Chapitre 1 : Problèmes et langages

Vincent Guisse
vincent.guisse@esisar.grenoble-inp.fr

Grenoble INP - ESISAR
3[e] année IR & C

15 février 2022

Organisation

- Volume horaire
  - 10 cours (15 h)
  - 7 TD (10.5 h)
- Crédits : 3 ECTS
- UE : CS 322 et CS 353 (Algorithmique et structure de données 5 ECTS)
- Modalités d'évaluation : 0,25 CC (partiel 1h le 12/04, date à confirmer) + 0,75 Ex

Organisation

- Volume horaire
  - 10 cours (15 h)
  - 7 TD (10.5 h)
- Crédits : 3 ECTS
- UE : CS 322 et CS 353 (Algorithmique et structure de données 5 ECTS)
- Modalités d'évaluation : 0,25 CC (partiel 1h le 12/04, date à confirmer) + 0,75 Ex

Organisation

- Volume horaire
  - 10 cours (15 h)
  - 7 TD (10.5 h)
- Crédits : 3 ECTS
- UE : CS 322 et CS 353 (Algorithmique et structure de données 5 ECTS)
- Modalités d'évaluation : 0,25 CC (partiel 1h le 12/04, date à confirmer) + 0,75 Ex

Prérequis et objectifs

- Pré-requis : Ensembles, relations, applications, raisonnement par récurrence.
- Objectifs :
  - Théoriques : comprendre les fondements théoriques de l'informatique, en particulier les notions de langage, de problème, de modèle de calcul, de calculabilité.
  - Pratiques : connaître les automates finis, les langages et les expressions régulières, les grammaires hors-contexte utilisés en compilation, analyse lexicale et syntaxique.
  - Ce cours n'abordera pas la question de la complexité des algorithmes ou des problèmes.
- Ce cours est un pré-requis au cours « langages et compilation » (CS 444).

Prérequis et objectifs

- Pré-requis : Ensembles, relations, applications, raisonnement par récurrence.
- Objectifs :
  - Théoriques : comprendre les fondements théoriques de l'informatique, en particulier les notions de langage, de problème, de modèle de calcul, de calculabilité.
  - Pratiques : connaître les automates finis, les langages et les expressions régulières, les grammaires hors-contexte utilisés en compilation, analyse lexicale et syntaxique.
  - Ce cours n'abordera pas la question de la complexité des algorithmes ou des problèmes.
- Ce cours est un pré-requis au cours « langages et compilation » (CS 444).

Prérequis et objectifs

- Pré-requis : Ensembles, relations, applications, raisonnement par récurrence.
- Objectifs :
  - Théoriques : comprendre les fondements théoriques de l'informatique, en particulier les notions de langage, de problème, de modèle de calcul, de calculabilité.
  - Pratiques : connaître les automates finis, les langages et les expressions régulières, les grammaires hors-contexte utilisés en compilation, analyse lexicale et syntaxique.
  - Ce cours n'abordera pas la question de la complexité des algorithmes ou des problèmes.
- Ce cours est un pré-requis au cours ≪ langages et compilation ≫ (CS 444).

Prérequis et objectifs

- Pré-requis : Ensembles, relations, applications, raisonnement par récurrence.
- Objectifs :
  - Théoriques : comprendre les fondements théoriques de l'informatique, en particulier les notions de langage, de problème, de modèle de calcul, de calculabilité.
  - Pratiques : connaître les automates finis, les langages et les expressions régulières, les grammaires hors-contexte utilisés en compilation, analyse lexicale et syntaxique.
  - Ce cours n'abordera pas la question de la complexité des algorithmes ou des problèmes.
- Ce cours est un pré-requis au cours « langages et compilation » (CS 444).

Prérequis et objectifs

- Pré-requis : Ensembles, relations, applications, raisonnement par récurrence.
- Objectifs :
  - Théoriques : comprendre les fondements théoriques de l'informatique, en particulier les notions de langage, de problème, de modèle de calcul, de calculabilité.
  - Pratiques : connaître les automates finis, les langages et les expressions régulières, les grammaires hors-contexte utilisés en compilation, analyse lexicale et syntaxique.
  - Ce cours n'abordera pas la question de la complexité des algorithmes ou des problèmes.
- Ce cours est un pré-requis au cours ≪ langages et compilation ≫ (CS 444).

Question fondamentale

Quels sont les problèmes résolubles par une procédure effective ?

- Qu'est-ce qu'un problème ? Réponse simple avec la notion de langage.
- Qu'est-ce qu'une procédure effective ? Pas de réponse définitive, plusieurs réponses avec plusieurs modélisations :
  - Automates à états finis (mémoire finie)
  - Automates à pile (mémoire infinie en accès LIFO)
  - Machines de Turing (mémoire infinie sans limitation d'accès)
  - Fonctions récursives, lambda-calcul, machine RAM, équivalents à la machine de Turing, ne seront pas abordés.
- La thèse de Church-Turing :

  "Les problèmes résolubles par une procédure effective sont ceux résolus par une machine de Turing"

  est non contredite à ce jour.

Question fondamentale

Quels sont les problèmes résolubles par une procédure effective ?

- Qu'est-ce qu'un problème ? Réponse simple avec la notion de langage.
- Qu'est-ce qu'une procédure effective ? Pas de réponse définitive, plusieurs réponses avec plusieurs modélisations :
  - Automates à états finis (mémoire finie)
  - Automates à pile (mémoire infinie en accès LIFO)
  - Machines de Turing (mémoire infinie sans limitation d'accès)
  - Fonctions récursives, lambda-calcul, machine RAM, équivalents à la machine de Turing, ne seront pas abordés.
- La thèse de Church-Turing :

"Les problèmes résolubles par une procédure effective sont ceux résolus par une machine de Turing"

est non contredite à ce jour.

Question fondamentale

Quels sont les problèmes résolubles par une procédure effective ?

- Qu'est-ce qu'un problème ? Réponse simple avec la notion de langage.
- Qu'est-ce qu'une procédure effective ? Pas de réponse définitive, plusieurs réponses avec plusieurs modélisations :
  - Automates à états finis (mémoire finie)
  - Automates à pile (mémoire infinie en accès LIFO)
  - Machines de Turing (mémoire infinie sans limitation d'accès)
  - Fonctions récursives, lambda-calcul, machine RAM, équivalents à la machine de Turing, ne seront pas abordés.
- La thèse de Church-Turing :

  "Les problèmes résolubles par une procédure effective sont ceux résolus par une machine de Turing"

  est non contredite à ce jour.

Question fondamentale

Quels sont les problèmes résolubles par une procédure effective ?

