin
- $A_3 = \{a, b, \ldots, z, A, B, \ldots Z, 0, 1, \ldots, 9\}$: alphabet ASCII
- $W = A_3 \cup \{\exists,\ \forall,\ \in,\ \subset,\ \cup,\ \cap,\ \emptyset,\ \Rightarrow,\ \Leftrightarrow,\ (,\ ),\ \vee,\ \wedge,\ +,\ -,\ *, \ldots\}$: alphabet des formules mathématiques.

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Exemples

Les ensembles suivants sont des alphabets

- $A_1 = \{0, 1\}$: alphabet binaire
- $A_2 = \{a, b, \ldots, z, A, B, \ldots Z\}$: alphabet latin
- $A_3 = \{a, b, \ldots, z, A, B, \ldots Z, 0, 1, \ldots, 9\}$: alphabet ASCII
- $W = A_3 \cup \{\exists,\ \forall,\ \in,\ \subset,\ \cup,\ \cap,\ \emptyset,\ \Rightarrow,\ \Leftrightarrow,\ (,\ ),\ \vee,\ \wedge,\ +,\ -,\ *, \ldots\}$: alphabet des formules mathématiques.

Définition (Lettres)

Les éléments d'un alphabet A sont appelés des **lettres**.

Alphabets

Définition

On appelle **alphabet** ou **vocabulaire** tout ensemble fini.

Exemples

Les ensembles suivants sont des alphabets

- $A_1 = \{0, 1\}$: alphabet binaire
- $A_2 = \{a, b, \ldots, z, A, B, \ldots Z\}$: alphabet latin
- $A_3 = \{a, b, \ldots, z, A, B, \ldots Z, 0, 1, \ldots, 9\}$: alphabet ASCII
- $W = A_3 \cup \{\exists,\ \forall,\ \in,\ \subset,\ \cup,\ \cap,\ \emptyset,\ \Rightarrow,\ \Leftrightarrow,\ (,\ ),\ \vee,\ \wedge,\ +,\ -,\ *, \ldots\}$: alphabet des formules mathématiques.

Définition (Lettres)

Les éléments d'un alphabet A sont appelés des **lettres**.

Exemple

L'alphabet binaire est composé des lettres 0 et 1.

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Remarques

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Remarques

- α un mot sur A, $\alpha\ :\ [1, n] \rightarrow A$

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Remarques

- α un mot sur A, $\alpha : [1, n] \rightarrow A$
- n s'appelle la longueur du mot α, on note $|\alpha| = n$

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Remarques

- α un mot sur A, $\alpha\ :\ [1, n] \rightarrow A$
- n s'appelle la longueur du mot α, on note $|\alpha| = n$
- pour $i \in [1, n]$ $\alpha(i)$ ou α_i est la ième lettre de α

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Remarques

- α un mot sur A, $\alpha : [1, n] \rightarrow A$
- n s'appelle la longueur du mot α, on note $|\alpha| = n$
- pour $i \in [1, n]$ $\alpha(i)$ ou α_i est la ième lettre de α
- si $n = 0$, α est le **mot vide** que l'on note ϵ (ou $\wedge$ selon les ouvrages) pour un mot α de longueur n tel que pour i dans $[1, n]$ $\alpha(i) = a_i$, on note $\alpha = a_1 \ldots a_n$.
  Par exemple, le mot α de longueur 5 défini par $\alpha(1) = n$, $\alpha(2) = a$, $\alpha(3) = n$, $\alpha(4) = c$, $\alpha(5) = y$, est noté *nancy*.

Mots

Définition (Mot)

Soit A un alphabet, un mot sur A est une suite finie à valeur dans A. L'ensemble des mots de A est noté A^*.

Remarques

- α un mot sur A, $\alpha \ : \ [1, n] \to A$
- n s'appelle la longueur du mot α, on note $|\alpha| = n$
- pour $i \in [1, n]$ $\alpha(i)$ ou α_i est la ième lettre de α
- si $n = 0$, α est le **mot vide** que l'on note ϵ (ou $\wedge$ selon les ouvrages) pour un mot α de longueur n tel que pour i dans $[1, n]$ $\alpha(i) = a_i$, on note $\alpha = a_1 \dots a_n$.
  Par exemple, le mot α de longueur 5 défini par $\alpha(1) = n$, $\alpha(2) = a$, $\alpha(3) = n$, $\alpha(4) = c$, $\alpha(5) = y$, est noté $nancy$.
- on peut confondre toute lettre a de A avec le mot a de A^* de longueur 1

Egalité de deux mots

Définition (Egalité de deux mots)

Deux mots α et β sur l'alphabet A sont égaux ssi

$$\begin{cases} |\,\alpha\,| = |\,\beta\,| = n \\ \forall i \in [1, n],\ \alpha_i = \beta_i \end{cases}$$

Egalité de deux mots

Définition (Egalité de deux mots)

Deux mots α et β sur l'alphabet A sont égaux ssi

$$\begin{cases} |\,\alpha\,| = |\,\beta\,| = n \\ \forall i \in [1, n],\ \ \alpha_i = \beta_i \end{cases}$$

Notation

A un alphabet, $a \in A$ et $\alpha \in A^*$, on note $|\alpha|_a$ le nombre d'occurrences de la lettre a dans le mot α, si $|\alpha| = n$: $|\alpha|_a = card\ \{i \in [1, n], \alpha_i = a\}$

Concaténation des mots

Définition (Concaténation de deux mots)

A un alphabet, la concaténation est une opération définie sur A^* de la façon suivante :

$$\begin{array}{cccc} . & : & A^* \times A^* & \rightarrow & A^* \\ & & (\alpha, \beta) & \mapsto & \alpha.\beta \end{array}$$

où $\alpha = \alpha_1\,\alpha_2 \ldots \alpha_p$, $\beta = \beta_1\,\beta_2 \ldots \beta_q$ et $\alpha.\beta = \alpha_1\,\alpha_2 \ldots \alpha_p \beta_1\,\beta_2 \ldots \beta_q$

