Langages formels et calculabilité
MIASHS L2

(d'après le cours de Julie Dugdale)

Jérôme GENSEL – Jerome.Gensel@univ-grenoble-alpes.fr
Université Grenoble Alpes France
UFR SHS – Laboratoire d'Informatique de Grenoble

Quelques informations pratiques…

Organisation du cours : Langages Formels et Calculabilité

- 10h30 de CM ➔ 7 séances de 1h30
  - Lundi de 8h30 à 10h
- 19h30 de TD : 3 groupes de TD ➔ 13 séances de 1h30
  - Le mardi, en général
- Si vous avez besoin de me contacter :
  - Bureau C14 à l'UFR
  - Jerome.Gensel@univ-grenoble-alpes.fr

- Ce cours et les TD associés seront déposés sur la plateforme Moodle
  - L2 - S4 - Langages formels et calculabilité

Examen & Contrôle Continu

La note finale est composée de :

- 50% de la note de l'Examen
- 50% de la note de Contrôle Continu

→ un test pendant un CM au mois de mars…

Plan du cours

- Chapitre 1 : mots, langages et expressions régulières
- Chapitre 2 : automates à états finis
- Chapitre 3 : automates à états finis non-déterministes
- Chapitre 4 : grammaires (2 cours)
- Chapitre 5 : automates à piles
- Chapitre 6 : machine de Turing

Mots, langages et expressions régulières

Sommaire

- Introduction au cours
- Qu'est-ce qu'un langage ?
- Alphabet, mots
- Propriétés des mots
- Langages
- L'étoile de Kleene
- Langages réguliers
- Expressions régulières

Introduction au cours

- Cours d'introduction à la théorie du calcul
- Une vue des fondements de l'informatique théorique
- Couvre des sujets tels que :
  les langages formels, les grammaires (formelles), les automates à états fini, la déterminisation des automates, les automates à piles, les *machines de Turing*, etc.

Introduction au cours

- C'est un cours théorique, mais ces sujets sont étroitement associés à des sujets pratiques
- Exemples:
  - La théorie des automates et les langages formels sont utilisés pour la construction de compilateurs, définition de langages de programmation
  - La définition de grammaires formelles est nécessaire pour la génération et la validation automatique de phrases (par exemple, pour la communication entre machines, pour vérifier la syntaxe des programmes, etc.)
  - Calculabilité : utilisé pour évaluer la possibilité d'écrire un programme pour résoudre un problème donné (peut être que le problème ne peut pas être résolu par un ordinateur)

Introduction au cours (d'après C. Solnon)

- Les langages servent à communiquer, à échanger des informations, des idées.
- Ils doivent aussi permettre aux humains de communiquer avec les ordinateurs, ces machines binaires ultra-rapides…
- Le langage commun entre les êtres humains est appelé *langage naturel*.
- Les phrases ou expressions de ce langage peuvent être informelles, ambiguës.
  - Elles nécessitent parfois une *interprétation* de notre cerveau en fonction du *contexte* pour en déterminer le sens.
- Par opposition, les langages créés par l'homme pour communiquer avec les ordinateurs sont des *langages artificiels*.
- Ils doivent être formalisés et non ambigus pour pouvoir être interprétés par une machine.

Introduction au cours (d'après C. Solnon)

- Basiquement, l'ordinateur ne comprend qu'un seul langage : son langage machine.
- Pour communiquer avec l'ordinateur à l'aide de langages plus évolués, il faut utiliser un *interprète* (qui traduit interactivement les instructions entrées au clavier), ou bien un *compilateur* (qui traduit tout un programme).
- L'interprétation ou la compilation d'un texte se décompose généralement en 3 étapes :
  1. L'analyse lexicale qui permet de décomposer le texte en entités élémentaires appelées lexèmes (*tokens* en anglais).
  2. L'analyse syntaxique qui permet de reconnaître des combinaisons de lexèmes formant des entités syntaxiques.
  3. L'analyse sémantique qui permet de générer le code objet directement compréhensible par la machine (ou bien un code intermédiaire qui devra être de nouveau traduit dans un code machine).