- Qu'est-ce qu'un problème ? Réponse simple avec la notion de langage.
- Qu'est-ce qu'une procédure effective ? Pas de réponse définitive, plusieurs réponses avec plusieurs modélisations :
  - Automates à états finis (mémoire finie)
  - Automates à pile (mémoire infinie en accès LIFO)
  - Machines de Turing (mémoire infinie sans limitation d'accès)
  - Fonctions récursives, lambda-calcul, machine RAM, équivalents à la machine de Turing, ne seront pas abordés.
- La thèse de Church-Turing :

  "Les problèmes résolubles par une procédure effective sont ceux résolus par une machine de Turing"

  est non contredite à ce jour.

Question fondamentale

Quels sont les problèmes résolubles par une procédure effective ?

- Qu'est-ce qu'un problème ? Réponse simple avec la notion de langage.
- Qu'est-ce qu'une procédure effective ? Pas de réponse définitive, plusieurs réponses avec plusieurs modélisations :
  - Automates à états finis (mémoire finie)
  - Automates à pile (mémoire infinie en accès LIFO)
  - Machines de Turing (mémoire infinie sans limitation d'accès)
  - Fonctions récursives, lambda-calcul, machine RAM, équivalents à la machine de Turing, ne seront pas abordés.
- La thèse de Church-Turing :

  "Les problèmes résolubles par une procédure effective sont ceux résolus par une machine de Turing"

  est non contredite à ce jour.

Problèmes de décisions

- On se limite aux problèmes de décision : la réponse est oui ou non.
- Ce n'est pas une restriction fondamentale : on peut décider successivement les valeurs des bits d'une réponse plus évoluée par exemple.
- Voyons quelques exemples pour cerner la notion de problème de décision.

Problèmes de décisions

- On se limite aux problèmes de décision : la réponse est oui ou non.
- Ce n'est pas une restriction fondamentale : on peut décider successivement les valeurs des bits d'une réponse plus évoluée par exemple.
- Voyons quelques exemples pour cerner la notion de problème de décision.

Problèmes de décisions

- On se limite aux problèmes de décision : la réponse est oui ou non.
- Ce n'est pas une restriction fondamentale : on peut décider successivement les valeurs des bits d'une réponse plus évoluée par exemple.
- Voyons quelques exemples pour cerner la notion de problème de décision.

Parité

Donnée : Un entier n

Question : L'entier n est-il pair ?

Pour envisager une procédure effective, il va falloir fixer un **encodage** des **instances**. Il y a plusieurs façons de le faire, par exemple :

- En base dix : L'ensemble des instances est l'ensemble des mots sur l'alphabet $\Sigma = \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$, l'ensemble des instances positives est

$$L = \{0\} \cup \{a_1 a_2 \dots a_n \in \Sigma^* \,|a_1 \neq 0 \text{ et } a_n \in \{0, 2, 4, 6, 8\}\}$$

  où Σ^* désigne l'ensemble des mots finis sur l'alphabet Σ.
- En base deux
- En unaire (base un)

Parité

Donnée : Un entier n

Question : L'entier n est-il pair ?

Pour envisager une procédure effective, il va falloir fixer un **encodage** des **instances**. Il y a plusieurs façons de le faire, par exemple :

- En base dix : L'ensemble des instances est l'ensemble des mots sur l'alphabet $\Sigma = \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$, l'ensemble des instances positives est

$$L = \{0\} \cup \{a_1 a_2 \ldots a_n \in \Sigma^* \mid a_1 \neq 0 \text{ et } a_n \in \{0, 2, 4, 6, 8\}\}$$

où Σ^* désigne l'ensemble des mots finis sur l'alphabet Σ.
- En base deux
- En unaire (base un)

Parité

Donnée : Un entier n

Question : L'entier n est-il pair ?

Pour envisager une procédure effective, il va falloir fixer un **encodage** des **instances**. Il y a plusieurs façons de le faire, par exemple :

- En base dix : L'ensemble des instances est l'ensemble des mots sur l'alphabet $\Sigma = \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$, l'ensemble des instances positives est

$$L = \{0\} \cup \{a_1 a_2 \dots a_n \in \Sigma^* \,| a_1 \neq 0 \text{ et } a_n \in \{0, 2, 4, 6, 8\}\}$$

  où Σ^* désigne l'ensemble des mots finis sur l'alphabet Σ.
- En base deux
- En unaire (base un)

Parité

Donnée : Un entier n

Question : L'entier n est-il pair ?

Pour envisager une procédure effective, il va falloir fixer un **encodage** des **instances**. Il y a plusieurs façons de le faire, par exemple :

- En base dix : L'ensemble des instances est l'ensemble des mots sur l'alphabet $\Sigma = \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$, l'ensemble des instances positives est

$$L = \{0\} \cup \{a_1 a_2 \dots a_n \in \Sigma^* \,|a_1 \neq 0 \text{ et } a_n \in \{0, 2, 4, 6, 8\}\}$$

  où Σ^* désigne l'ensemble des mots finis sur l'alphabet Σ.
- En base deux
- En unaire (base un)

Parité

Donnée : Un entier n

Question : L'entier n est-il pair ?

Pour envisager une procédure effective, il va falloir fixer un **encodage** des **instances**. Il y a plusieurs façons de le faire, par exemple :

- En base dix : L'ensemble des instances est l'ensemble des mots sur l'alphabet $\Sigma = \{0; 1; 2; 3; 4; 5; 6; 7; 8; 9\}$, l'ensemble des instances positives est

$$L = \{0\} \cup \{a_1 a_2 \dots a_n \in \Sigma^* \mid a_1 \neq 0 \text{ et } a_n \in \{0, 2, 4, 6, 8\}\}$$

  où Σ^* désigne l'ensemble des mots finis sur l'alphabet Σ.
- En base deux
- En unaire (base un)

Primalité

Donnée : Un entier n encodé en base dix

Question : L'entier n est il premier ?

Les instances sont encore des **mots** sur l'alphabet

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

l'ensemble des instances positives est

$$L = \{2, 5, 7, 11, \dots\}$$

Primalité

Donnée : Un entier n encodé en base dix

Question : L'entier n est il premier ?

Les instances sont encore des **mots** sur l'alphabet

$$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

l'ensemble des instances positives est

$$L = \{2, 5, 7, 11, \dots\}$$

Accessibilité

Donnée : Un graphe orienté $\overrightarrow{G}$ donné sous forme de liste d'adjacence et un couple (x, y)

Question : Le sommet y est-il accessible depuis le sommet x dans $\overrightarrow{G}$?