Concaténation des mots

Définition (Concaténation de deux mots)

A un alphabet, la concaténation est une opération définie sur A^* de la façon suivante :

$$\begin{array}{cccc} . & : & A^* \times A^* & \rightarrow & A^* \\ & & (\alpha, \beta) & \mapsto & \alpha.\beta \end{array}$$

où $\alpha = \alpha_1\, \alpha_2 \ldots \alpha_p$, $\beta = \beta_1\, \beta_2 \ldots \beta_q$ et $\alpha.\beta = \alpha_1\, \alpha_2 \ldots \alpha_p \beta_1\, \beta_2 \ldots \beta_q$

Proposition

L'opération de concaténation possède les propriétés suivantes :

1. $\forall \alpha, \beta \in A^*,\ |\,\alpha.\beta\,| = |\,\alpha\,| + |\,\beta\,|$
2. $\forall \alpha, \beta, \gamma \in A^*,\ (\alpha.\beta).\gamma = \alpha.(\beta.\gamma)$ (l'opération . est associative)
3. $\forall \alpha \in A^*,\ \alpha.\varepsilon = \varepsilon.\alpha = \alpha$ (le mot vide ε est l'élément neutre de la concaténation)
4. $\forall \alpha \in A^*,\ \alpha.\alpha = \alpha \Leftrightarrow \alpha = \varepsilon$ (le mot vide ε est le seul mot idempotent)

Définitions (Facteur, facteur propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **facteur** de β si

$$\exists \gamma, \delta \in A^*,\ \beta = \gamma.\alpha.\delta$$

Si de plus $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **facteur propre** de β.

Définitions (Facteur, facteur propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **facteur** de β si

$$\exists \gamma, \delta \in A^*, \; \beta = \gamma.\alpha.\delta$$

Si de plus $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **facteur propre** de β.

Exemple

an est un facteur propre de *nancy*

Définitions (Facteur, facteur propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **facteur** de β si

$$\exists \gamma, \delta \in A^*,\ \beta = \gamma.\alpha.\delta$$

Si de plus $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **facteur propre** de β.

Exemple

an est un facteur propre de *nancy*

Définitions (Préfixe, préfixe propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **préfixe** de β et l'on note $\alpha \sqsubseteq \beta$ si

$$\exists \delta \in A^*,\ \beta = \alpha.\delta$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **préfixe propre** de β et l'on note alors $\alpha \sqsubset \beta$.

Définitions (Facteur, facteur propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **facteur** de β si

$$\exists \gamma, \delta \in A^*,\ \beta = \gamma.\alpha.\delta$$

Si de plus $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **facteur propre** de β.

Exemple

an est un facteur propre de *nancy*

Définitions (Préfixe, préfixe propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **préfixe** de β et l'on note $\alpha \sqsubseteq \beta$ si

$$\exists \delta \in A^*,\ \beta = \alpha.\delta$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **préfixe propre** de β et l'on note alors $\alpha \sqsubset \beta$.

Exemple

Mars est un préfixe de *Marseille*, et c'est un préfixe propre car *Mars* $\neq$ *Marseille* et *Mars* $\neq \varepsilon$

Remarque

La relation $\sqsubseteq$ est une relation d'ordre :

(*i*) $\forall \alpha \in A^*, \ \alpha \sqsubseteq \alpha$ (refléxivité)

(*ii*) $\forall \alpha, \beta, \gamma \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \gamma$ alors $\alpha \sqsubseteq \gamma$ (transitivité)

(*iii*) $\forall \alpha, \beta \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \alpha$ alors $\alpha = \beta$ (antisymétrie)

Remarque

La relation $\sqsubseteq$ est une relation d'ordre :

(*i*) $\forall \alpha \in A^*, \ \alpha \sqsubseteq \alpha$ (refléxivité)

(*ii*) $\forall \alpha, \beta, \gamma \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \gamma$ alors $\alpha \sqsubseteq \gamma$ (transitivité)

(*iii*) $\forall \alpha, \beta \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \alpha$ alors $\alpha = \beta$ (antisymétrie)

Définitions (Suffixe, suffixe propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **suffixe** de β si

$$\exists \gamma \in A^*, \ \beta = \gamma . \alpha$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **suffixe propre** de β.

Remarque

La relation $\sqsubseteq$ est une relation d'ordre :

(*i*) $\forall \alpha \in A^*, \ \alpha \sqsubseteq \alpha$ (refléxivité)

(*ii*) $\forall \alpha, \beta, \gamma \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \gamma$ alors $\alpha \sqsubseteq \gamma$ (transitivité)

(*iii*) $\forall \alpha, \beta \in A^*$, si $\alpha \sqsubseteq \beta$ et $\beta \sqsubseteq \alpha$ alors $\alpha = \beta$ (antisymétrie)

Définitions (Suffixe, suffixe propre)

Soit A un alphabet. Soient $\alpha, \beta \in A^*$, on dit que α est **suffixe** de β si

$$\exists \gamma \in A^*, \ \beta = \gamma.\alpha$$

Si $\alpha \neq \beta$ et $\alpha \neq \varepsilon$, alors α est dit **suffixe propre** de β.

Exemple

eaux est un suffixe propre de *Bordeaux*

Langages

Définition (Langage)

Soit A un alphabet. On appelle langage sur A toute partie (sous-ensemble) L de A^*.
L'ensemble des langages sur A est donc :

$$\mathcal{P}(A^*) = \{L,\ L \subset A^*\}$$

Un langage sur un alphabet est donc un ensemble de mots sur cet alphabet.

Langages

Définition (Langage)

Soit A un alphabet. On appelle langage sur A toute partie (sous-ensemble) L de A^*.
L'ensemble des langages sur A est donc :

$$\mathcal{P}(A^*) = \{L,\ L \subset A^*\}$$

Un langage sur un alphabet est donc un ensemble de mots sur cet alphabet.

