Cours Théorie des langages de programmation

Filière Génie logiciel & Digitalisation

Professeur MOUSSAID LAILA

Année universitaire 2023 -2024

Table de matières

1. Introduction

Ce travail s'inscrit dans un cadre académique visant à explorer les concepts fondamentaux de la théorie des langages, tout en reliant ces notions à leurs applications concrètes. Le but de ce document est de présenter une vision claire et structurée des principaux éléments de cette discipline, en couvrant à la fois les aspects théoriques et pratiques.

La théorie des langages constitue une pierre angulaire de l'informatique théorique. Elle se concentre sur la manière dont les langages, en tant que systèmes de symboles, peuvent être définis, manipulés et interprétés. Cette discipline, issue des travaux pionniers de chercheurs comme Noam Chomsky, Alan Turing et Stephen Kleene, a permis de formaliser les bases de la communication entre les humains et les machines. Elle est devenue indispensable pour de nombreux domaines, tels que le développement des compilateurs, la vérification des logiciels, l'analyse syntaxique, ou encore les systèmes de reconnaissance vocale.

Les **notions fondamentales** abordées dans les premiers chapitres établissent une base solide pour comprendre les concepts clés, comme les alphabets, les mots et les langages, qui servent de fondation pour des sujets plus complexes. Ces notions permettent d'introduire des opérations sur les langages, telles que l'union, la concaténation ou l'étoile de Kleene, qui sont au cœur de la manipulation des langages.

Les **grammaires formelles** constituent une autre thématique majeure de ce document. Ces dernières permettent de décrire, de manière concise et rigoureuse, la structure des langages. La classification de Chomsky, qui divise les grammaires en quatre types distincts, fournit une hiérarchie précieuse pour évaluer la puissance et les limites des langages en termes de complexité et d'expressivité.

Le lien entre les **grammaires et les langages** est ensuite exploré afin de mettre en lumière les relations qui existent entre ces deux concepts fondamentaux. Ces relations jouent un rôle essentiel dans la conception des langages de programmation modernes et des outils associés, tels que les compilateurs.

Par ailleurs, les **langages réguliers** occupent une place particulière dans ce document en raison de leur simplicité et de leur pertinence pratique. Leur définition, leurs propriétés et leur représentation à travers les expressions régulières sont étudiées en détail, ouvrant la voie à l'analyse des **automates à états finis**, des structures mathématiques utilisées pour reconnaître ces langages.

L'une des sections clés du document est consacrée à l'**équivalence entre les automates finis et les langages réguliers**, qui illustre la puissance des automates en tant qu'outil pour modéliser et traiter les langages. Cette correspondance, à la fois théorique et pratique, constitue un pont entre les concepts abstraits de la théorie des langages et les applications réelles dans le développement de systèmes informatiques.

Au-delà des concepts théoriques, ce document met également en évidence l'importance des langages formels dans des applications modernes. Par exemple, les expressions régulières sont largement utilisées dans les moteurs de recherche et les outils de validation syntaxique, tandis que les automates et les grammaires servent de base aux outils d'analyse de code et à la conception des langages de programmation. Les avancées dans la théorie des langages ont également trouvé des applications dans des domaines variés tels que l'intelligence artificielle, les bases de données, et même la bio-informatique, où les systèmes de reconnaissance des motifs s'inspirent directement de ces concepts.

En conclusion, ce document vise à fournir une introduction approfondie et pédagogique à la théorie des langages, tout en soulignant son importance pour la compréhension et la résolution des problèmes informatiques. En combinant rigueur théorique et illustrations pratiques, il offre aux lecteurs une opportunité de plonger dans une discipline fascinante et d'acquérir les bases nécessaires pour explorer des sujets plus avancés. Ce travail s'adresse à la fois aux étudiants, aux chercheurs et aux professionnels désireux de mieux comprendre les fondements de l'informatique théorique et leurs implications dans la résolution des défis contemporains.

2. Notions fondamentales de la théorie des langages

2.1.Définitions

2.1.1. Symbole

Les symboles sont des éléments indivisibles qui vont servir de briques de base pour construire des mots.

Exemple :

– les chiffres 0,1,...,9 sont des symboles

– les lettres latine ,Arabe,Tamazight …….

2.1.2. Alphabet

Un alphabet est un ensemble fini de symboles. (vocabulaire :∑)

Exemple :{0,1,2,3,4,5,6,7,8,9}

{A,B,C,D,E,F,G,H,I,J,K,L,M}

2.1.3. Mot

Une suite de symboles, appartenant à un alphabet ∑ , mis bout à bout est appelé un mot sur ∑ .

Remarque :

- Le nombre de symboles entrant dans la composition d'un mot est appelé la longueur de a, que l'on note à l'aide de deux barres verticales : |a|.
- le mot vide (ε) ,sa longueur vaut zéro.
- La concaténation de deux mots a et b , notée a·b ou simplement ab est le mot obtenu en juxtaposant les symboles de b à la suite de ceux de a .
- L'ensemble de tous les mots que l'on peut construire sur un alphabet ∑, ε inclu, est noté ∑ *.