Exemples d'instances avec $\Sigma = \{0, 1, \{, \}, [,], :, ,\}$ et l'encodage du code Python de MA351 :

instance	réponse
$\{0 : [1, 11], 1 : [0], 10 : [1], 11 : [0]\}, 10, 11$	
$\{0 : [1, 11], 1 : [0], 10 : [1], 11 : [0]\}, 1, 10$	

Accessibilité

Donnée : Un graphe orienté $\overrightarrow{G}$ donné sous forme de liste d'adjacence et un couple (x, y)

Question : Le sommet y est-il accessible depuis le sommet x dans $\overrightarrow{G}$?

Exemples d'instances avec $\Sigma = \{0, 1, \{, \}, [,], :, ,\}$ et l'encodage du code Python de MA351 :

instance	réponse
$\{0 : [1, 11], 1 : [0], 10 : [1], 11 : [0]\}, 10, 11$	
$\{0 : [1, 11], 1 : [0], 10 : [1], 11 : [0]\}, 1, 10$	

Analyse syntaxique

Donnée : Un texte UTF-8

Question : Est-ce un programme C valide ?

Ce problème sera étudié en détail dans le cours de compilation de 4A, mais des éléments seront présents dans ce cours : langages réguliers, expression régulières et automates à état finis (analyse lexicale), grammaires et langages hors-contexte, automates à pile (analyse syntaxique).

On termine cette liste d'exemples avec trois problèmes **indécidables**.

Analyse syntaxique

Donnée : Un texte UTF-8

Question : Est-ce un programme C valide ?

Ce problème sera étudié en détail dans le cours de compilation de 4A, mais des éléments seront présents dans ce cours : langages réguliers, expression régulières et automates à état finis (analyse lexicale), grammaires et langages hors-contexte, automates à pile (analyse syntaxique).

On termine cette liste d'exemples avec trois problèmes **indécidables**.

Analyse syntaxique

Donnée : Un texte UTF-8

Question : Est-ce un programme C valide ?

Ce problème sera étudié en détail dans le cours de compilation de 4A, mais des éléments seront présents dans ce cours : langages réguliers, expression régulières et automates à état finis (analyse lexicale), grammaires et langages hors-contexte, automates à pile (analyse syntaxique).

On termine cette liste d'exemples avec trois problèmes **indécidables**.

Problème de l'arrêt

Donnée : Un texte UTF-8

Question : Est-ce une fonction Python qui retourne sur toute entrée ?

Exemple d'instance avec le code Python suivant :

```
def collatz(n):
  while n != 1:
    if n%2==0:
      n = n//2
    else:
      n = 3*n+1
  return "Done"
```

se termine-t-il pour tout entier $n \geqslant 1$?

Problème de l'arrêt

Donnée : Un texte UTF-8

Question : Est-ce une fonction Python qui retourne sur toute entrée ?

Exemple d'instance avec le code Python suivant :

```
def collatz(n):
  while n != 1:
    if n%2==0:
      n = n//2
    else:
      n = 3*n+1
  return "Done"
```

se termine-t-il pour tout entier $n \geqslant 1$?

Problème de correspondance de Post (PCP)

Donnée : Une liste de n couples de mots $(u_i, v_i)_{i \in [\![1..n]\!]}$ sur l'alphabet $\{0, 1\}$

Question : Existe-il une suite de k indices $i_1, i_2, \dots, i_k$ telle que les mots $u_{i_1} u_{i_2} \dots u_{i_k}$ et $v_{i_1} v_{i_2} \dots v_{i_k}$ soient égaux ?

Exemple d'instance pour $n = 3$ (chaque couple est un "domino") :

u_i	0101	101	111
v_i	01	011	0111101

Solution : 1,2,3,1. La réponse est oui pour cette instance. Dire que ce problème est indécidable ne signifie pas qu'on ne peut pas résoudre des instances.

Problème de correspondance de Post (PCP)

Donnée : Une liste de n couples de mots $(u_i, v_i)_{i \in [\![1..n]\!]}$ sur l'alphabet $\{0, 1\}$

Question : Existe-il une suite de k indices i_1, i_2, ..., i_k telle que les mots $u_{i_1} u_{i_2} \dots u_{i_k}$ et $v_{i_1} v_{i_2} \dots v_{i_k}$ soient égaux ?

Exemple d'instance pour $n = 3$ (chaque couple est un "domino") :

u_i	0101	101	111
v_i	01	011	0111101

Solution : 1,2,3,1. La réponse est oui pour cette instance. Dire que ce problème est indécidable ne signifie pas qu'on ne peut pas résoudre des instances.

Problème de correspondance de Post (PCP)

Donnée : Une liste de n couples de mots $(u_i, v_i)_{i \in [\![1..n]\!]}$ sur l'alphabet $\{0, 1\}$

Question : Existe-il une suite de k indices i_1, i_2, ..., i_k telle que les mots $u_{i_1} u_{i_2} \dots u_{i_k}$ et $v_{i_1} v_{i_2} \dots v_{i_k}$ soient égaux ?

Exemple d'instance pour $n = 3$ (chaque couple est un "domino") :

u_i	0101	101	111
v_i	01	011	0111101

Solution : 1,2,3,1. La réponse est oui pour cette instance. Dire que ce problème est indécidable ne signifie pas qu'on ne peut pas résoudre des instances.

10e problème de Hilbert

Donnée : Une équation polynomiale multivariée à coefficients entiers

Question : Y a-t-il des racines entières strictement positives ?

instance	réponse
$15x + 21y = 102$	
$x^2 - 2y^2 = 0$	
$x^3 + y^3 - z^3 = 0$	

10ᵉ problème de Hilbert

Donnée : Une équation polynomiale multivariée à coefficients entiers

Question : Y a-t-il des racines entières strictement positives ?

instance	réponse
$15x + 21y = 102$	
$x^2 - 2y^2 = 0$	
$x^3 + y^3 - z^3 = 0$	

Définition : Alphabet, mot sur un alphabet

- Un alphabet Σ est un ensemble fini.
- Un mot w sur Σ est une suite finie $w_1 w_2 \ldots w_n$ d'éléments de Σ, sa longueur est l'entier n et on la note $|w|$.
- Le mot vide, noté ϵ est l'unique mot de longueur 0.
- L'ensemble de tous les mots sur Σ est noté Σ^*.

Définition : Alphabet, mot sur un alphabet

- Un alphabet Σ est un ensemble fini.
- Un mot w sur Σ est une suite finie $w_1 w_2 \dots w_n$ d'éléments de Σ, sa longueur est l'entier n et on la note $|w|$.
- Le mot vide, noté ϵ est l'unique mot de longueur 0.
- L'ensemble de tous les mots sur Σ est noté Σ^*.