Remarque

- A^* est le plus grand langage sur A au sens de l'inclusion.
- $L = \emptyset$ est le langage vide. Il ne contient aucun mot.
- $L = \{\varepsilon\}$ est un langage contenant uniquement le mot vide.

Langages

Définition (Langage)

Soit A un alphabet. On appelle langage sur A toute partie (sous-ensemble) L de A^*. L'ensemble des langages sur A est donc :

$$\mathcal{P}(A^*) = \{L,\ L \subset A^*\}$$

Un langage sur un alphabet est donc un ensemble de mots sur cet alphabet.

Remarque

- A^* est le plus grand langage sur A au sens de l'inclusion.
- $L = \emptyset$ est le langage vide. Il ne contient aucun mot.
- $L = \{\varepsilon\}$ est un langage contenant uniquement le mot vide.

Exemples

- $L = \{aa, ab, ba, bb\}$ est le langage sur l'alphabet $A = \{a, b\}$ composé des mots de longueur 2.
- $L = \{\alpha \in \{a, b\}^*,\ |\,\alpha\,|_a = |\,\alpha\,|_b\}$ est le langage sur l'alphabet $A = \{a, b\}$ composé des mots contenant autant de a que de b.

Opérations ensemblistes sur les langages

Définition (Union de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **union** de L_1 et de L_2 et l'on note $L_1 \cup L_2$ (ou $L_1 + L_2$) le langage défini par :

$$L_1 \cup L_2 = L_1 + L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \textit{ ou } \alpha \in L_2\}$$

Opérations ensemblistes sur les langages

Définition (Union de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **union** de L_1 et de L_2 et l'on note $L_1 \cup L_2$ (ou $L_1 + L_2$) le langage défini par :

$$L_1 \cup L_2 = L_1 + L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \text{ } \mathit{ou} \text{ } \alpha \in L_2\}$$

Remarque

On utilisera indifféremment la notation ensembliste ($\cup$) ou additive ($+$) pour représenter l'opération union.

Opérations ensemblistes sur les langages

Définition (Union de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **union** de L_1 et de L_2 et l'on note $L_1 \cup L_2$ (ou $L_1 + L_2$) le langage défini par :

$$L_1 \cup L_2 = L_1 + L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \textit{ ou } \alpha \in L_2\}$$

Remarque

On utilisera indifféremment la notation ensembliste ($\cup$) ou additive ($+$) pour représenter l'opération union.

Définition (Intersection de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **intersection** de L_1 et de L_2 et l'on note $L_1 \cap L_2$ le langage défini par :

$$L_1 \cap L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \textit{ et } \alpha \in L_2\}$$

Opérations ensemblistes (suite)

Définition (Complémentaire d'un langage)

Soit L un langage sur A. On appelle **complémentaire** de L et l'on note $\overline{L}$ le langage défini par :

$$\overline{L} = A^* \setminus L = \{\alpha \in A^*,\ \alpha \notin L\}$$

Opérations ensemblistes (suite)

Définition (Complémentaire d'un langage)

Soit L un langage sur A. On appelle **complémentaire** de L et l'on note $\overline{L}$ le langage défini par :

$$\overline{L} = A^* \setminus L = \{\alpha \in A^*,\ \alpha \notin L\}$$

Définition (Différence symétrique)

Soient L_1 et L_2 deux langages sur A, on appelle **différence symétrique** (ou **réunion disjointe**) entre L_1 et L_2 et l'on note $L_1 \Delta L_2$ le langage défini par :

$$L_1 \Delta L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \oplus \alpha \in L_2\} = L_1 \cup L_2 \setminus L_1 \cap L_2$$

Opérations ensemblistes (suite)

Définition (Complémentaire d'un langage)

Soit L un langage sur A. On appelle **complémentaire** de L et l'on note $\overline{L}$ le langage défini par :

$$\overline{L} = A^* \setminus L = \{\alpha \in A^*,\ \alpha \notin L\}$$

Définition (Différence symétrique)

Soient L_1 et L_2 deux langages sur A, on appelle **différence symétrique** (ou **réunion disjointe**) entre L_1 et L_2 et l'on note $L_1 \Delta L_2$ le langage défini par :

$$L_1 \Delta L_2 = \{\alpha \in A^*,\ \alpha \in L_1 \oplus \alpha \in L_2\} = L_1 \cup L_2 \setminus L_1 \cap L_2$$

On s'autorise également l'utilisation des autres opérations ensemblistes (produit cartésien, différence, ...) même si l'on n'en rappelle pas les définitions.

Produit de langages

Définition (Produit de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha, \beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Produit de langages

Définition (Produit de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha, \beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Exemple

Soient $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$. $L_1.L_2 = \{a^p b^q,\ p, q \geq 0\}$

Produit de langages

Définition (Produit de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha,\beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Exemple

Soient $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$. $L_1.L_2 = \{a^p b^q,\ p, q \geq 0\}$

Définition

Soit L un langage sur A, on appelle **puissance nième** de L et l'on note L^n le langage défini par :

Produit de langages

Définition (Produit de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha, \beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Exemple

Soient $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$. $L_1.L_2 = \{a^p b^q,\ p, q \geq 0\}$

Définition

Soit L un langage sur A, on appelle **puissance nième** de L et l'on note L^n le langage défini par :

$$\begin{cases} L^0 = & \{\varepsilon\} \\ & \end{cases}$$

Produit de langages

Définition (Produit de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha, \beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Exemple

Soient $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$. $L_1.L_2 = \{a^p b^q,\ p, q \geq 0\}$

Définition

Soit L un langage sur A, on appelle **puissance nième** de L et l'on note L^n le langage défini par :

$$\begin{cases} L^0 = & \{\varepsilon\} \\ L^1 = & L \\ & \end{cases}$$

Produit de langages

Définition (Produit de deux langages)