Introduction au cours (d'après C. Solnon)

- Considérons par exemple, le (morceau de) texte C suivant : *cpt = i + 3.14;*

  1. L'*analyse lexicale* permet d'identifier les lexèmes suivants : un *IDENTIFICATEUR* de valeur *cpt* , un *OPERATEUR* de valeur = , un *IDENTIFICATEUR* de valeur *i* , un *OPERATEUR* de valeur + , un *REEL* de valeur *3.14* et un *POINT VIRGULE* .
  2. L'*analyse syntaxique* permet de reconnaître que cette combinaison de lexèmes forme une instruction (expression) d'un langage de programmation (ici, le langage C) syntaxiquement correcte, et qu'il s'agit d'une affectation entre la variable d'identificateur *cpt* et l'expression arithmétique résultant de l'addition de la variable d'identificateur *i* avec le réel *3.14*.
  3. Enfin, l'*analyse sémantique* vérifie le bon typage des variables *cpt* et *i* , puis génère le code objet correspondant à cette instruction.

- Les phases d'analyse lexicale et syntaxique constituent en fait un même problème (à deux niveaux différents) : il s'agit de **reconnaître une combinaison valide d'entités** : une combinaison de caractères formant des lexèmes pour l'analyse lexicale, et une combinaison de lexèmes formant des programmes pour l'analyse syntaxique.
- La théorie des langages permet de résoudre ce type de problème.

Qu’est-ce qu’un langage ?

- Clarifier les termes de langage formel et langage naturel

- On peut différencier ces deux termes par la question :
  - Lequel est arrivé le premier, le langage ou ses règles grammaticales ?

Qu'est-ce qu'un langage ?

Langage naturel (par exemple, le français, l'anglais, l'espagnol, *etc.*) évolue avec le temps

- Sans considération pour les règles grammaticales formelles
- Les règles se sont développées plus tard pour expliquer la structure du langage, plutôt que pour la déterminer.

Qu'est-ce qu'un langage ?

- Au contraire, les **langages formels** (tels que ceux étudiés dans ce cours) sont définis par et se conforment à des **règles préétablies**

- Exemples :
  les langages de programmation, les langages mathématiques (algèbre, logique propositionnelle, etc.)

Qu'est-ce qu'un langage ?

- Nous nous situons ici à un niveau très rudimentaire de l'étude des «langages»

- Pour nous, un langage est simplement un ensemble de ***suites de lettres***…

  on dit aussi : mots,

  on dit aussi : chaînes de caractères

Qu'est-ce qu'un langage ?

- L'ensemble des phrases grammaticalement correctes du *français* standard est un langage
- L'ensemble des programmes syntaxiquement corrects écrits en *JAVA* est un langage
- {0, 01, 10, 00, 11, 000, 001, 010, 100, 011, 101, 110, 111} est un langage
- {*la, lala, lalala, lalalala, lalalalala, …, lalalala…la,* ….} est un langage
- {T, ⊥T, ⊥ ⊥T, ⊥ ⊥ ⊥T, …} est un langage

Alphabet, mots

- Alphabet : tout ensemble fini *non vide* de symboles dont les éléments sont appelés des lettres

- Remarque : dans certaines applications, on remplacera la notion d'alphabet par la notion de vocabulaire

- Un vocabulaire sera un ensemble de mots (cf. plus loin), pris eux-mêmes comme symboles de départ

Alphabet

- {0, 1}
- {a, b}
- {a, b, c}
- {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
- {α, β, χ, δ, ε, ϕ, γ, η, ι, φ, κ, λ, μ, ν, ο, π, θ, ρ, σ, τ, υ, ϖ, ω, ξ, ψ, ζ}

sont des alphabets couramment utilisés

Alphabet

- {0, 1}
  - On peut le considérer comme l'alphabet binaire
- {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
  - On peut le considérer comme l'alphabet français
- {α, β, χ, δ, ε, ϕ, γ, η, ι, φ, κ, λ, μ, ν, ο, π, θ, ρ, σ, τ, υ, ϖ, ω, ξ, ψ, ζ}
  - On peut le considérer comme l'alphabet grec