Définition : Alphabet, mot sur un alphabet

- Un alphabet Σ est un ensemble fini.
- Un mot w sur Σ est une suite finie $w_1 w_2 \dots w_n$ d'éléments de Σ, sa longueur est l'entier n et on la note $|w|$.
- Le mot vide, noté ϵ est l'unique mot de longueur 0.
- L'ensemble de tous les mots sur Σ est noté Σ^*.

Définition : Alphabet, mot sur un alphabet

- Un alphabet Σ est un ensemble fini.
- Un mot w sur Σ est une suite finie $w_1 w_2 \dots w_n$ d'éléments de Σ, sa longueur est l'entier n et on la note $|w|$.
- Le mot vide, noté ϵ est l'unique mot de longueur 0.
- L'ensemble de tous les mots sur Σ est noté Σ^*.

Exemples

1. Sur l'alphabet $\Sigma = \{0, 1\}$, il y a 4 mots de longueur 2 : 00, 01, 10, 11.
2. Sur l'alphabet $\Sigma = \{a, b, c\}$, on a :

$$\Sigma^* = \{\epsilon, a, b, c, aa, ab, ac, ba, \dots\}$$

l'ensemble de tous les mots peut être énuméré dans l'ordre hiérarchique (par longueur croissante puis par ordre lexicographique).

3. Sur l'alphabet $\Sigma = \{0, 1, \dots, 9, +, \times, (,)\}$:

$$2 + 3 \times (1 + 50) \text{ et } (()$$

sont des mots de longueurs 10 et 3.

Exemples

1. Sur l'alphabet $\Sigma = \{0, 1\}$, il y a 4 mots de longueur 2 : 00, 01, 10, 11.
2. Sur l'alphabet $\Sigma = \{a, b, c\}$, on a :

$$\Sigma^* = \{\epsilon, a,\ b,\ c,\ aa,\ ab,\ ac,\ ba,\ \ldots\}$$

l'ensemble de tous les mots peut être énuméré dans l'ordre hiérarchique (par longueur croissante puis par ordre lexicographique).

3. Sur l'alphabet $\Sigma = \{0, 1, \ldots, 9, +, \times, (,)\}$:

$$2 + 3 \times (1 + 50) \text{ et } (()$$

sont des mots de longueurs 10 et 3.

Exemples

1. Sur l'alphabet $\Sigma = \{0, 1\}$, il y a 4 mots de longueur 2 : 00, 01, 10, 11.
2. Sur l'alphabet $\Sigma = \{a, b, c\}$, on a :

$$\Sigma^* = \{\epsilon, a,\ b,\ c,\ aa,\ ab,\ ac,\ ba,\ \dots\}$$

l'ensemble de tous les mots peut être énuméré dans l'ordre hiérarchique (par longueur croissante puis par ordre lexicographique).

3. Sur l'alphabet $\Sigma = \{0, 1, \dots, 9, +, \times, (,)\}$:

$$2 + 3 \times (1 + 50) \text{ et } (()$$

sont des mots de longueurs 10 et 3.

Définition : produit de concaténation

Si $u = u_1 u_2 \dots u_n$ et $v = v_1 v_2 \dots v_m$ sont deux mots sur Σ alors la concaténation de u et de v, notée uv est le mot $uv = u_1 u_2 \dots u_n v_1 v_2 \dots v_m$.

Exemple

Si $u = aab$ et $v = a$, alors $uv = aaba$ et $vu = aaab$.

Définition : puissance

Soit w un mot sur l'alphabet Σ. Alors $w^0 = \epsilon$ et pour $n \in \mathbb{N}$, $w^{n+1} = ww^n$.

Exemple

$(aab)^3 = aabaabaab$, $aab^3 = aabbb$.

Définition : produit de concaténation

Si $u = u_1 u_2 \ldots u_n$ et $v = v_1 v_2 \ldots v_m$ sont deux mots sur Σ alors la concaténation de u et de v, notée uv est le mot $uv = u_1 u_2 \ldots u_n v_1 v_2 \ldots v_m$.

Exemple

Si $u = aab$ et $v = a$, alors $uv = aaba$ et $vu = aaab$.

Définition : puissance

Soit w un mot sur l'alphabet Σ. Alors $w^0 = \epsilon$ et pour $n \in \mathbb{N}$, $w^{n+1} = ww^n$.

Exemple

$(aab)^3 = aabaabaab$, $aab^3 = aabbb$.

Définition : produit de concaténation

Si $u = u_1 u_2 \dots u_n$ et $v = v_1 v_2 \dots v_m$ sont deux mots sur Σ alors la concaténation de u et de v, notée uv est le mot $uv = u_1 u_2 \dots u_n v_1 v_2 \dots v_m$.

Exemple

Si $u = aab$ et $v = a$, alors $uv = aaba$ et $vu = aaab$.

Définition : puissance

Soit w un mot sur l'alphabet Σ. Alors $w^0 = \epsilon$ et pour $n \in \mathbb{N}$, $w^{n+1} = ww^n$.

Exemple

$(aab)^3 = aabaabaab$, $aab^3 = aabbb$.

Définition : produit de concaténation

Si $u = u_1 u_2 \dots u_n$ et $v = v_1 v_2 \dots v_m$ sont deux mots sur Σ alors la concaténation de u et de v, notée uv est le mot $uv = u_1 u_2 \dots u_n v_1 v_2 \dots v_m$.

Exemple

Si $u = aab$ et $v = a$, alors $uv = aaba$ et $vu = aaab$.

Définition : puissance

Soit w un mot sur l'alphabet Σ. Alors $w^0 = \epsilon$ et pour $n \in \mathbb{N}$, $w^{n+1} = ww^n$.

Exemple

$(aab)^3 = aabaabaab$, $aab^3 = aabbb$.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Définition : langage

Un langage L sur l'alphabet Σ est un ensemble de mots sur Σ. Autrement dit, $L \subset \Sigma^*$, et l'ensemble des langages sur Σ est $\mathcal{P}(\Sigma^*)$.

Exemples

1. $\{a, bab\}$ est un langage fini à deux éléments sur l'alphabet $\Sigma = \{a, b\}$.
2. $\{w \in \Sigma^* \mid |w| \text{ est paire}\}$ est un langage sur $\{a, b\}$, il contient ϵ.
3. $\emptyset$ est le langage vide, $\{\epsilon\}$ le langage ayant pour unique élément le mot vide.
4. L'ensemble des écritures en base 2 des entiers pairs (resp. des nombres premiers) est un langage sur $\{0, 1\}^*$, comportant une infinité de mots.
5. L'ensemble des mots représentant des programmes écrits en C qui s'arrêtent sur toute entrée est un langage.
6. L'ensemble des textes ayant un sens en français est un langage.