Exemple : Salam ,Digitalization,Then,If

- Propriété de la concaténation

Soient w, w1 et w2 trois mots définis sur l'alphabet ∑ :

– |w1 .w2 | = |w1 | + |w2 |

– $\forall a \in \sum : |w1 .w2 |a = |w1 |a + |w2 |a$
– $(w1 .w2).w3 = w1 .(w2 .w3)$

- – $w.\varepsilon = \varepsilon.w = w$
- L'exposant

L'opération w.w est notée par w^2 . En généralisant, on note w^n = w...w . En particulier, n fois

$w^0 = \varepsilon$.

- Le mot miroir

Soit $w = a^1 a^2 ...a^n$ un mot sur $\sum$. On appelle mot miroir de w et on le note par w^R et on l'écrit w l'envers, c'est-à-dire que $w^R = a^n ...a^2 a^1$.

Les mots qui sont égaux à leur miroir, appelés palindromes,.

- Préfixe et suffixe

Soit w un mot défini sur un alphabet $\sum$. Un mot x (resp. y) formé sur $\sum$ est un préfixe (resp. suffixe) de w s'il existe un mot u formé sur $\sum$ (resp. v formé sur $\sum$) tel que w = xu (resp.w = vy). Si $w = a^1 a^2 ...a^n$ alors tous les mots de l'ensemble $\{\varepsilon, a^1 , a^1 a^2 , a^1 a^2 a^3 , ..., a^1 a^2 ...a^n \}$

sont des préfixes de w. De même, tous les mots de l'ensemble $\{\varepsilon, a^n , a^{n-1} a^n , a^{n-2} a^{n-1} a^n , ...,a^1 a^2 ...a^n \}$ sont des suffixes de w.

- Mots conjugués

Deux mots x et y sont dits conjugués s'il existe deux mots u et v tels que : x = uv et y = vu.

3. Langages

3.1.Définition

Un langage, défini sur un alphabet $\sum$, est un ensemble de mots définis sur $\sum$. Autrement dit, un langage est un sous-ensemble de $\sum$ *.

Remarque :

- Deux langages particuliers sont indépendants de l'alphabet $\sum$:

- le langage vide (L = Ø),

- le langage contenant le seul mot vide (L = {ε}).

Exemple : Langage des mots de longueur 2 défini sur l'alphabet {0, 1}est {00, 01, 10, 11} ;

4. Opérations sur les langages

Pour les langages, plusieurs opérations ensemblistes peuvent être appliquées. Soient L, L^1 et L^2 trois langages définis sur l'alphabet ∑, on définit les opérations suivantes **:**

4.1.Union

L'union de deux langages L^1 et L^2, est le langage, noté:

$L^1 \cup L^2$ constitué des mots appartenant à L^1 ou à L^2.

$$L^1 \cup L^2 = \{x | x \in L^1 \text{ ou } x \in L^2\}$$

4.2.Intersection

L'intersection de L^1 et L^2, est le langage, noté : $L^1 \cap L^2$ constitué des mots appartenant à L^1 et à L^2.

$$L^1 \cap L^2 = \{x | x \in L^1 \text{ et } x \in L^2\}$$

4.3. Différence

La différence de L^1 et L^2 est le langage, noté $L^1 - L^2$, constitué des mots appartenant à L^1 et n'appartenant pas à L^2.

$$L^1 - L^2 = \{x | x \in L^1 \text{ et } x /\in L^2\}$$

4.4.Concaténation

La concaténation de deux langages L^1 et L^2 est le langage noté $L^1 L^2$ composé des mots xy tels que $x \in L^1$ et $y \in L^2$.

$$L^1 L^2 = \{xy / x \in L^1 \text{ et } y \in L^2\}$$

4.5.Fermeture de Kleene

on note L^n la concaténation de L avec lui-même n fois :

On définit enfin la fermeture de Kleene du langage L, notée L^* de la façon suivante :

$$L^* = \cup L^k$$
$$k \geq 0$$

5. Grammaires

5.1.Définition

Une grammaire est un quadruplet G = (T,N, S,R) tel que :

– T est le vocabulaire terminal ou l'alphabet sur lequel est défini le langage.

– N est le vocabulaire non terminal, c'est-à-dire l'ensemble des symboles qui n'apparaissent pas dans les mots générés, mais qui sont utilisés au cours de la génération.

Un symbole non terminal désigne une "catégorie syntaxique".

– R est un ensemble de règles dites de réécriture ou de production de la forme :

$$u^1 \longrightarrow u^2, \text{ avec } u^1 \in (NUT)^+ \text{ et } u^2 \in (NUT)^*$$

La signification intuitive de ces règles est que la suite non vide de symboles terminaux ou non terminaux u^1 peut être remplacée par la suite éventuellement vide de symboles terminaux ou non terminaux u^2.

– S ∈ N est le symbole de départ ou axiome. C'est à partir de ce symbole non terminal que l'on commencera la génération de mots au moyen des règles de la grammaire.

5.2.Terminologie :

- une suite de symboles terminaux et non terminaux $(N \cup T)^*$ est appelée une forme.
- une règle $u1 \longrightarrow u$ telle que u ∈ T est appelée une règle terminale.
- Lorsque plusieurs règles de grammaire ont une même forme en partie gauche :

Remarque:

S ⟶ ab,

S ⟶ aSb, pourra s'écrire S ⟶ ab|aSb|c.