Mots

- Un mot est une suite finie de lettres
- Exemples:
  - 0010111100 est un mot sur {0, 1}
  - aaabbb est un mot sur {a, b}
  - acababbc est un mot sur {a, b, c}
  - BONJOUR est un mot sur {A, …, Z}
  - λογοσ est un mot sur {α, …, ζ}

Petite précision…

- Une suite finie d'éléments d'un ensemble A peut se définir de plusieurs manières :
  - C'est une application
    - s : {1, …, n} → A pour un n donné,

    Exemple:

    Avec Alphabet = A

    Application s(1) = B, s(2) = E, s(3) = N définit le mot BEN.

  - C'est un élément du produit cartésien A^n

    Exemple

    (B,E,N) est un élément de AxAxA (c.-à-d. A^3)

Petite précision…

- L'application `s` en question, de *{1,…, n}* dans A, n'est autre que l'application rang ➔ elle donne, pour un entier *i* tel que $1 \leq i \leq n$ et pour un certain *n*, la lettre qui, dans le mot, occupe le rang *n*

- On se convaincra facilement qu'il est équivalent de se donner un « mot », tel que abba et de se donner *l'application* qui indique pour chaque rang quelle est la lettre qui occupe ce rang

Propriétés des mots - Rang

- Définir un mot est donc équivalent à définir une application «rang », notée *s*: {*1,…,n*} → alphabet
- Exemples
  - 0010111100(1) = ?
    0010111100(1) = 0
  - 0010111100(2) = ?
    0010111100(2) = 0
  - 0010111100(3) = ?
    0010111100(3) = 1
  - BONJOUR(4) = ?
    BONJOUR(4) = J
  - BONJOUR(6) = ?
    BONJOUR(6) = U
  - BONJOUR(7) = ?
    BONJOUR(7) = R

Propriétés des mots - Longueur

- L'entier *n* maximum pour lequel *s* est défini s'appelle la longueur du mot s
- Exemples
  - BONJOUR(7) = R
  - BONJOUR(8) = non défini
  - BONJOUR est un mot de longueur 7

- La **longueur** d'un mot est la **position** de sa **dernière lettre ou..**
- **La *longueur* d'une chaîne est le nombre de caractères dans la chaîne**

Propriétés des mots – Ensemble de tous les mots

- Soit ∑ un alphabet = {a, b, c}
- L'ensemble de **tous les mots** qui peuvent être construits à partir de l'alphabet est noté $\sum^*$
- Les éléments de ∑* incluent :

  Longueur 0: ***ε*** ➔ ***mot de longueur 0***
  Longueur 1: a b c ➔ ***mots de longueur 1***
  Longueur 2: aa ab ac ba bb bc ca cb cc ➔ ***mots de longueur 2***
  Longueur 3: aaa aab aac aba abb abc aca acb acc
  baa bab bac bba bbb bbc bca bcb bcc
  caa cab cac cba cbb cbc cca ccb ccc ➔ ***mots de longueur 3***
  …
  Longueur *n* : … ➔ ***mots de longueur*** **n**
  …
  Longueur ∞

ε est unique, inclus dans tout ∑* quelque soit ∑

Propriétés des mots – mot vides

- **Le *mot vide* est l'unique chaîne sur Σ de longueur 0, et est notée *ε* ou *λ***
- Soit ε le mot vide
  - D'après ce qui précède, ε est unique.
  - Soit A* l'ensemble des mots sur l'alphabet A, on a toujours : ε ∈A*

Concaténation de mots

- Soit 2 mots s et s' sur A
- S'ils sont définis par les applications
  s: {1, …, n} → A et s' : {1, …, m} → A
- La concaténation de s et s' est notée
  s^s'
- Le nouveau mot s^s' est défini par:
  s^s' : {1, …, n+m} → A, définie par:
  - pour tout i, si $1 \leq i \leq n$, s^s'(i) = s(i)
  - si $n < i \leq n+m$, s^s' (i) = s' (i - n)