Problème de reconnaissance

Soit L un langage sur Σ, alors le problème de reconnaissance (ou de décision) de L est le problème :

Donnée : $w \in \Sigma^*$

Question : A-t-on $w \in L$?

.

Ceci induit une bijection entre les problème dont les instances sont encodées sur Σ et les langages sur Σ, en résumé :

$$\text{Problème} = \text{Langage}$$

Exemple

Si on encode les entiers en base deux par des mots sur $\{0,1\}^*$, le problème de la primalité est associé au langage des mots encodant des nombres premiers.

Problème de reconnaissance

Soit L un langage sur Σ, alors le problème de reconnaissance (ou de décision) de L est le problème :

Donnée : $w \in \Sigma^*$

Question : A-t-on $w \in L$?

.

Ceci induit une bijection entre les problème dont les instances sont encodées sur Σ et les langages sur Σ, en résumé :

$$\text{Problème} = \text{Langage}$$

Exemple

Si on encode les entiers en base deux par des mots sur $\{0,1\}^*$, le problème de la primalité est associé au langage des mots encodant des nombres premiers.

Problème de reconnaissance

Soit L un langage sur Σ, alors le problème de reconnaissance (ou de décision) de L est le problème :

Donnée : $w \in \Sigma^*$

Question : A-t-on $w \in L$?

.

Ceci induit une bijection entre les problème dont les instances sont encodées sur Σ et les langages sur Σ, en résumé :

$$\text{Problème} = \text{Langage}$$

Exemple

Si on encode les entiers en base deux par des mots sur $\{0,1\}^*$, le problème de la primalité est associé au langage des mots encodant des nombres premiers.

Problème de reconnaissance

Soit L un langage sur Σ, alors le problème de reconnaissance (ou de décision) de L est le problème :

Donnée : $w \in \Sigma^*$

Question : A-t-on $w \in L$?

.

Ceci induit une bijection entre les problème dont les instances sont encodées sur Σ et les langages sur Σ, en résumé :

$$\text{Problème} = \text{Langage}$$

Exemple

Si on encode les entiers en base deux par des mots sur $\{0,1\}^*$, le problème de la primalité est associé au langage des mots encodant des nombres premiers.

Définition : opérations ensemblistes

Sur un alphabet Σ, l'ensemble des langages $\mathcal{P}(\Sigma^*)$ est muni des opérations ensemblistes bien connues :

- $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ ou } w \in L_2\}$.
- $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ et } w \in L_2\}$.
- $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$.

Définition : opérations ensemblistes

Sur un alphabet Σ, l'ensemble des langages $\mathcal{P}(\Sigma^*)$ est muni des opérations ensemblistes bien connues :

- $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ ou } w \in L_2\}$.
- $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ et } w \in L_2\}$.
- $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$.

Définition : opérations ensemblistes

Sur un alphabet Σ, l'ensemble des langages $\mathcal{P}(\Sigma^*)$ est muni des opérations ensemblistes bien connues :

- $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ ou } w \in L_2\}$.
- $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ et } w \in L_2\}$.
- $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$.

Définition : opérations ensemblistes

Sur un alphabet Σ, l'ensemble des langages $\mathcal{P}(\Sigma^*)$ est muni des opérations ensemblistes bien connues :

- $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ ou } w \in L_2\}$.
- $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ et } w \in L_2\}$.
- $\overline{L} = \{w \in \Sigma^* \mid w \notin L\}$.

Définition : opérations liées à la concaténation

Soit Σ un alphabet. Le produit de deux langages L_1 et L_2 sur Σ est :

$$L_1L_2 = \{w \in \Sigma^* \mid w = w_1w_2 \text{ avec } w_1 \in L_1 \text{ et } w_2 \in L_2\}$$

Soit L un langage sur Σ, pour $n \in N$ et on définit L^n par :

Base : $L^0 = \{\epsilon\}$.

Induction : Pour $n \in \mathbb{N}$, $L^{n+1} = LL^n$.

La fermeture de Kleen de L, notée L^* est le langage :

$$L^* = \bigcup_{n \geqslant 0} L^n,$$

enfin on note $L^+ = \bigcup_{n \geqslant 1} L^n$.

Définition : opérations liées à la concaténation

Soit Σ un alphabet. Le produit de deux langages L_1 et L_2 sur Σ est :

$$L_1L_2 = \{w \in \Sigma^* \mid w = w_1w_2 \text{ avec } w_1 \in L_1 \text{ et } w_2 \in L_2\}$$

Soit L un langage sur Σ, pour $n \in N$ et on définit L^n par :

Base : $L^0 = \{\epsilon\}$.

Induction : Pour $n \in \mathbb{N}$, $L^{n+1} = LL^n$.

La fermeture de Kleen de L, notée L^* est le langage :

$$L^* = \bigcup_{n \geqslant 0} L^n,$$

enfin on note $L^+ = \bigcup_{n \geqslant 1} L^n$.

Définition : opérations liées à la concaténation

Soit Σ un alphabet. Le produit de deux langages L_1 et L_2 sur Σ est :

$$L_1L_2 = \{w \in \Sigma^* \mid w = w_1w_2 \text{ avec } w_1 \in L_1 \text{ et } w_2 \in L_2\}$$

Soit L un langage sur Σ, pour $n \in N$ et on définit L^n par :

Base : $L^0 = \{\epsilon\}$.

Induction : Pour $n \in \mathbb{N}$, $L^{n+1} = LL^n$.

La fermeture de Kleen de L, notée L^* est le langage :

$$L^* = \bigcup_{n \geqslant 0} L^n,$$

enfin on note $L^+ = \bigcup_{n \geqslant 1} L^n$.

Définition : opérations liées à la concaténation

Soit Σ un alphabet. Le produit de deux langages L_1 et L_2 sur Σ est :

$$L_1L_2 = \{w \in \Sigma^* \mid w = w_1w_2 \text{ avec } w_1 \in L_1 \text{ et } w_2 \in L_2\}$$

Soit L un langage sur Σ, pour $n \in N$ et on définit L^n par :

Base : $L^0 = \{\epsilon\}$.

Induction : Pour $n \in \mathbb{N}$, $L^{n+1} = LL^n$.

La fermeture de Kleen de L, notée L^* est le langage :

$$L^* = \bigcup_{n \geqslant 0} L^n,$$

enfin on note $L^+ = \bigcup_{n \geqslant 1} L^n$.

Définition : opérations liées à la concaténation

Soit Σ un alphabet. Le produit de deux langages L_1 et L_2 sur Σ est :

$$L_1 L_2 = \{w \in \Sigma^* \mid w = w_1 w_2 \text{ avec } w_1 \in L_1 \text{ et } w_2 \in L_2\}$$

Soit L un langage sur Σ, pour $n \in N$ et on définit L^n par :

Base : $L^0 = \{\epsilon\}$.