Soient L_1 et L_2 deux langages sur A, on appelle **produit** (ou **concaténation**) de L_1 et de L_2 et l'on note $L_1.L_2$ le langage défini par :

$$L_1.L_2 = \{\gamma \in A^*,\ \exists(\alpha, \beta) \in L_1 \times L_2,\ \gamma = \alpha.\beta\}$$

Exemple

Soient $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$. $L_1.L_2 = \{a^p b^q,\ p, q \geq 0\}$

Définition

Soit L un langage sur A, on appelle **puissance nième** de L et l'on note L^n le langage défini par :

$$\begin{cases} L^0 = & \{\varepsilon\} \\ L^1 = & L \\ L^n = & L.L^{n-1} \quad \text{si } n > 1 \end{cases}$$

Définition (Itéré d'un langage)

Soit L un langage sur A, on appelle **fermeture itérative** (ou **étoile** ou **fermeture de Kleene**) de L et l'on note L^* le langage défini par :

$$L^* = \bigcup_{n \geq 0} L^n = \sum_{n \geq 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et le mot vide et stable par l'opération de concaténation.

Définition (Itéré d'un langage)

Soit L un langage sur A, on appelle **fermeture itérative** (ou **étoile** ou **fermeture de Kleene**) de L et l'on note L^* le langage défini par :

$$L^* = \bigcup_{n \geq 0} L^n = \sum_{n \geq 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et le mot vide et stable par l'opération de concaténation.

Remarques

Soient $A = \{a, b\}$ un alphabet, $L_1 = \{a\},\ L_2 = \{ab\}$ et $L_3 = A$ trois langages sur A. On a :

Définition (Itéré d'un langage)

Soit L un langage sur A, on appelle **fermeture itérative** (ou **étoile** ou **fermeture de Kleene**) de L et l'on note L^* le langage défini par :

$$L^* = \bigcup_{n \geq 0} L^n = \sum_{n \geq 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et le mot vide et stable par l'opération de concaténation.

Remarques

Soient $A = \{a, b\}$ un alphabet, $L_1 = \{a\},\ L_2 = \{ab\}$ et $L_3 = A$ trois langages sur A. On a :

- $L_1^* = \{a^n,\ n \geq 0\} = \{\varepsilon,\ a,\ aa,\ \ldots,\ a^i, \ldots\}$

Définition (Itéré d'un langage)

Soit L un langage sur A, on appelle **fermeture itérative** (ou **étoile** ou **fermeture de Kleene**) de L et l'on note L^* le langage défini par :

$$L^* = \bigcup_{n \geq 0} L^n = \sum_{n \geq 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et le mot vide et stable par l'opération de concaténation.

Remarques

Soient $A = \{a, b\}$ un alphabet, $L_1 = \{a\},\ L_2 = \{ab\}$ et $L_3 = A$ trois langages sur A. On a :

- $L_1^* = \{a^n,\ n \geq 0\} = \{\varepsilon,\ a,\ aa,\ \ldots,\ a^i, \ldots\}$
- $L_2^* = \{(ab)^n,\ n \geq 0\} = \{\varepsilon,\ ab,\ abab,\ \ldots,\ (ab)^i,\ \ldots,\}$

Définition (Itéré d'un langage)

Soit L un langage sur A, on appelle **fermeture itérative** (ou **étoile** ou **fermeture de Kleene**) de L et l'on note L^* le langage défini par :

$$L^* = \bigcup_{n \geq 0} L^n = \sum_{n \geq 0} L^n$$

Il s'agit du plus petit langage sur A contenant L et le mot vide et stable par l'opération de concaténation.

Remarques

Soient $A = \{a, b\}$ un alphabet, $L_1 = \{a\},\ L_2 = \{ab\}$ et $L_3 = A$ trois langages sur A. On a :

- $L_1^* = \{a^n,\ n \geq 0\} = \{\varepsilon,\ a,\ aa,\ \ldots,\ a^i, \ldots\}$
- $L_2^* = \{(ab)^n,\ n \geq 0\} = \{\varepsilon,\ ab,\ abab,\ \ldots,\ (ab)^i,\ \ldots,\}$
- $L_3^* = A^*$ (l'ensemble de tous les mots construits sur l'alphabet A)

Itéré strict d'un langage

Définition (Itéré strict)

Soient L un langage sur A, on appelle **étoile stricte** (ou **itéré strict**) de L et l'on note L^+ le langage défini par :

$$L^+ = \bigcup_{n>0} L^n = \sum_{n>0} L^n$$

Il s'agit du plus petit langage sur A contenant L et stable par l'opération de concaténation.

Itéré strict d'un langage

Définition (Itéré strict)

Soient L un langage sur A, on appelle **étoile stricte** (ou **itéré strict**) de L et l'on note L^+ le langage défini par :

$$L^+ = \bigcup_{n>0} L^n = \sum_{n>0} L^n$$

Il s'agit du plus petit langage sur A contenant L et stable par l'opération de concaténation.

Remarques

- On a $L^* = L^+ \cup \{\varepsilon\}$
- On a $L^+ = L.L^*$
- $L^* = L^+ \Leftrightarrow \varepsilon \in L$

Langage miroir (ou réfléchi)

Définition (Langage miroir)

Soit L un langage sur A. On appelle **miroir** (ou **réfléchi**) de L et l'on note $\tilde{L}$ le langage défini par :

$$\tilde{L} = \{\phi(\alpha),\ \alpha \in L\}$$

où $\phi : A^* \to A^*$ est l'application définie par :

$$\begin{cases} \phi(\varepsilon) = \varepsilon \\ \forall \alpha \in A^*, a \in A, \phi(a.\alpha) = \phi(\alpha).a \end{cases}$$

Langage miroir (ou réfléchi)

Définition (Langage miroir)

Soit L un langage sur A. On appelle **miroir** (ou **réfléchi**) de L et l'on note $\tilde{L}$ le langage défini par :

$$\tilde{L} = \{\phi(\alpha),\ \alpha \in L\}$$

où $\phi : A^* \to A^*$ est l'application définie par :

$$\begin{cases} \phi(\varepsilon) = \varepsilon \\ \forall \alpha \in A^*, a \in A, \phi(a.\alpha) = \phi(\alpha).a \end{cases}$$