Concaténation de mots - Exemple

- Sur {a, b, c}, abbac est un mot, qui peut être défini par l'application s suivante :
  s(1) = a,
  s(2) = b,
  s(3) = b,
  s(4) = a,
  s(5) = c

- Sur {a, b, c}, bcca est un autre mot, qui peut être défini par s' :
  s'(1) = b,
  s'(2) = c,
  s'(3) = c,
  s'(4) = a

- On vérifiera que leur concaténation s^s' correspond au mot :
  abbacbcca

Propriétés de la concaténation

- La concaténation est associative : (A^B)^C = A^(B^C)

- La concaténation a un élément neutre : ε

  ε^x = x^ε = x

Stratification

- Parmi tous les mots sur A, il y a :
  - Les mots de longueur nulle, formant l'ensemble {ε}
  - Les mots de longueur 1, donnant simplement l'ensemble A (car on ne fait pas dans ce cas de différence entre *lettre* et *mot d'1 seule lettre*)
  - Les mots de longueur 2, formant A^2,

    Exemple

    A= {1, 2, 3}

    A^2={(1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2)(3,3)}
  - …
  - Les mots de longueur n, formant A^n
  - *etc.*

Stratification

- Dans l'exemple précédent :

  Soit $\sum$ l'alphabet = {a, b, c}

  Longueur 0 ➔ $\sum^0 = \{\varepsilon\}$

  Longueur 1:➔ $\sum$ ={a,b,c}

  Longueur 2:

  ➔ $\sum^2$ ={aa, ab, ac, ba, bb, bc, ca, cb, cc}

  Longueur 3:

  ➔ $\sum^3$ ={aaa, aab, aac, aba, abb, abc, aca, acb, acc,baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc,caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc}

  …

Stratification

- A* (l'ensemble de *tous* les mots sur l'alphabet A) peut être noté en utilisant la stratification :