S ⟶ c

5.3.Définition (Dérivation en une étape)

Soient une grammaire G = (T,N,S,R), une forme non vide u ∈ $(NUT)^+$ et une forme éventuellement vide v ∈(NUT)*. La grammaire G permet de dériver v de u en une étape (noté u ⇒ v) si et seulement si :

- u = x ù y (u peut être décomposé en x, ù et y ; x et y peuvent être vides),
- v = x v̀ y (v peut être décomposé en x, v̀ et y),
- ù ⟶ v̀ est une règle de R.

5.4. Définition (Dérivation en plusieurs étapes)

Une forme v peut être dérivée d'une forme u en plusieurs étapes :

- u ⇒ v : si v peut être obtenue de u par une succession de 1 ou plusieurs dérivations en une étape.
- u ⇒* v : si v peut être obtenue de u par une succession de 0, 1 ou plusieurs dérivations en une étape.

6. Relations entre les Langages et les Grammaires

6.1.Langage généré par une grammaire

Définition

Le langage généré par une grammaire G = (T,N,S,R) est l'ensemble des mots sur T qui peuvent être dérivés à partir de S :

$$L(G) = \{v \in T^* / S \Rightarrow^* v\}$$

Remarque:

- Une grammaire définit un seul langage.
- un même langage peut être engendré par plusieurs grammaires différentes.
- Deux grammaires G et G′ sont équivalentes si les langages L(G) et L(G′) sont identiques.

7. Grammaire régulière

7.1.Définition1

une grammaire G = (T,N, S,R) est régulière

- à droite si les règles de R sont de la forme:

$A \longrightarrow aB$ ou $A \longrightarrow a$ avec$A,B \in N$ et $a \in T$

Exemple : soit la grammaire régulière à droite : $G1 = (T^1,N^1, S^1,R^1)$ avec

$T^1 = \{ a, b \}$

$N^1 = \{S^1,U^1 \}$

$R^1 = \{S^1 \longrightarrow aS^1 | aU^1$

$U^1 \longrightarrow bU^1 | b \}$

7.2.Définition2

Une grammaire G = (T,N, S,R) est régulière

– à gauche si les règles de R sont de la forme

$A \longrightarrow Ba$ ou $A \longrightarrow a$ avec$A,B \in N$ et $a \in T$

Exemple : soit la grammaire régulière à gauche : $G2 = (T^2,N^2, S^2,R^2)$ avec

$T^2 = \{ a, b \}$

$N^2 = \{S^2,U^2 \}$

$R^2 = \{S^2 \longrightarrow S^2b | U^2b$

$U^2 \longrightarrow U^2a | a \}$

8. Langages réguliers

8.1. Définition

Un langage est régulier si et seulement s'il existe une grammaire régulière générant ce langage .

Remarque :

Soit un alphabet $\sum$ fini.

- $\emptyset$ est un langage régulier ;
- $\{\varepsilon\}$ est un langage régulier ;
- Pour tout $a \in \sum$, $\{a\}$ est un langage régulier ;
- Si L^1 et L^2 sont réguliers, alors $L^1 \cdot L^2$ est régulier ;
- Si L est régulier, alors L* est régulier ;

Si L^1 et L^2 sont réguliers, alors $L^1 \cup L^2$ est régulier

9. Automates à états Finis

9.1. Définition

M = {Q, $\sum$,δ,q0, F} où

Q :ensemble fini des états;

q0 : état initial;

$\sum$: alphabet des symboles à l'entrée;

$\delta : Q \times \sum \rightarrow Q$: fonction de transition;

$F \subseteq Q$: ensemble des états accepteurs.

9.2. Représentation graphique d'un AF

Un **automate fini déterministe** (AFD) est un modèle mathématique utilisé en informatique théorique pour représenter et manipuler des langages formels. Il est

composé d'un ensemble fini d'états et de transitions entre ces états, et il fonctionne de manière déterministe : à partir d'un état donné et d'un symbole d'entrée, l'automate peut passer à un seul état suivant.

10.Notions fondamentales de la théorie des langages

10.1. Définitions

10.1.1. Symbole

Les symboles sont des éléments indivisibles qui vont servir de briques de base pour construire des mots.

Exemple :

– les chiffres 0,1,...,9 sont des symboles

– les lettres latine ,Arabe,Tamazight …….

10.1.2. Alphabet

Un alphabet est un ensemble fini de symboles. (vocabulaire :∑)

Exemple :{0,1,2,3,4,5,6,7,8,9}

{A,B,C,D,E,F,G,H,I,J,K,L,M}

10.1.3. Mot

Une suite de symboles, appartenant à un alphabet ∑ , mis bout à bout est appelé un mot sur ∑ .

Remarque :

- Le nombre de symboles entrant dans la composition d'un mot est appelé la longueur de a, que l'on note à l'aide de deux barres verticales : |a|.
- le mot vide (ε) ,sa longueur vaut zéro.
- La concaténation de deux mots a et b , notée a·b ou simplement ab est le mot obtenu en juxtaposant les symboles de b à la suite de ceux de a .
- L'ensemble de tous les mots que l'on peut construire sur un alphabet ∑, ε inclu, est noté ∑ *.