Remarque

ϕ est la fonction qui renverse un mot :

$$\phi(\alpha_1\,\alpha_2 \ldots \alpha_n) = \alpha_n \ldots \alpha_2\,\alpha_1$$

Quotient gauche

Définition (Quotient gauche)

Soient L_1 et L_2 deux langages sur A, on appelle **quotient gauche** de L_2 par L_1 et l'on note $L_1^{-1}.L_2$ le langage défini par :

$$L_1^{-1}.L_2 = \{\gamma \in A^*,\ \exists \alpha \in L_1,\ \alpha.\gamma \in L_2\}$$

Quotient gauche

Définition (Quotient gauche)

Soient L_1 et L_2 deux langages sur A, on appelle **quotient gauche** de L_2 par L_1 et l'on note $L_1^{-1}.L_2$ le langage défini par :

$$L_1^{-1}.L_2 = \{\gamma \in A^*,\ \exists \alpha \in L_1,\ \alpha.\gamma \in L_2\}$$

Exemple

$L_1 = \{a,\ ab,\ aabb,\ ababb\}$ et $L_2 = \{ab,\ aa,\ aba,\ baba\}$

Quotient gauche

Définition (Quotient gauche)

Soient L_1 et L_2 deux langages sur A, on appelle **quotient gauche** de L_2 par L_1 et l'on note $L_1^{-1}.L_2$ le langage défini par :

$$L_1^{-1}.L_2 = \{\gamma \in A^*,\ \exists \alpha \in L_1,\ \alpha.\gamma \in L_2\}$$

Exemple

$L_1 = \{a,\ ab,\ aabb,\ ababb\}$ et $L_2 = \{ab,\ aa,\ aba,\ baba\}$
$L_1^{-1}.L_2 = \{b,\ a,\ ba,\ \varepsilon\}$

Déterminer $L_2^{-1}.L_1$.

Quotient gauche

Définition (Quotient gauche)

Soient L_1 et L_2 deux langages sur A, on appelle **quotient gauche** de L_2 par L_1 et l'on note $L_1^{-1}.L_2$ le langage défini par :

$$L_1^{-1}.L_2 = \{\gamma \in A^*,\ \exists \alpha \in L_1,\ \alpha.\gamma \in L_2\}$$

Exemple

$L_1 = \{a,\ ab,\ aabb,\ ababb\}$ et $L_2 = \{ab,\ aa,\ aba,\ baba\}$
$L_1^{-1}.L_2 = \{b,\ a,\ ba,\ \varepsilon\}$

Déterminer $L_2^{-1}.L_1$.

Remarque

On utilise le quotient gauche en théorie des codes

Définition (Quotient droit)

Le **quotient droit** de L_1 par L_2 que l'on note $L_1.L_2^{-1}$ est le langage défini par :

$$L_1.L_2^{-1} = \{\gamma \in A^*,\ \exists \alpha \in L_2,\ \gamma.\alpha \in L_1\}$$

.

Définition (Quotient droit)

Le **quotient droit** de L_1 par L_2 que l'on note $L_1.L_2^{-1}$ est le langage défini par :

$$L_1.L_2^{-1} = \{\gamma \in A^*,\ \exists \alpha \in L_2,\ \gamma.\alpha \in L_1\}$$

.

Exemple

$L_1 = \{abba,\ aab,\ bbaa,\ ab\}$ $L_2 = \{b,\ ab,\ bab,\ \varepsilon\}$

Définition (Quotient droit)

Le **quotient droit** de L_1 par L_2 que l'on note $L_1.L_2^{-1}$ est le langage défini par :

$$L_1.L_2^{-1} = \{\gamma \in A^*,\ \exists \alpha \in L_2,\ \gamma.\alpha \in L_1\}$$

.

Exemple

$L_1 = \{abba,\ aab,\ bbaa,\ ab\}$ $L_2 = \{b,\ ab,\ bab,\ \varepsilon\}$
$L_1.L_2^{-1} = \{aa,\ a,\ \varepsilon,\ abba,\ aab,\ bbaa,\ ab\}$

Propriétés des opérations

Propriétés

L'union est associative, commutative, d'élément neutre $\emptyset$ et d'élément absorbant A^*. Autrement dit, $\forall L, L_1, L_2, L_3 \in \mathcal{P}(A^*)$ (langages sur A) :

- $(L_1 \cup L_2) \cup L_3 = L_1 \cup (L_2 \cup L_3)$
- $L_1 \cup L_2 = L_2 \cup L_1$
- $\emptyset \cup L = L \cup \emptyset = L$
- $L \cup A^* = A^* \cup L = A^*$

Le produit est associatif, d'élément neutre $\{\varepsilon\}$, d'élément absorbant $\emptyset$ et distributif par rapport à l'union (même infinie). Autrement dit, $\forall L, L_1, L_2, L_3 \in \mathcal{P}(A^*)$ (langages sur A) :

- $(L_1.L_2).L_3 = L_1.(L_2.L_3)$
- $\{\varepsilon\}.L = L.\{\varepsilon\} = L$
- $\emptyset.L = L.\emptyset = \emptyset$
- $L_1.(L_2 \cup L_3) = L_1.L_2 \cup L_1.L_3$

Remarque

Le produit n'est pas commutatif en général (sauf si $\mathrm{card}(A) = 1$).
En effet, si $A = \{a, b\}$, $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$:

$$L_1.L_2 = \{a^p b^q,\ p, q \geq 0\} \neq L_2.L_1 = \{b^p a^q,\ p, q \geq 0\}$$

Remarque

Le produit n'est pas commutatif en général (sauf si $\mathrm{card}(A) = 1$).
En effet, si $A = \{a, b\}$, $L_1 = \{a^n,\ n \geq 0\}$ et $L_2 = \{b^n,\ n \geq 0\}$:

$$L_1.L_2 = \{a^p b^q,\ p, q \geq 0\} \neq L_2.L_1 = \{b^p a^q,\ p, q \geq 0\}$$

Proposition

L'opération de fermeture itérative(ainsi que l'étoile stricte) est idempotente, autrement dit, $\forall L \in \mathcal{P}(A^*)$:

- $(L^*)^* = L^*$
- $(L^+)^+ = L^+$

et possède la propriété suivante : $\forall L_1, L_2 \in \mathcal{P}(A^*)$,

$$(L_1^* + L_2^*)^* = (L_1 + L_2)^* = (L_1^*.L_2^*)^*$$

Langages réguliers (ou rationnels)

Définition (Ensemble des langages réguliers)

L'ensemble $\mathcal{R}at(A^*)$ des langages **rationnels** (ou **réguliers**) sur un alphabet A est défini inductivement par :

- la base $B = \{\emptyset,\ \{\varepsilon\}\} \cup \{\{a\},\ a \in A\}$
- l'ensemble des opérations $\mathcal{O}p = \{union,\ produit,\ itéré\}$

Langages réguliers (ou rationnels)

Définition (Ensemble des langages réguliers)

L'ensemble $\mathcal{R}at(A^*)$ des langages **rationnels** (ou **réguliers**) sur un alphabet A est défini inductivement par :

- la base $B = \{\emptyset,\ \{\varepsilon\}\} \cup \{\{a\},\ a \in A\}$
- l'ensemble des opérations $\mathcal{O}p = \{union,\ produit,\ itéré\}$

Remarques

Langages réguliers (ou rationnels)

Définition (Ensemble des langages réguliers)

L'ensemble $\mathcal{R}at(A^*)$ des langages **rationnels** (ou **réguliers**) sur un alphabet A est défini inductivement par :

- la base $B = \{\emptyset, \ \{\varepsilon\}\} \cup \{\{a\}, \ a \in A\}$
- l'ensemble des opérations $\mathcal{O}p = \{union, \ produit, \ it\acute{e}r\acute{e}\}$

Remarques

- On définit aussi parfois la base comme étant l'ensemble de tous les langages finis sur A.

Langages réguliers (ou rationnels)

Définition (Ensemble des langages réguliers)

L'ensemble $\mathcal{R}at(A^*)$ des langages **rationnels** (ou **réguliers**) sur un alphabet A est défini inductivement par :

- la base $B = \{\emptyset,\ \{\varepsilon\}\} \cup \{\{a\},\ a \in A\}$
- l'ensemble des opérations $\mathcal{O}p = \{union,\ produit,\ itéré\}$

Remarques

- On définit aussi parfois la base comme étant l'ensemble de tous les langages finis sur A.
- Pour le pas d'induction on a formellement les assertions suivantes :
  $(\forall L_1 \in \mathcal{R}at(A^*))\ (\forall L_2 \in \mathcal{R}at(A^*))\ L_1 \cup L_2 \in \mathcal{R}at(A^*)$ et $L_1.L_2 \in \mathcal{R}at(A^*)$ et $(L_1)^* \in \mathcal{R}at(A^*)$ et $(L_1)^+ \in \mathcal{R}at(A^*)$.

Langages réguliers (ou rationnels)

Définition (Ensemble des langages réguliers)

L'ensemble $\mathcal{R}at(A^*)$ des langages **rationnels** (ou **réguliers**) sur un alphabet A est défini inductivement par :

- la base $B = \{\emptyset,\ \{\varepsilon\}\} \cup \{\{a\},\ a \in A\}$
- l'ensemble des opérations $\mathcal{O}p = \{union,\ produit,\ it\acute{e}r\acute{e}\}$

Remarques

- On définit aussi parfois la base comme étant l'ensemble de tous les langages finis sur A.
- Pour le pas d'induction on a formellement les assertions suivantes :
  $(\forall L_1 \in \mathcal{R}at(A^*))\ (\forall L_2 \in \mathcal{R}at(A^*))\ L_1 \cup L_2 \in \mathcal{R}at(A^*)$ et $L_1.L_2 \in \mathcal{R}at(A^*)$ et $(L_1)^* \in \mathcal{R}at(A^*)$ et $(L_1)^+ \in \mathcal{R}at(A^*)$.
- La définition inductive de $\mathcal{R}at(A^*)$ permet de définir un principe d'induction sur $\mathcal{R}at(A^*)$.

Expressions régulières (ou rationnelles)

Définition (Ensemble des expressions régulières ou rationnelles)

L'ensemble $\mathcal{R}_A$ des **expressions régulières (ou rationnelles)** sur un alphabet A est défini inductivement de la façon suivante :

- la base $B = \{\emptyset,\ \varepsilon\} \cup \{a,\ a \in A\}$
- $(\forall e_1 \in \mathcal{R}_A)\ (\forall e_2 \in \mathcal{R}_A)\ (e_1 + e_2) \in \mathcal{R}_A$ et $(e_1 e_2) \in \mathcal{R}_A$ et $(e_1)^* \in \mathcal{R}_A$ et $(e_1)^+ \in \mathcal{R}_A$

Expressions régulières (ou rationnelles)

Définition (Ensemble des expressions régulières ou rationnelles)

L'ensemble $\mathcal{R}_A$ des **expressions régulières (ou rationnelles)** sur un alphabet A est défini inductivement de la façon suivante :

- la base $B = \{\emptyset, \ \varepsilon\} \cup \{a, \ a \in A\}$
- $(\forall e_1 \in \mathcal{R}_A)$ $(\forall e_2 \in \mathcal{R}_A)$ $(e_1 + e_2) \in \mathcal{R}_A$ et
  $(e_1 e_2) \in \mathcal{R}_A$ et $(e_1)^* \in \mathcal{R}_A$ et $(e_1)^+ \in \mathcal{R}_A$

Remarques

Expressions régulières (ou rationnelles)