Induction : Pour $n \in \mathbb{N}$, $L^{n+1} = LL^n$.

La fermeture de Kleen de L, notée L^* est le langage :

$$L^* = \bigcup_{n \geqslant 0} L^n,$$

enfin on note $L^+ = \bigcup_{n \geqslant 1} L^n$.

Exemple

$$\{a, aa\}^2 = \{a, a^2\}^2 = \{a^2, a^3, a^4\} = \{aa, aaa, aaaa\}$$

Exercices

1. Soit $L_1 = \{\varepsilon, b\}$ et $L_2 = \{a, ba\}$.
   Déterminer L_1L_2, L_2L_1, L_1^2, L_2^2, L_1^* et $L_1 \cup L_2$.
2. Montrer que pour tout langage L, il y a équivalence entre :
   - $\varepsilon \in L$
   - $L \subset L^2$

Exemple

$$\{a, aa\}^2 = \{a, a^2\}^2 = \{a^2, a^3, a^4\} = \{aa, aaa, aaaa\}$$

Exercices

1. Soit $L_1 = \{\varepsilon, b\}$ et $L_2 = \{a, ba\}$.
   Déterminer L_1L_2, L_2L_1, L_1^2, L_2^2, L_1^* et $L_1 \cup L_2$.
2. Montrer que pour tout langage L, il y a équivalence entre :
   - $\varepsilon \in L$
   - $L \subset L^2$

Définition : langages réguliers

Soit Σ un alphabet. L'ensemble $\mathcal{R}$ des langages régulier sur Σ est définit inductivement :

Base : $\emptyset$, $\{\epsilon\}$, et $\{a\}$ pour tout $a \in \Sigma$ sont des langages réguliers.

Induction :
- Si L est régulier, L^* est régulier.
- Si L_1 et L_2 sont régulier, $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, $L_1 L_2$ est régulier.

Exemples

Les langages

$$(\{a\}^* \{b\} \cup \{b\}^*)^*, \quad \emptyset^* \cup \{a\} \cup \{b\}, \quad (\{a\} \cup \{b\}^*)^*$$

sont réguliers.

Définition : langages réguliers

Soit Σ un alphabet. L'ensemble $\mathcal{R}$ des langages régulier sur Σ est définit inductivement :

Base : $\emptyset$, $\{\epsilon\}$, et $\{a\}$ pour tout $a \in \Sigma$ sont des langages réguliers.

Induction :
- Si L est régulier, L^* est régulier.
- Si L_1 et L_2 sont régulier, $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, $L_1 L_2$ est régulier.

Exemples

Les langages

$$(\{a\}^* \{b\} \cup \{b\}^*)^*, \quad \emptyset^* \cup \{a\} \cup \{b\}, \quad (\{a\} \cup \{b\}^*)^*$$

sont réguliers.

Définition : langages réguliers

Soit Σ un alphabet. L'ensemble $\mathcal{R}$ des langages régulier sur Σ est définit inductivement :

Base : $\emptyset$, $\{\epsilon\}$, et $\{a\}$ pour tout $a \in \Sigma$ sont des langages réguliers.

Induction :
- Si L est régulier, L^* est régulier.
- Si L_1 et L_2 sont régulier, $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, $L_1 L_2$ est régulier.

Exemples

Les langages

$$(\{a\}^* \{b\} \cup \{b\}^*)^*, \quad \emptyset^* \cup \{a\} \cup \{b\}, \quad (\{a\} \cup \{b\}^*)^*$$

sont réguliers.

Définition : langages réguliers

Soit Σ un alphabet. L'ensemble $\mathcal{R}$ des langages régulier sur Σ est définit inductivement :

Base : $\emptyset$, $\{\epsilon\}$, et $\{a\}$ pour tout $a \in \Sigma$ sont des langages réguliers.

Induction :
- Si L est régulier, L^* est régulier.
- Si L_1 et L_2 sont régulier, $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, $L_1 L_2$ est régulier.

Exemples

Les langages

$$(\{a\}^* \{b\} \cup \{b\}^*)^*, \quad \emptyset^* \cup \{a\} \cup \{b\}, \quad (\{a\} \cup \{b\}^*)^*$$

sont réguliers.

Définition : langages réguliers

Soit Σ un alphabet. L'ensemble $\mathcal{R}$ des langages régulier sur Σ est définit inductivement :

Base : $\emptyset$, $\{\epsilon\}$, et $\{a\}$ pour tout $a \in \Sigma$ sont des langages réguliers.

Induction :
- Si L est régulier, L^* est régulier.
- Si L_1 et L_2 sont régulier, $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, $L_1 L_2$ est régulier.

Exemples

Les langages

$$(\{a\}^* \{b\} \cup \{b\}^*)^*, \quad \emptyset^* \cup \{a\} \cup \{b\}, \quad (\{a\} \cup \{b\}^*)^*$$

sont réguliers.

Définition : langages réguliers

Soit Σ un alphabet. L'ensemble $\mathcal{R}$ des langages régulier sur Σ est définit inductivement :

Base : $\emptyset$, $\{\epsilon\}$, et $\{a\}$ pour tout $a \in \Sigma$ sont des langages réguliers.

Induction :
- Si L est régulier, L^* est régulier.
- Si L_1 et L_2 sont régulier, $L_1 \cup L_2$ est régulier.
- Si L_1 et L_2 sont réguliers, $L_1 L_2$ est régulier.

Exemples

Les langages

$$(\{a\}^* \{b\} \cup \{b\}^*)^*, \quad \emptyset^* \cup \{a\} \cup \{b\}, \quad (\{a\} \cup \{b\}^*)^*$$

sont réguliers.

Définition : expression régulière

Soit Σ un alphabet. Une expression régulière E est un mot sur $\Sigma \cup \{^*, +, (,)\}$ dénotant un langage régulier $L(E)$ selon la définition inductive suivante :

base : $\emptyset$, ϵ et a pour $a \in \Sigma$ sont des expressions régulières dénotant les langages $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ et pour $a \in \Sigma$, $L(a) = \{a\}$.

Induction :
- Si E est une expression régulière, alors (E^*) est une expression régulière dénotant le langage $L((E^*)) = L(E)^*$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 + E_2)$ est une expression régulière dénotant le langage $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 E_2)$ est une expression régulière dénotant le langage $L((E_1 E_2)) = L(E_1)L(E_2)$.

Définition : expression régulière

Soit Σ un alphabet. Une expression régulière E est un mot sur $\Sigma \cup \{^*, +, (,)\}$ dénotant un langage régulier $L(E)$ selon la définition inductive suivante :

base : $\emptyset$, ϵ et a pour $a \in \Sigma$ sont des expressions régulières dénotant les langages $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ et pour $a \in \Sigma$, $L(a) = \{a\}$.