  $A^* = \{\varepsilon\} \cup A \cup A^2 \cup A^3 \ldots \cup A^n \ldots$

- Ou bien encore

$$A^* = \bigcup_{n=0}^{\infty} A^n$$

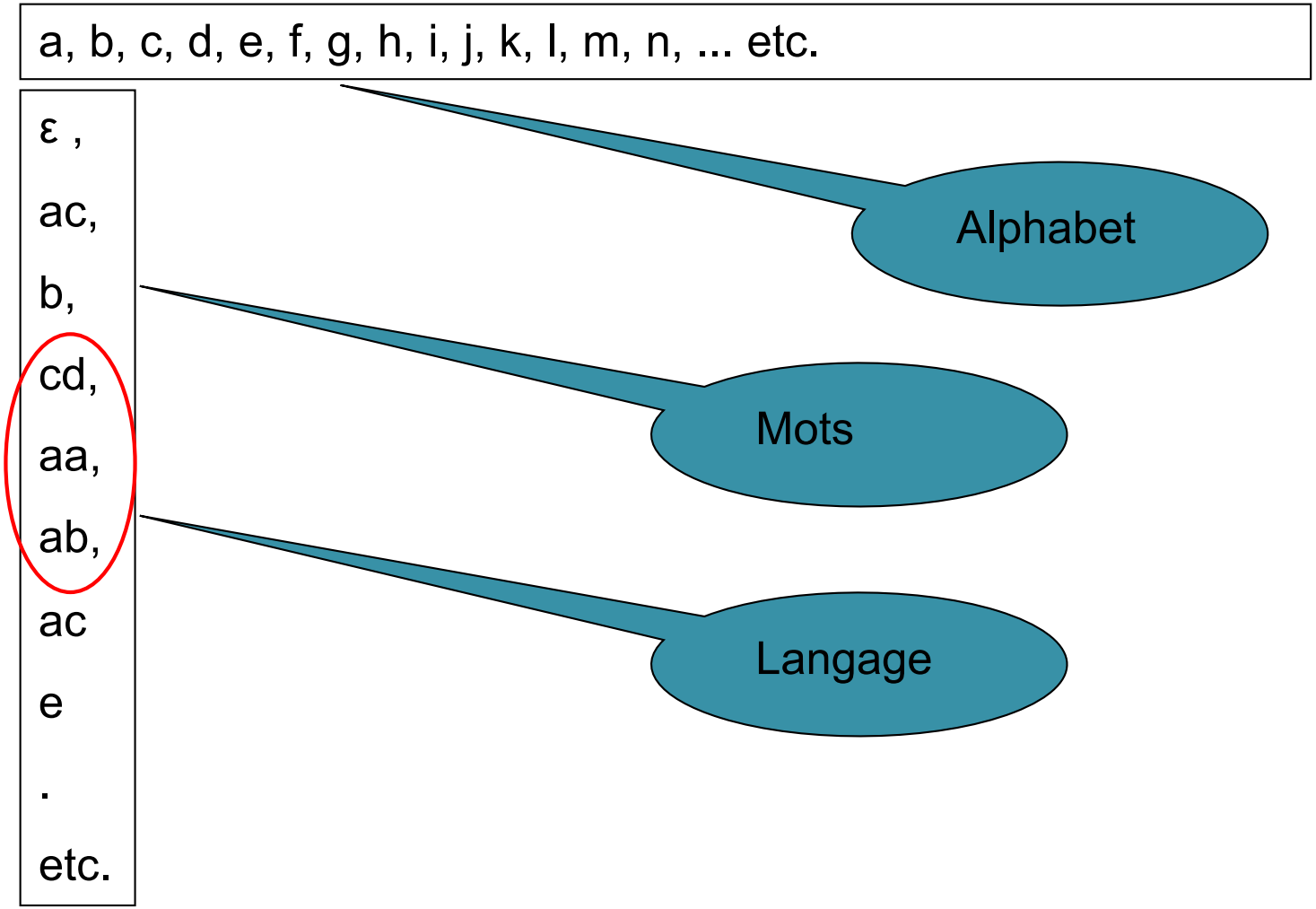
  - Avec $A^0 = \{\varepsilon\}$

Langages - Définition

- Un langage sur un alphabet A sera défini simplement comme une partie de A*
- Donc, un sous-ensemble de A* ($\sum^*$) est appelé un langage
- Ainsi, parmi les langages possibles sur A:
  - $\varnothing$ est le langage vide
  - $\{\varepsilon\}$ est le langage réduit au mot vide
  - A* est le langage plein

Langages – Relations entre alphabet, mots, et langages

- Exemple



Langages – Opérations - **Union**

- Si L et L' sont des langages respectivement définis sur les alphabets A et A', l'**union de L et de L'**, notée **L∪L'**, est un langage sur l'alphabet A∪A' qui contient les mots qui sont, soit contenus dans L, soit contenus dans L'
- L∪L' = {u, tel que u∈L ou u∈L'}
- Exemple:

L = {de}
L' = {main}
L∪L' = {de, main}

Langages – Opérations - **Produit**

- Si L et L' sont des langages respectivement définis sur les alphabets A et A', le **produit ou concaténation de L et de L'**, noté **L.L'**, est un langage sur l'alphabet $A \cup A'$ contenant tous les mots formés d'un mot de L suivi d'un mot de L'
- $L.L' = \{w=u\hat{}v$, tels que $u \in L$ et $v \in L'\}$
- Exemple:

L = {de}
L' = {main}
L.L' = {demain}

Langages – Propriétés des opérations

- Soit sur {a, b, c} les langages:
  L = {aa, bb}
  L' = {ca, cb, cc}
- On a:
  $L \cup L'$ = {aa, bb, ca, cb, cc}
  $L' \cup L$ = {ca, cb, cc, aa, bb}
  $L \cup L' = L' \cup L$
  ➔ L'union est une opération commutative
- On a:
  L.L' = {aaca, aacb, aacc, bbca, bbcb, bbcc}
  L'.L = {caaa, cabb, cbaa, cbbb, ccaa, ccbb}
  $L.L' \neq L'.L$
  ➔ Le produit n'est pas une opération commutative