Définition (Ensemble des expressions régulières ou rationnelles)

L'ensemble $\mathcal{R}_A$ des **expressions régulières (ou rationnelles)** sur un alphabet A est défini inductivement de la façon suivante :

- la base $B = \{\emptyset, \ \varepsilon\} \cup \{a, \ a \in A\}$
- $(\forall e_1 \in \mathcal{R}_A) \ (\forall e_2 \in \mathcal{R}_A) \ (e_1 + e_2) \in \mathcal{R}_A$ et $(e_1 e_2) \in \mathcal{R}_A$ et $(e_1)^* \in \mathcal{R}_A$ et $(e_1)^+ \in \mathcal{R}_A$

Remarques

- Les expressions régulières forment un langage sur l'alphabet $A \cup \{ \ \emptyset, \ \varepsilon, \ +, \ ^*, \ ^+, (, \ )\}$

Expressions régulières (ou rationnelles)

Définition (Ensemble des expressions régulières ou rationnelles)

L'ensemble $\mathcal{R}_A$ des **expressions régulières (ou rationnelles)** sur un alphabet A est défini inductivement de la façon suivante :

- la base $B = \{\emptyset,\ \varepsilon\} \cup \{a,\ a \in A\}$
- $(\forall e_1 \in \mathcal{R}_A)\ (\forall e_2 \in \mathcal{R}_A)\ (e_1 + e_2) \in \mathcal{R}_A$ et $(e_1 e_2) \in \mathcal{R}_A$ et $(e_1)^* \in \mathcal{R}_A$ et $(e_1)^+ \in \mathcal{R}_A$

Remarques

- Les expressions régulières forment un langage sur l'alphabet $A \cup \{\ \emptyset,\ \varepsilon,\ +,\ ^*,\ ^+, (,\ )\}$
- Les expressions régulières sont une notation pour représenter les langages réguliers. Une expression régulière indique comment le langage régulier dénoté par cette expression est construit à partir des ensembles réguliers élémentaires.

Expressions régulières, langages réguliers

Définition

On définit une application $\mathcal{L} : \mathcal{R}_A \to \mathcal{R}at(A^*)$ de l'ensemble des expressions régulières vers l'ensemble des langages réguliers de la manière suivante :

- $\mathcal{L}(\emptyset) = \emptyset$
- $\mathcal{L}(\varepsilon) = \{\varepsilon\}$
- $\mathcal{L}(a) = \{a\}$ pour tout a de A,
- $\mathcal{L}((e_1 + e_2)) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$,
- $\mathcal{L}((e_1 e_2)) = \mathcal{L}(e_1).\mathcal{L}(e_2)$
- $\mathcal{L}((e)^*) = \mathcal{L}(e)^*$
- $\mathcal{L}((e)^+) = \mathcal{L}(e)^+$

Soient α un langage et e une expression régulière tels que $\mathcal{L}(e) = \alpha$, on dit que e dénote α.

Théorème

Un langage est régulier ssi il est dénoté par une expression régulière.

Exemples

Quelques exemples d'expressions régulières :

- L'expression régulière $a^* b$ dénote le langage des mots commençant par un nombre quelconque de a et terminant par la lettre b.
- L'expression régulière $b(a + b)^*$ dénote le langage des mots commençant par la lettre b.
- L'ensemble de tous les mots construits à partir de $A = \{a_1, \ldots, a_n\}$ et que l'on note A^* est dénoté par l'expression régulière $(a_1 + \ldots + a_n)^*$.
- Le langage dénoté par l'expression régulière $(a + b)^* a(a + b)^*$ est le langage des mots composés avec les lettres a et b qui contiennent au moins un a.

Définition d'un code

Définition

On dit qu'un langage non vide $C \subset A^+$ est un code si tout mot de C^* se décompose de manière unique comme produit de mots de C.

Définition d'un code

Définition

On dit qu'un langage non vide $C \subset A^+$ est un code si tout mot de C^* se décompose de manière unique comme produit de mots de C.

La définition précédente peut se reformuler de façon plus formelle de la façon suivante :

Définition

On dit qu'un langage non vide $C \subset A^+$ est un code si
$\forall \alpha = \alpha_1 \, \alpha_2 \ldots \alpha_p, \;\; \beta = \beta_1 \, \beta_2 \ldots \beta_q \in C^*$:

$$\alpha = \beta \Leftrightarrow \begin{cases} p = q \\ \forall i \in [1, p], \;\; \alpha_i = \beta_i \end{cases}$$

Propriétés

Proposition

Soit C un sous-ensemble fini de A^+. Si C est un code, alors tout sous-ensemble de C est un code.

Proposition

Soit C un sous-ensemble fini de A^* :

(*i*) Si C est un code, $\varepsilon \notin C$

(*ii*) Si C est un code, C et $\bigcup_{n \geq 2} C^n$ sont deux ensembles disjoints

(*iii*) Si tous les mots de C sont de même longueur non nulle, alors C est un code. On parle alors de code **uniforme**.

(*iv*) Si aucun mot de C n'est **préfixe** (resp. suffixe) d'un autre mot de C, C est un code. Dans ce cas, le code C est qualifié de code **préfixe** (resp. suffixe).