Induction :
- Si E est une expression régulière, alors (E^*) est une expression régulière dénotant le langage $L((E^*)) = L(E)^*$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 + E_2)$ est une expression régulière dénotant le langage $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 E_2)$ est une expression régulière dénotant le langage $L((E_1 E_2)) = L(E_1) L(E_2)$.

Définition : expression régulière

Soit Σ un alphabet. Une expression régulière E est un mot sur $\Sigma \cup \{^*, +, (,)\}$ dénotant un langage régulier $L(E)$ selon la définition inductive suivante :

base : $\emptyset$, ϵ et a pour $a \in \Sigma$ sont des expressions régulières dénotant les langages $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ et pour $a \in \Sigma$, $L(a) = \{a\}$.

Induction :

- Si E est une expression régulière, alors (E^*) est une expression régulière dénotant le langage $L((E^*)) = L(E)^*$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 + E_2)$ est une expression régulière dénotant le langage $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 E_2)$ est une expression régulière dénotant le langage $L((E_1 E_2)) = L(E_1)L(E_2)$.

Définition : expression régulière

Soit Σ un alphabet. Une expression régulière E est un mot sur $\Sigma \cup \{^*, +, (,)\}$ dénotant un langage régulier $L(E)$ selon la définition inductive suivante :

base : $\emptyset$, ϵ et a pour $a \in \Sigma$ sont des expressions régulières dénotant les langages $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ et pour $a \in \Sigma$, $L(a) = \{a\}$.

Induction :

- Si E est une expression régulière, alors (E^*) est une expression régulière dénotant le langage $L((E^*)) = L(E)^*$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 + E_2)$ est une expression régulière dénotant le langage $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 E_2)$ est une expression régulière dénotant le langage $L((E_1 E_2)) = L(E_1)L(E_2)$.

Définition : expression régulière

Soit Σ un alphabet. Une expression régulière E est un mot sur $\Sigma \cup \{^*, +, (,)\}$ dénotant un langage régulier $L(E)$ selon la définition inductive suivante :

base : $\emptyset$, ϵ et a pour $a \in \Sigma$ sont des expressions régulières dénotant les langages $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ et pour $a \in \Sigma$, $L(a) = \{a\}$.

Induction :

- Si E est une expression régulière, alors (E^*) est une expression régulière dénotant le langage $L((E^*)) = L(E)^*$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 + E_2)$ est une expression régulière dénotant le langage $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 E_2)$ est une expression régulière dénotant le langage $L((E_1 E_2)) = L(E_1)L(E_2)$.

Définition : expression régulière

Soit Σ un alphabet. Une expression régulière E est un mot sur $\Sigma \cup \{^*, +, (,)\}$ dénotant un langage régulier $L(E)$ selon la définition inductive suivante :

base : $\emptyset$, ϵ et a pour $a \in \Sigma$ sont des expressions régulières dénotant les langages $L(\emptyset) = \emptyset$, $L(\epsilon) = \{\epsilon\}$ et pour $a \in \Sigma$, $L(a) = \{a\}$.

Induction :

- Si E est une expression régulière, alors (E^*) est une expression régulière dénotant le langage $L((E^*)) = L(E)^*$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 + E_2)$ est une expression régulière dénotant le langage $L((E_1 + E_2)) = L(E_1) \cup L(E_2)$.
- Si E_1 et E_2 sont des expressions régulières, alors $(E_1 E_2)$ est une expression régulière dénotant le langage $L((E_1 E_2)) = L(E_1)L(E_2)$.

Exemples

Les trois langages de l'exemple précédent sont avantageusement dénotés par les expressions régulières (une fois supprimées les parenthèses inutiles) :

$$(a^*b + b^*)^*, \quad \emptyset^* + a + b, \quad (a + b^*)^*$$

Pour utiliser un minimum de parenthèses, on convient des priorités suivantes :

1. Les étoiles et les puissances sont évaluées en premier.
2. Les concaténation sont évaluées en 2e, et l'associativité dispense d'écrire des parenthèses : $(ab)c$ et $a(bc)$ correspondent au même langage régulier et sont écrites abc.
3. Les unions $(+)$ sont évaluées en 3e, à nouveau en se dispensant de parenthèses étant donnée l'associativité.

On observe essentiellement la disparition des accolades pour les singletons et le remplacement de $\cup$ par $+$.

Exemples

Les trois langages de l'exemple précédent sont avantageusement dénotés par les expressions régulières (une fois supprimées les parenthèses inutiles) :

$$(a^*b + b^*)^*, \quad \emptyset^* + a + b, \quad (a + b^*)^*$$

Pour utiliser un minimum de parenthèses, on convient des priorités suivantes :

1. Les étoiles et les puissances sont évaluées en premier.
2. Les concaténation sont évaluées en 2e, et l'associativité dispense d'écrire des parenthèses : $(ab)c$ et $a(bc)$ correspondent au même langage régulier et sont écrites abc.
3. Les unions $(+)$ sont évaluées en 3e, à nouveau en se dispensant de parenthèses étant donnée l'associativité.

On observe essentiellement la disparition des accolades pour les singletons et le remplacement de $\cup$ par $+$.

Exemples

Les trois langages de l'exemple précédent sont avantageusement dénotés par les expressions régulières (une fois supprimées les parenthèses inutiles) :

$$(a^* b + b^*)^*, \quad \emptyset^* + a + b, \quad (a + b^*)^*$$

Pour utiliser un minimum de parenthèses, on convient des priorités suivantes :

1. Les étoiles et les puissances sont évaluées en premier.
2. Les concaténation sont évaluées en 2e, et l'associativité dispense d'écrire des parenthèses : $(ab)c$ et $a(bc)$ correspondent au même langage régulier et sont écrites abc.
3. Les unions $(+)$ sont évaluées en 3e, à nouveau en se dispensant de parenthèses étant donnée l'associativité.

On observe essentiellement la disparition des accolades pour les singletons et le remplacement de $\cup$ par $+$.

Exemples

Les trois langages de l'exemple précédent sont avantageusement dénotés par les expressions régulières (une fois supprimées les parenthèses inutiles) :

$$(a^* b + b^*)^*, \quad \emptyset^* + a + b, \quad (a + b^*)^*$$

Pour utiliser un minimum de parenthèses, on convient des priorités suivantes :

1. Les étoiles et les puissances sont évaluées en premier.
2. Les concaténation sont évaluées en 2e, et l'associativité dispense d'écrire des parenthèses : $(ab)c$ et $a(bc)$ correspondent au même langage régulier et sont écrites abc.
3. Les unions $(+)$ sont évaluées en 3e, à nouveau en se dispensant de parenthèses étant donnée l'associativité.