Exemple : Salam ,Digitalization,Then,If

- Propriété de la concaténation

Soient w, w1 et w2 trois mots définis sur l'alphabet ∑ :
– |w1 .w2 | = |w1 | + |w2 |
– ∀a ∈ ∑ : |w1 .w2 |a = |w1 |a + |w2 |a
– (w1 .w2).w3 = w1 .(w2 .w3)

- – w.ε = ε.w = w
- L'exposant

L'opération w.w est notée par w^2 . En généralisant, on note w^n = w...w . En particulier, n fois

$w^0 = \varepsilon$.

- Le mot miroir

Soit $w = a^1 a^2 ...a^n$ un mot sur ∑. On appelle mot miroir de w et on le note par w^R et on l'écrit w l'envers, c'est-à-dire que $w^R = a^n ...a^2 a^1$.

Les mots qui sont égaux à leur miroir, appelés palindromes,.

- Préfixe et suffixe

Soit w un mot défini sur un alphabet ∑. Un mot x (resp. y) formé sur ∑ est un préfixe (resp. suffixe) de w s'il existe un mot u formé sur ∑ (resp. v formé sur ∑) tel que w = xu (resp.w = vy). Si $w = a^1 a^2 ...a^n$ alors tous les mots de l'ensemble $\{\varepsilon, a^1, a^1 a^2, a^1 a^2 a^3, ..., a^1 a^2 ...a^n\}$

sont des préfixes de w. De même, tous les mots de l'ensemble $\{\varepsilon, a^n, a^{n-1} a^n, a^{n-2} a^{n-1} a^n, ..., a^1 a^2 ...a^n\}$ sont des suffixes de w.

- Mots conjugués

Deux mots x et y sont dits conjugués s'il existe deux mots u et v tels que : x = uv et y = vu.

11.Langages

11.1. Définition

Un langage, défini sur un alphabet ∑ , est un ensemble de mots définis sur ∑ . Autrement dit, un langage est un sous-ensemble de ∑ *.

Remarque :

- Deux langages particuliers sont indépendants de l'alphabet ∑ :

- le langage vide (L = Ø),

- le langage contenant le seul mot vide (L = {ε}).

Exemple : Langage des mots de longueur 2 défini sur l'alphabet {0, 1}est {00, 01, 10, 11} ;

12.Opérations sur les langages

Pour les langages, plusieurs opérations ensemblistes peuvent être appliquées. Soient L, L^1 et L^2 trois langages définis sur l'alphabet ∑, on définit les opérations suivantes **:**

12.1. Union

L'union de deux langages L^1 et L^2, est le langage, noté:

$L^1 \cup L^2$ constitué des mots appartenant à L^1 ou à L^2.

$$L^1 \cup L^2 = \{x | x \in L^1 \text{ ou } x \in L^2\}$$

12.2. Intersection

L'intersection de L^1 et L^2, est le langage, noté : $L^1 \cap L^2$ constitué des mots appartenant à L^1 et à L^2.

$$L^1 \cap L^2 = \{x | x \in L^1 \text{ et } x \in L^2\}$$

12.3. Différence

La différence de L^1 et L^2 est le langage, noté $L^1 - L^2$, constitué des mots appartenant à L^1 et n'appartenant pas à L^2.

$$L^1 - L^2 = \{x | x \in L^1 \text{ et } x /\in L^2\}$$

12.4. Concaténation

La concaténation de deux langages L^1 et L^2 est le langage noté $L^1 L^2$ composé des mots xy tels que $x \in L^1$ et $y \in L^2$.

$$L^1 L^2 = \{xy / x \in L^1 \text{ et } y \in L^2\}$$

12.5. Fermeture de Kleene

on note L^n la concaténation de L avec lui-même n fois :

On définit enfin la fermeture de Kleene du langage L, notée L^* de la façon suivante :

$$L^* = \bigcup_{k \geq 0} L^k$$

13.Grammaires

13.1. Définition

Une grammaire est un quadruplet G = (T,N, S,R) tel que :

– T est le vocabulaire terminal ou l'alphabet sur lequel est défini le langage.

– N est le vocabulaire non terminal, c'est-à-dire l'ensemble des symboles qui n'apparaissent pas dans les mots générés, mais qui sont utilisés au cours de la génération.

Un symbole non terminal désigne une "catégorie syntaxique".

– R est un ensemble de règles dites de réécriture ou de production de la forme :

$$u^1 \longrightarrow u^2, \text{ avec } u^1 \in (NUT)^+ \text{ et } u^2 \in (NUT)^*$$

La signification intuitive de ces règles est que la suite non vide de symboles terminaux ou non terminaux u^1 peut être remplacée par la suite éventuellement vide de symboles terminaux ou non terminaux u^2.

– $S \in N$ est le symbole de départ ou axiome. C'est à partir de ce symbole non terminal que l'on commencera la génération de mots au moyen des règles de la grammaire.

13.2. Terminologie :

- une suite de symboles terminaux et non terminaux $(N \cup T)^*$ est appelée une forme.
- une règle $u1 \longrightarrow u$ telle que $u \in T$ est appelée une règle terminale.
- Lorsque plusieurs règles de grammaire ont une même forme en partie gauche :

Remarque:

S ⟶ ab,
S ⟶ aSb,
S ⟶ c

pourra s'écrire S ⟶ ab|aSb|c.