Exemples de codes

Exemples de codes

- $C = \{aba,\ ab,\ \varepsilon,\ a\}$ n'est pas un code car $\varepsilon \in C$

Exemples de codes

- $C = \{aba,\ ab,\ \varepsilon,\ a\}$ n'est pas un code car $\varepsilon \in C$
- $C = \{aba,\ bba,\ abb,\ baa\}$ est un code uniforme

Exemples de codes

- $C = \{aba,\ ab,\ \varepsilon,\ a\}$ n'est pas un code car $\varepsilon \in C$
- $C = \{aba,\ bba,\ abb,\ baa\}$ est un code uniforme
- $C = \{11,\ 011,\ 0011,\ 0010,\ 0000\}$ est un code préfixe

Exemples de codes

- $C = \{aba,\ ab,\ \varepsilon,\ a\}$ n'est pas un code car $\varepsilon \in C$
- $C = \{aba,\ bba,\ abb,\ baa\}$ est un code uniforme
- $C = \{11,\ 011,\ 0011,\ 0010,\ 0000\}$ est un code préfixe
- $C = \{aa,\ aab,\ aabb,\ babb,\ bbbb\}$ est un code suffixe

Algorithme de Sardinas Patterson

Algorithme

Soit $C \in \mathcal{P}(A^*)$ un langage sur A. On cherche à déterminer si C est un code. L'**algorithme de Sardinas et Patterson** consiste en la construction d'une suite d'ensembles :

Algorithme de Sardinas Patterson

Algorithme

Soit $C \in \mathcal{P}(A^*)$ un langage sur A. On cherche à déterminer si C est un code. L'**algorithme de Sardinas et Patterson** consiste en la construction d'une suite d'ensembles :

(*i*) *Initialisation* : $U_0 = C^{-1}.C \setminus \{\varepsilon\}$

Algorithme de Sardinas Patterson

Algorithme

Soit $C \in \mathcal{P}(A^*)$ un langage sur A. On cherche à déterminer si C est un code. L'**algorithme de Sardinas et Patterson** consiste en la construction d'une suite d'ensembles :

(*i*) *Initialisation* : $U_0 = C^{-1}.C \setminus \{\varepsilon\}$

(*ii*) *Itération* : $U_{n+1} = U_n^{-1}.C \cup C^{-1}.U_n$

Algorithme de Sardinas Patterson

Algorithme

Soit $C \in \mathcal{P}(A^*)$ un langage sur A. On cherche à déterminer si C est un code. L'**algorithme de Sardinas et Patterson** consiste en la construction d'une suite d'ensembles :

(*i*) *Initialisation* : $U_0 = C^{-1}.C \setminus \{\varepsilon\}$

(*ii*) *Itération* : $U_{n+1} = U_n^{-1}.C \cup C^{-1}.U_n$

(*iii*) *Condition d'arrêt* : $\left\{ \begin{array}{ll} \varepsilon \in U_n & \Rightarrow C \text{ n'est pas un code} \\ & \end{array} \right.$

Algorithme de Sardinas Patterson

Algorithme

Soit $C \in \mathcal{P}(A^*)$ un langage sur A. On cherche à déterminer si C est un code. L'**algorithme de Sardinas et Patterson** consiste en la construction d'une suite d'ensembles :

(*i*) *Initialisation* : $U_0 = C^{-1}.C \setminus \{\varepsilon\}$

(*ii*) *Itération* : $U_{n+1} = U_n^{-1}.C \cup C^{-1}.U_n$

(*iii*) *Condition d'arrêt* : $\begin{cases} \varepsilon \in U_n & \Rightarrow C \text{ n'est pas un code} \\ (\exists j)\ (\exists i)\ i > j \text{ et } U_i = U_j & \Rightarrow C \text{ est un code} \end{cases}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation :

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2 = \{11\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2 = \{11\}$
   - $U_4 = U_3^{-1}.C \cup C^{-1}.U_3$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2 = \{11\}$
   - $U_4 = U_3^{-1}.C \cup C^{-1}.U_3 = \{011\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$

2. Itération

   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2 = \{11\}$
   - $U_4 = U_3^{-1}.C \cup C^{-1}.U_3 = \{011\}$
   - $U_5 = U_4^{-1}.C \cup C^{-1}.U_4$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$

2. Itération

   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2 = \{11\}$
   - $U_4 = U_3^{-1}.C \cup C^{-1}.U_3 = \{011\}$
   - $U_5 = U_4^{-1}.C \cup C^{-1}.U_4 = \{\varepsilon\}$

Exemple d'exécution de l'algorithme de Sardinas Patterson

Déterminer si l'ensemble $C = \{10,\ 011,\ 1001,\ 11011\}$ est un code ou non en utilisant l'algorithme de Sardinas Patterson.

1. Initialisation : $U_0 = C^{-1}.C \setminus \{\varepsilon\} = \{01\}$
2. Itération
   - $U_1 = U_0^{-1}.C \cup C^{-1}.U_0 = \{1\}$
   - $U_2 = U_1^{-1}.C \cup C^{-1}.U_1 = \{0,\ 001,\ 1011\}$
   - $U_3 = U_2^{-1}.C \cup C^{-1}.U_2 = \{11\}$
   - $U_4 = U_3^{-1}.C \cup C^{-1}.U_3 = \{011\}$
   - $U_5 = U_4^{-1}.C \cup C^{-1}.U_4 = \{\varepsilon\}$
3. $\varepsilon \in U_5 \Rightarrow C$ n'est pas un code.

Remarques

Dans le cas où l'ensemble de mots n'est pas un code, l'algorithme de Sardinas Patterson permet de détecter un mot de longueur minimale et qui possède deux décompositions différentes. Pour cela il suffit de considérer les mots ayant permis de génèrer ε. Dans l'exemple on considère le tableau suivant :

C	U_0	U_1	U_2	U_3	U_4	U_5
10 011 1001 11011	01	1	0 001 1011	11	011	ε

On obtient ainsi le mot $m = 10\ 01\ 1\ 0\ 11\ 011$.
Si on nomme α_1, α_2, α_3 et α_4 les mots de C comme suit :

$$C = \{\overset{\alpha_1}{10},\ \overset{\alpha_2}{011},\ \overset{\alpha_3}{1001},\ \overset{\alpha_4}{11011}\}$$

on a les deux décompositions différentes suivantes du mot m :

$$m = \overbrace{10}^{\alpha_1}\overbrace{011}^{\alpha_2}\overbrace{011}^{\alpha_2}\overbrace{011}^{\alpha_2} = \overbrace{1001}^{\alpha_3}\overbrace{10}^{\alpha_1}\overbrace{11011}^{\alpha_4}$$