Langages – Opérations – **L'étoile de Kleene**

- L'étoile de Kleene est une opération qui élargit un langage unique
  - Plutôt que de combiner plusieurs langages (comme on le faisait avec le produit), on forme toutes les concaténations de zéro ou plusieurs mots du même langage
  - Notez que le mot vide sera un élément du langage élargi (L*)

L’étoile de Kleene

- L'opération de l’étoile de Kleene est notée par ‘*’

- Note : nous avons déjà rencontré l’étoile de Kleene en utilisant $\sum^*$ (tous les mots finis qui peuvent être formés à partir de l'alphabet $\sum$)

L'étoile de Kleene

Exemple :

- Si L_1 est le langage {y}, alors L_1^* est le langage formé de la chaîne vide et de toutes les chaînes finies de 'y'

  $\{\varepsilon, y, yy, yyy, yyyy, \ldots.\}$

- Si L_2 est le langage {yy}, alors L_2^* est le langage formé de la chaîne vide et de toutes les chaînes constituées d'un nombre pair de 'y'

  $\{\varepsilon, yy, yyyy, yyyyyy, yyyyyyyy, \ldots.\}$

L’étoile de Kleene

- Un autre exercice :

  si L = {0, 1},

  L* = {ε, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, …}.

L'étoile de Kleene

- On peut définir également l'opération suivante, appelée opération de fermeture (de Kleene, ou itérative…)

$L^* = \{w \in A^*$, tel que $\exists n \geq 0, \exists w_1, w_2, \ldots, w_n \in A^*, w = w_1 \hat{} w_2 \ldots \hat{} w_n\}$

c.-à-d.: Toutes les concaténations possibles des mots de ce langage

L’étoile de Kleene - Exemple

- Un autre exemple :
  Soit L = {ab, ba} sur A = {a, b}
  L* = {ε, ab, ba, abab, abba, baab, baba, ababab, ababba, abbaab, abbaba, baabab, baabba, babaab, bababa, ...}