13.3. Définition (Dérivation en une étape)

Soient une grammaire G = (T,N,S,R), une forme non vide u ∈ $(NUT)^+$ et une forme éventuellement vide v ∈(NUT)*. La grammaire G permet de dériver v de u en une étape (noté u ⇒ v) si et seulement si :

- u = x ù y (u peut être décomposé en x, ù et y ; x et y peuvent être vides),
- v = x v̀ y (v peut être décomposé en x, v̀ et y),
- ù ⟶ v̀ est une règle de R.

13.4. Définition (Dérivation en plusieurs étapes)

Une forme v peut être dérivée d'une forme u en plusieurs étapes :

- u ⇒ v : si v peut être obtenue de u par une succession de 1 ou plusieurs dérivations en une étape.
- u ⇒* v : si v peut être obtenue de u par une succession de 0, 1 ou plusieurs dérivations en une étape.

14. Relations entre les Langages et les Grammaires

14.1. Langage généré par une grammaire

Définition

Le langage généré par une grammaire G = (T,N,S,R) est l'ensemble des mots sur T qui peuvent être dérivés à partir de S :

$$L(G) = \{v \in T^* / S \Rightarrow^* v\}$$

Remarque:

- Une grammaire définit un seul langage.

- un même langage peut être engendré par plusieurs grammaires différentes.
- Deux grammaires G et G′ sont équivalentes si les langages L(G) et L(G′) sont identiques.

15.Grammaire régulière

15.1. Définition1

une grammaire G = (T,N, S,R) est régulière

- à droite si les règles de R sont de la forme:

A $\longrightarrow$ aB ou A $\longrightarrow$ a avecA,B ∈ N et a ∈ T

Exemple : soit la grammaire régulière à droite : G1 = (T^1,N^1, S^1,R^1) avec

T^1 = { a, b }

N^1 = {S^1,U^1 }

R^1 = {S^1 $\longrightarrow$ aS^1| aU^1

U^1 $\longrightarrow$ bU^1| b }

15.2. Définition2

Une grammaire G = (T,N, S,R) est régulière

– à gauche si les règles de R sont de la forme

A$\longrightarrow$Ba ou A$\longrightarrow$ a avecA,B ∈ N et a ∈ T

Exemple : soit la grammaire régulière à gauche : G2 = (T^2,N^2, S^2,R^2) avec

T^2 = { a, b }

N^2 = {S^2,U^2 }

R^2 = {S^2 $\longrightarrow$ S^2b |U^2b

$U^2 \longrightarrow U^2a \mid a \}$

16.Langages réguliers

16.1. Définition

Un langage est régulier si et seulement s'il existe une grammaire régulière générant ce langage .

Remarque :

Soit un alphabet $\sum$ fini.

- $\emptyset$ est un langage régulier ;
- $\{\varepsilon\}$ est un langage régulier ;
- Pour tout $a \in \sum$, $\{a\}$ est un langage régulier ;
- Si L^1 et L^2 sont réguliers, alors $L^1 \cdot L^2$ est régulier ;
- Si L est régulier, alors L* est régulier ;

Si L^1 et L^2 sont réguliers, alors $L^1 \cup L^2$ est régulier

17.Automates à états Finis

17.1. Définition

M = {Q, $\sum$,δ,q0, F} où

Q :ensemble fini des états;

q0 : état initial;

$\sum$: alphabet des symboles à l'entrée;

$\delta : Q \times \sum \rightarrow Q$: fonction de transition;

$F \subseteq Q$: ensemble des états accepteurs.

17.2. Représentation graphique d'un AF

Un **automate fini déterministe** (AFD) est un modèle mathématique utilisé en informatique théorique pour représenter et manipuler des langages formels. Il est composé d'un ensemble fini d'états et de transitions entre ces états, et il fonctionne de manière déterministe : à partir d'un état donné et d'un symbole d'entrée, l'automate peut passer à un seul état suivant.

18.Notions fondamentales de la théorie des langages

18.1. Définitions

18.1.1. Symbole

Les symboles sont des éléments indivisibles qui vont servir de briques de base pour construire des mots.

Exemple :

– les chiffres 0,1,...,9 sont des symboles

– les lettres latine ,Arabe,Tamazight …….

18.1.2. Alphabet

Un alphabet est un ensemble fini de symboles. (vocabulaire :∑)

Exemple :{0,1,2,3,4,5,6,7,8,9}

{A,B,C,D,E,F,G,H,I,J,K,L,M}

18.1.3. Mot

Une suite de symboles, appartenant à un alphabet ∑ , mis bout à bout est appelé un mot sur ∑ .

Remarque :

- Le nombre de symboles entrant dans la composition d'un mot est appelé la longueur de a, que l'on note à l'aide de deux barres verticales : |a|.
- le mot vide (ε) ,sa longueur vaut zéro.
- La concaténation de deux mots a et b , notée a·b ou simplement ab est le mot obtenu en juxtaposant les symboles de b à la suite de ceux de a .