On observe essentiellement la disparition des accolades pour les singletons et le remplacement de $\cup$ par $+$.

Exemples

Les trois langages de l'exemple précédent sont avantageusement dénotés par les expressions régulières (une fois supprimées les parenthèses inutiles) :

$$(a^* b + b^*)^*, \quad \emptyset^* + a + b, \quad (a + b^*)^*$$

Pour utiliser un minimum de parenthèses, on convient des priorités suivantes :

1. Les étoiles et les puissances sont évaluées en premier.
2. Les concaténation sont évaluées en 2e, et l'associativité dispense d'écrire des parenthèses : $(ab)c$ et $a(bc)$ correspondent au même langage régulier et sont écrites abc.
3. Les unions $(+)$ sont évaluées en 3e, à nouveau en se dispensant de parenthèses étant donnée l'associativité.

On observe essentiellement la disparition des accolades pour les singletons et le remplacement de $\cup$ par $+$.

Exemples

Les trois langages de l'exemple précédent sont avantageusement dénotés par les expressions régulières (une fois supprimées les parenthèses inutiles) :

$$(a^* b + b^*)^*, \quad \emptyset^* + a + b, \quad (a + b^*)^*$$

Pour utiliser un minimum de parenthèses, on convient des priorités suivantes :

1. Les étoiles et les puissances sont évaluées en premier.
2. Les concaténation sont évaluées en 2e, et l'associativité dispense d'écrire des parenthèses : $(ab)c$ et $a(bc)$ correspondent au même langage régulier et sont écrites abc.
3. Les unions $(+)$ sont évaluées en 3e, à nouveau en se dispensant de parenthèses étant donnée l'associativité.

On observe essentiellement la disparition des accolades pour les singletons et le remplacement de $\cup$ par $+$.

Théorème

Un langage est régulier si et seulement si il est dénoté par une expression régulière

Remarque

L'application L de l'ensemble des expressions régulières bien formées dans l'ensemble des langages réguliers est surjective par construction, mais elle n'a pas de raison d'être injective. Et elle ne l'est pas, par exemple :

$$L((0+1)^*) = L((0^*1^*)^*) = \{0,1\}^*$$

Théorème

Un langage est régulier si et seulement si il est dénoté par une expression régulière

Remarque

L'application L de l'ensemble des expressions régulières bien formées dans l'ensemble des langages réguliers est surjective par construction, mais elle n'a pas de raison d'être injective. Et elle ne l'est pas, par exemple :

$$L((0+1)^*) = L((0^*1^*)^*) = \{0,1\}^*$$

Théorème

Un langage est régulier si et seulement si il est dénoté par une expression régulière

Remarque

L'application L de l'ensemble des expressions régulières bien formées dans l'ensemble des langages réguliers est surjective par construction, mais elle n'a pas de raison d'être injective. Et elle ne l'est pas, par exemple :

$$L((0+1)^*) = L((0^*1^*)^*) = \{0,1\}^*$$

Théorème

Un langage est régulier si et seulement si il est dénoté par une expression régulière

Remarque

L'application L de l'ensemble des expressions régulières bien formées dans l'ensemble des langages réguliers est surjective par construction, mais elle n'a pas de raison d'être injective. Et elle ne l'est pas, par exemple :

$$L((0+1)^*) = L((0^*1^*)^*) = \{0,1\}^*$$

Exercice

1. Montrer que $(a^*b)^* + (b^*a)^* = (a+b)^*$.
2. Montrer que pour tout mot u et v, il y a équivalence entre :
   (a) u et v commutent
   (b) Il existe un mot w et $(p,q) \in \mathbb{N}^2$ tels que $u = w^p$ et $v = w^q$.
3. Soit R et S deux expressions régulières définies comme suit :

$$R = a(a+b)^*ba$$

$$S = (ab)^* + (ba)^* + (a^* + b^*)$$

   a) Déterminer un mot appartenant au langage dénoté par R, mais pas au langage dénoté par S.
   b) Déterminer un mot appartenant au langage dénoté par S, mais pas au langage dénoté par R.
   c) Déterminer un mot appartenant au langage dénoté par R et au langage dénoté par S.
   d) Déterminer un mot qui n'appartient ni au langage dénoté par R ni au langage dénoté par S.

Exercice

1. Montrer que $(a^*b)^* + (b^*a)^* = (a+b)^*$.
2. Montrer que pour tout mot u et v, il y a équivalence entre :
   (a) u et v commutent
   (b) Il existe un mot w et $(p, q) \in \mathbb{N}^2$ tels que $u = w^p$ et $v = w^q$.
3. Soit R et S deux expressions régulières définies comme suit :

$$R = a(a+b)^*ba$$

$$S = (ab)^* + (ba)^* + (a^* + b^*)$$

   a) Déterminer un mot appartenant au langage dénoté par R, mais pas au langage dénoté par S.
   b) Déterminer un mot appartenant au langage dénoté par S, mais pas au langage dénoté par R.
   c) Déterminer un mot appartenant au langage dénoté par R et au langage dénoté par S.
   d) Déterminer un mot qui n'appartient ni au langage dénoté par R ni au langage dénoté par S.

Exercice

1. Montrer que $(a^*b)^* + (b^*a)^* = (a+b)^*$.
2. Montrer que pour tout mot u et v, il y a équivalence entre :
   (a) u et v commutent
   (b) Il existe un mot w et $(p,q) \in \mathbb{N}^2$ tels que $u = w^p$ et $v = w^q$.
3. Soit R et S deux expressions régulières définies comme suit :

$$R = a(a+b)^*ba$$

$$S = (ab)^* + (ba)^* + (a^* + b^*)$$

   a) Déterminer un mot appartenant au langage dénoté par R, mais pas au langage dénoté par S.
   b) Déterminer un mot appartenant au langage dénoté par S, mais pas au langage dénoté par R.
   c) Déterminer un mot appartenant au langage dénoté par R et au langage dénoté par S.
   d) Déterminer un mot qui n'appartient ni au langage dénoté par R ni au langage dénoté par S.

Langage non régulier

Il existe des langages qui ne sont pas réguliers sur un alphabet donné.

- L'ensemble des langages réguliers est un ensemble dénombrable sur un alphabet donné : on peut énumérer les expressions régulières dans l'ordre hiérarchique, cela induit une surjection de $\mathbb{N}$ dans l'ensemble des langages réguliers.
- L'ensemble des langages est l'ensemble des parties de Σ^*. Il n'est pas dénombrable.