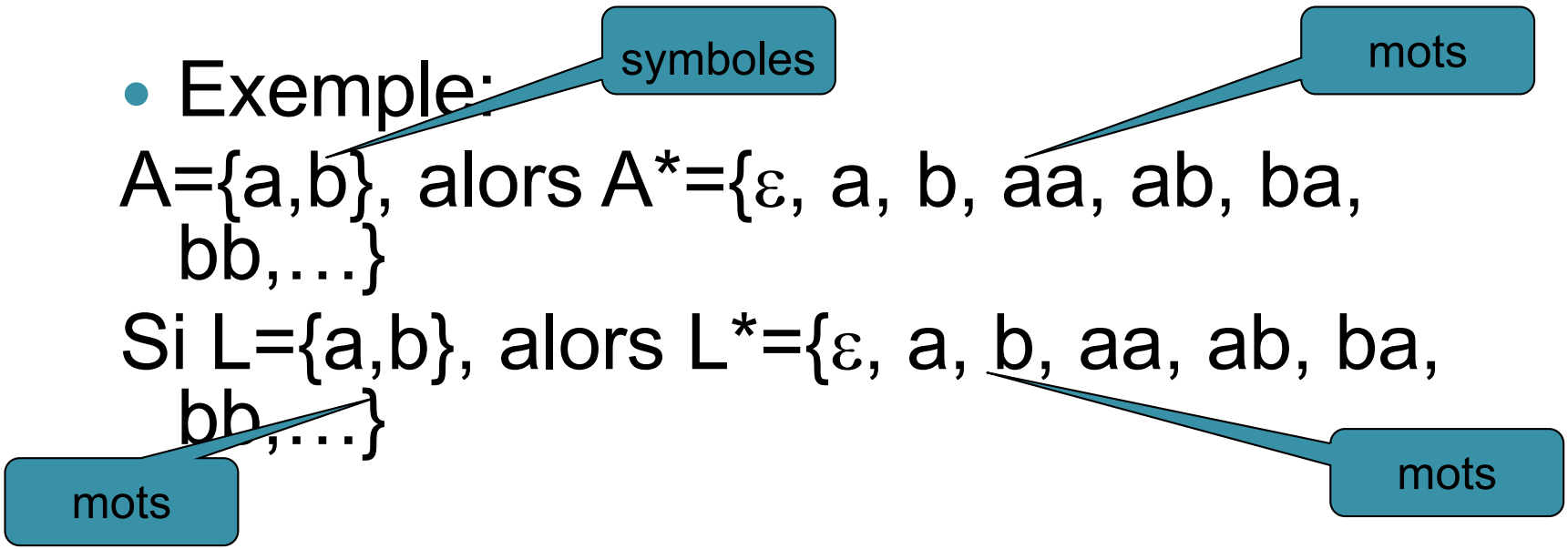
Langage des mots d'une lettre

- Soit L le langage des mots d'une lettre sur A, on vérifiera bien sûr que L* = A* autrement dit, l'étoile de «A*» (ensemble des mots sur A) est bien l'étoile de Kleene ! (ouf!)

- Exemple:

A={a,b}, alors A*={ε, a, b, aa, ab, ba, bb,…}

Si L={a,b}, alors L*={ε, a, b, aa, ab, ba, bb,…}



L'étoile de Kleene - Propriété

- On peut vérifier que, comme pour A*, on a la décomposition générale suivante :

$$L^* = \bigcup_{n=0}^{\infty} L^n = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cup L^4 \cup L^5 \ldots$$

- Où, bien entendu, L^n = L.L…L (n fois)
- On peut donc avoir une notation stratification (par longueur de mots) pour les langages

Propriétés des langages - Associativité

- Soit A, B et C, 3 langages. On a (associativité) :
  - A.(B.C) = (A.B).C
  - A∪(B∪C) = (A∪B) ∪C
- Exemple:

  L={a,b}

  G= {c}

  H={d,e}
- L.(G.H)= {a,b}.{c^d, c^e}

  = {a^(c^d), a^(c^e), b^(c^d), b^(c^e)}

  = {a^c^d, a^c^e, b^c^d, b^c^e}
- (L.G).H= {a^c, b^c}. {d,e}

  = {(a^c)^d, (a^c)^e, (b^c)^d, (b^c)^e}

  = {a^c^d, a^c^e, b^c^d, b^c^e}

Propriétés des langages – Distributivité et autres

- Soit G, H et F, 3 langages. On a (distributivité) :
  - $G.(H \cup F) = G.H \cup G.F$
- autres
  - $L.\varnothing = \varnothing$ $\quad L \cup \varnothing = L$
  - $L.\{\varepsilon\} = L$
  - $\varnothing^* = \{\varepsilon\}$ Convention
  - $\{\varepsilon\} \cup L.L^* = L^*$

Exemple d'utilisation de la distributivité

- Prouver que $\{\varepsilon\} \cup L.L^* = L^*$
  - On a $L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \ldots \cup L^n \ldots$
  - Donc $L.L^* = L.(\{\varepsilon\} \cup L \cup L^2 \cup L^3 \ldots \cup L^n \ldots)$
  - Par distributivité, $L.L^* = L.\{\varepsilon\} \cup L.L \cup L.L^2 \cup \ldots \cup L.L^n \ldots = L \cup L^2 \cup L^3 \ldots \cup L^n \ldots$
  - Donc $\{\varepsilon\} \cup L.L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \ldots \cup L^n \ldots = L^*$
  - cqfd !