- L'ensemble de tous les mots que l'on peut construire sur un alphabet ∑, ε inclu, est noté ∑ *.

Exemple : Salam ,Digitalization,Then,If

- Propriété de la concaténation

Soient w, w1 et w2 trois mots définis sur l'alphabet ∑ :
– |w1 .w2 | = |w1 | + |w2 |
– ∀a ∈ ∑ : |w1 .w2 |a = |w1 |a + |w2 |a
– (w1 .w2).w3 = w1 .(w2 .w3)

- – w.ε = ε.w = w
- L'exposant

L'opération w.w est notée par w^2 . En généralisant, on note w^n = w...w . En particulier, n fois

$w^0 = \varepsilon$.

- Le mot miroir

Soit w = a^1 a^2 ...a^n un mot sur ∑. On appelle mot miroir de w et on le note par w^R et on l'écrit w l'envers, c'est-à-dire que w^R = a^n ...a^2 a^1 .

Les mots qui sont égaux à leur miroir, appelés palindromes,.

- Préfixe et suffixe

Soit w un mot défini sur un alphabet ∑. Un mot x (resp. y) formé sur ∑ est un préfixe (resp. suffixe) de w s'il existe un mot u formé sur ∑ (resp. v formé sur ∑) tel que w = xu (resp.w = vy). Si w = a^1 a^2 ...a^n alors tous les mots de l'ensemble {ε, a^1 , a^1 a^2 , a^1 a^2 a^3 , ..., a^1 a^2 ...a^n }

sont des préfixes de w. De même, tous les mots de l'ensemble {ε, a^n , a^{n-1} a^n , a^{n-2} a^{n-1} a^n , ...,a^1 a^2 ...a^n } sont des suffixes de w.

- Mots conjugués

Deux mots x et y sont dits conjugués s'il existe deux mots u et v tels que : x = uv et y = vu.

19.Langages

19.1. Définition

Un langage, défini sur un alphabet $\sum$, est un ensemble de mots définis sur $\sum$. Autrement dit, un langage est un sous-ensemble de $\sum$ *.

Remarque :

- Deux langages particuliers sont indépendants de l'alphabet $\sum$:

- le langage vide (L = Ø),

- le langage contenant le seul mot vide (L = {ε}).

Exemple : Langage des mots de longueur 2 défini sur l'alphabet {0, 1}est {00, 01, 10, 11} ;

20. Opérations sur les langages

Pour les langages, plusieurs opérations ensemblistes peuvent être appliquées. Soient L, L^1 et L^2 trois langages définis sur l'alphabet $\sum$, on définit les opérations suivantes **:**

20.1. Union

L'union de deux langages L^1 et L^2, est le langage, noté:

$L^1 \cup L^2$ constitué des mots appartenant à L^1 ou à L^2.

$$L^1 \cup L^2 = \{x | x \in L^1 \text{ ou } x \in L^2\}$$

20.2. Intersection

L'intersection de L^1 et L^2, est le langage, noté : $L^1 \cap L^2$ constitué des mots appartenant à L^1 et à L^2.

$$L^1 \cap L^2 = \{x | x \in L^1 \text{ et } x \in L^2\}$$

20.3. Différence

La différence de L^1 et L^2 est le langage, noté $L^1 - L^2$, constitué des mots appartenant à L^1 et n'appartenant pas à L^2.

$$L^1 - L^2 = \{x | x \in L^1 \text{ et } x /\in L^2\}$$

20.4. Concaténation

La concaténation de deux langages L^1 et L^2 est le langage noté $L^1 L^2$ composé des mots xy tels que $x \in L^1$ et $y \in L^2$.

$$L^1 L^2 = \{xy / x \in L^1 \text{ et } y \in L^2\}$$

20.5. Fermeture de Kleene

on note L^n la concaténation de L avec lui-même n fois :

On définit enfin la fermeture de Kleene du langage L, notée L^* de la façon suivante :

$$L^* = \cup L^k \quad k \geq 0$$

21.Grammaires

21.1. Définition

Une grammaire est un quadruplet G = (T,N, S,R) tel que :

– T est le vocabulaire terminal ou l'alphabet sur lequel est défini le langage.

– N est le vocabulaire non terminal, c'est-à-dire l'ensemble des symboles qui n'apparaissent pas dans les mots générés, mais qui sont utilisés au cours de la génération.

Un symbole non terminal désigne une "catégorie syntaxique".

– R est un ensemble de règles dites de réécriture ou de production de la forme :

$$u^1 \longrightarrow u^2, \text{ avec } u^1 \in (NUT)^+ \text{ et } u^2 \in (NUT)^*$$

La signification intuitive de ces règles est que la suite non vide de symboles terminaux ou non terminaux u^1 peut être remplacée par la suite éventuellement vide de symboles terminaux ou non terminaux u^2.

– $S \in N$ est le symbole de départ ou axiome. C'est à partir de ce symbole non terminal que l'on commencera la génération de mots au moyen des règles de la grammaire.

21.2. Terminologie :

- une suite de symboles terminaux et non terminaux $(N \cup T)^*$ est appelée une forme.

- une règle u1 ⟶ u telle que u ∈ T est appelée une règle terminale.
- Lorsque plusieurs règles de grammaire ont une même forme en partie gauche :

Remarque:

S ⟶ ab,
S ⟶ aSb,
S ⟶ c

pourra s'écrire S ⟶ ab|aSb|c.

21.3. Définition (Dérivation en une étape)

Soient une grammaire G = (T,N,S,R), une forme non vide u ∈ $(NUT)^+$ et une forme éventuellement vide v ∈(NUT)*. La grammaire G permet de dériver v de u en une étape (noté u ⇒ v) si et seulement si :

- u = x ù y (u peut être décomposé en x, ù et y ; x et y peuvent être vides),
- v = x v̀ y (v peut être décomposé en x, v̀ et y),
- ù ⟶ v̀ est une règle de R.

21.4. Définition (Dérivation en plusieurs étapes)

Une forme v peut être dérivée d'une forme u en plusieurs étapes :

- u ⇒ v : si v peut être obtenue de u par une succession de 1 ou plusieurs dérivations en une étape.
- u ⇒* v : si v peut être obtenue de u par une succession de 0, 1 ou plusieurs dérivations en une étape.

22. Relations entre les Langages et les Grammaires

22.1. Langage généré par une grammaire

Définition

Le langage généré par une grammaire G = (T,N,S,R) est l'ensemble des mots sur T qui peuvent être dérivés à partir de S :

$$L(G) = \{v \in T^* / S \Rightarrow^* v\}$$

Remarque:

- Une grammaire définit un seul langage.
- un même langage peut être engendré par plusieurs grammaires différentes.
- Deux grammaires G et G′ sont équivalentes si les langages L(G) et L(G′) sont identiques.

22.2. Types de grammaire

En introduisant des critères plus ou moins restrictifs sur la forme des règles de grammaire, on obtient des classes de grammaires hiérarchisées, ordonnées par inclusion. La classification des grammaires, définie en 1957 par Noam CHOMSKY,distingue les quatre classes suivantes :

Type 0 : pas de restriction sur les règles**.**

Type 1 : grammaires contextuelles(sensibles au contexte) ou Les règles de R sont de la forme :

$$uAv \rightarrow uwv \text{ avec } A \in N, u,v \in (N \cup T)^* \text{ et } w \in (N \cup T)^+$$

Autrement dit, le symbole non terminal A est remplacé par w si on a les contextes u à gauche et v à droite.

Type 2 : grammaires hors-contexte. Les règles de R sont de la forme :

$$A \rightarrow w \text{ avec } A \in N \text{ et } w \in (N \cup T)^*$$

Autrement dit, le membre de gauche de chaque règle est constitué d'un seul symbole non terminal.

Type 3 : une grammaire G = (T,N, S,R) est régulière :

- à droite si les règles de R sont de la forme:

$A \rightarrow aB$ ou $A \rightarrow a$ avec$A,B \in N$ et $a \in T$

Exemple de grammaire régulière à droite :

$G^1 = (T^1,N^1, S^1,R^1)$ avec

$T^1 = \{ a, b \}$

$N^1 = \{S^1, U^1\}$

$R^1 = \{S^1 \longrightarrow aS^1\ aU^1$

$U^1 \longrightarrow bU^1\ b\}$

- à droite si les règles de R sont de la forme:

$A \rightarrow aB$ ou $A \rightarrow a$ avec$A,B \in N$ et $a \in T$

Exemple de grammaire régulière à droite :

$G^1 = (T^1, N^1, S^1, R^1)$ avec

$T^1 = \{a, b\}$

$N^1 = \{S^1, U^1\}$

$R^1 = \{S^1 \longrightarrow aS^1\ aU^1$

$U^1 \longrightarrow bU^1\ b\}$

23.Grammaire régulière

23.1. Définition1

une grammaire G = (T,N, S,R) est régulière

- à droite si les règles de R sont de la forme:

$A \longrightarrow aB$ ou $A \longrightarrow a$ avec$A,B \in N$ et $a \in T$

Exemple : soit la grammaire régulière à droite : $G1 = (T^1, N^1, S^1, R^1)$ avec

$T^1 = \{a, b\}$

$N^1 = \{S^1, U^1\}$

$R^1 = \{S^1 \longrightarrow aS^1 | aU^1$

$U^1 \longrightarrow bU^1 | b\}$

23.2. Définition2

Une grammaire G = (T,N, S,R) est régulière

– à gauche si les règles de R sont de la forme

$A \longrightarrow Ba$ ou $A \longrightarrow a$ avec$A,B \in N$ et $a \in T$

Exemple : soit la grammaire régulière à gauche : $G2 = (T^2,N^2, S^2,R^2)$ avec

$T^2 = \{ a, b \}$

$N^2 = \{S^2,U^2 \}$

$R^2 = \{S^2 \longrightarrow S^2b \mid U^2b$

$U^2 \longrightarrow U^2a \mid a \}$

24.Langages réguliers

24.1. Définition

Un langage est régulier si et seulement s'il existe une grammaire régulière générant ce langage .

Remarque :

Soit un alphabet $\sum$ fini.

- $\emptyset$ est un langage régulier ;
- $\{\varepsilon\}$ est un langage régulier ;
- Pour tout $a \in \sum$, $\{a\}$ est un langage régulier ;
- Si L^1 et L^2 sont réguliers, alors $L^1 \cdot L^2$ est régulier ;
- Si L est régulier, alors L* est régulier ;

Si L^1 et L^2 sont réguliers, alors $L^1 \cup L^2$ est régulier

25.Automates à états Finis

25.1. Définition

M = {Q, ∑,δ,q0, F} où

Q :ensemble fini des états;

q0 : état initial;

∑ : alphabet des symboles à l'entrée;

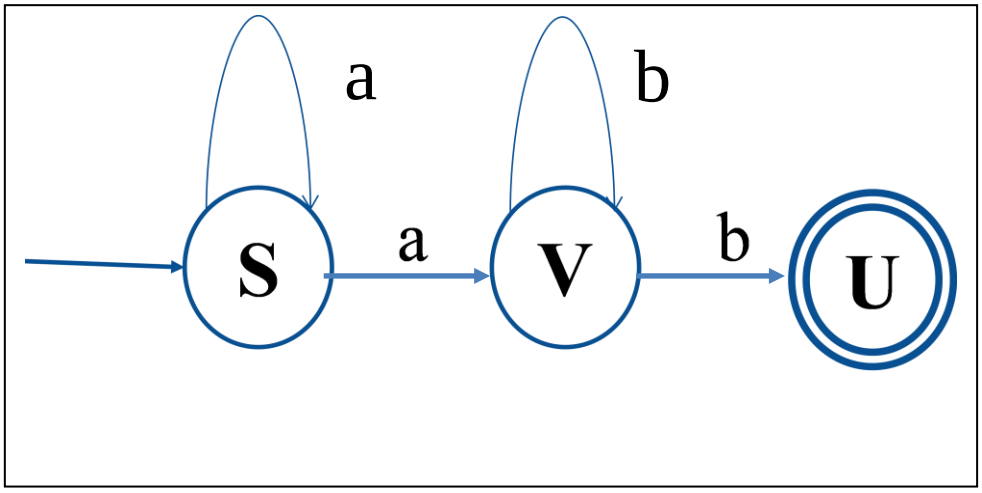
δ : Q × ∑ → Q : fonction de transition;

F ⊆ Q : ensemble des états accepteurs.

25.2. Représentation graphique d'un AF

Un **automate fini déterministe** (AFD) est un modèle mathématique utilisé en informatique théorique pour représenter et manipuler des langages formels. Il est composé d'un ensemble fini d'états et de transitions entre ces états, et il fonctionne de manière déterministe : à partir d'un état donné et d'un symbole d'entrée, l'automate peut passer à un seul état suivant.

On représente généralement un automate fini par un graphe orienté dont les arcs sont étiquetés.
Chaque état de l'automate est représenté par un sommet du graphe.
A chaque transition (Si; a; Sj) ∈ M on associe un arc du sommet Si vers le sommet Sj étiqueté par a.
Les sommets du graphe correspondant à des états initiaux de l'automate sont repérés par une pointe de flèche.
Les sommets du graphe correspondant à des états finaux sont entourés de deux cercles.



Automate Finale

26. Automate fini Déterministe /Indéterministe

Un AFD fonctionne comme un AFI, et on définit de la même manière les notions de configuration, dérivation entre configurations et acceptation d'un mot, la seule différence étant que la fonction de transition M détermine de façon unique le nouvel état dans lequel l'automate doit se placer au moment de faire une dérivation.

27. Equivalence entre automates finis et langages réguliers

27.1. Théorème 1

Tout langage accepté par un automate fini est régulier.

27.2. Théorème 2

Tout langage régulier est accepté par un automate fini.

28.Conclusion

La **théorie des langages de programmation** constitue un domaine clé de l'informatique, où s'entrelacent les concepts mathématiques, la logique formelle et l'ingénierie logicielle. Elle fournit les bases pour concevoir, analyser et comprendre les langages utilisés pour écrire des programmes informatiques, en mettant en lumière les principes sous-jacents à leur structure, leur syntaxe, leur sémantique et leur mise en œuvre.

Cette discipline explore les différents paradigmes de programmation (impératif, fonctionnel, logique, orienté objet, etc.), chacun offrant des approches spécifiques pour résoudre les problèmes. Elle examine également des aspects tels que l'optimisation des performances, la sûreté des programmes, et les propriétés de terminaison et de correction.

Enjeux et Contributions :

- **Compréhension** : Elle aide à clarifier les notions fondamentales des langages et à établir des liens entre eux.
- **Conception** : Elle permet de créer de nouveaux langages adaptés à des besoins spécifiques ou des domaines d'application.
- **Fiabilité** : Elle contribue à la formalisation et à la vérification de programmes, garantissant ainsi des logiciels plus sûrs et robustes.

En conclusion, la théorie des langages de programmation est un pilier fondamental pour l'évolution des technologies de programmation et pour le développement de systèmes logiciels modernes. Elle continue d'évoluer pour répondre aux défis posés par les systèmes distribués, l'intelligence artificielle et la transformation numérique.

Et pour bien assimiler le cours, les séances de travaux pratiques et travaux dirigées seront une occasion pour bien découvrir et comprendre ces concepts.

.