Langages réguliers

- Un langage régulier sur un alphabet A est :
  - ou bien : $\varnothing$
  - ou bien : $\{\varepsilon\}$
  - ou bien : $\{x\}$ pour n'importe quel $x \in A$
    (aussi appelé un langage 'singleton' – un ensemble avec exactement un élément)
  - ou bien : un langage quelconque obtenu à partir de deux langages réguliers par **union** ou par **produit**
  - ou bien : un langage quelconque obtenu à partir d'un langage régulier par **fermeture de Kleene**

Langages réguliers

- Exemples typiques :
  - le langage formé de tous les mots de l'alphabet {*a*, *b*}
  - le langage formé de tous les mots de la forme : plusieurs 'a' suivis de plusieurs 'b'
  - le langage contenant un nombre pair de 'a',

Exemples - Langages réguliers

- Par constrcuction, voici des langages réguliers sur {a,b,c} :
  - {a}∪{b}
  - ({a}∪{b})*
  - {a}{a}{a}
  - {a}{a}{a} ({a}∪{b})*
  - ({ε}∪{a}{a})*
  - ({ε}∪{a}{a})*{b}{b}{b} ({a}∪{b})*
- Mais que sont-ils exactement ?

Exemples - Langages réguliers

- $\{a\} \cup \{b\} = \{a, b\}$
- $(\{a\} \cup \{b\})^* = \{a, b\}^*$
- $\{a\}\{a\}\{a\} = \{aaa\}$
- $\{a\}\{a\}\{a\}\ (\{a\} \cup \{b\})^*$ = ensemble des mots sur {a, b} qui commencent par aaa
- $(\{\varepsilon\} \cup \{a\}\{a\})^*$ = ensemble des mots constitués d'un nombre pair de a (y compris 0 'a' possible)
- $(\{\varepsilon\} \cup \{a\}\{a\})^*\{b\}\{b\}\{b\}\ (\{a\} \cup \{b\})^*$ = ensemble des mots sur {a, b} commençant par un nombre pair de a, y compris aucun, suivis de trois b

Exemples - Langages réguliers

- L'ensemble des représentations en binaire des entiers pairs est un langage régulier sur {0, 1}
  - car il s'écrit: ({0}∪{1})*0
- L'ensemble des mots se terminant en 'tion' est un langage régulier sur {a, b, …, z}
  - car il s'écrit: ({a}∪{b}∪{c}∪…{z})*{t}{i}{o}{n}
- L'ensemble des mots de 3 lettres sur {a, b} est un langage régulier
  - car il s'écrit:
    {a}{a}{a}∪{b}{a}{a}∪{a}{b}{a}∪{a}{a}{b}∪
    {a}{b}{b}∪{b}{a}{b}∪{b}{b}{a}∪{b}{b}{b}

Expressions régulières

- Les expressions régulières permettent de représenter un langage régulier
- On peut voir les expressions régulières comme une notation compacte pour les langages réguliers
  - ({ε}∪{a}{a})*{b}{b}{b} ({a}∪{b})* n'est pas compacte !

Expressions régulières

- Les expressions régulières servent à désigner les langages réguliers
  - $\varnothing$, ε et toute lettre $x \in A$ sont des expressions régulières sur A,
  - Si α et β sont des expressions régulières sur A, alors:
    - $\alpha+\beta$ et $\alpha\beta$ sont des expressions régulières sur A
    - $(\alpha)^*$ est aussi une expression régulière sur A
  - Note : '+' est la notation pour l'union dans les expressions régulières

Expressions régulières

- Des exemples d'expressions régulières sur {a,b,c} :
  - a+b
  - (a+b)*
  - aaa
  - aaa(a+b)*
  - (ε+aa)*
  - (ε+aa)*bbb(a+b)*

Expressions régulières

- a+b désigne {a}∪{b}
- (a+b)* désigne ({a}∪{b})*
- aaa désigne {a}{a}{a}
- aaa(a+b)* désigne {a}{a}{a} ({a}∪{b})*
- (ε+aa)* désigne ({ε}∪{a}{a})*
- (ε+aa)*bbb(a+b)* désigne ({ε}∪{a}{a})*{b}{b}{b} ({a}∪{b})*

Expressions régulières - Propriétés

- On retrouve sur les expressions régulières. les propriétés induites par les langages réguliers :
- associativité
  - $\alpha(\beta\gamma) = (\alpha\beta)\gamma$
  - $\alpha+(\beta+\gamma) = (\alpha+\beta)+\gamma$
- distributivité
  - $\alpha(\beta+\gamma) = \alpha\beta+ \alpha\gamma$
- autres
  - $\alpha\varnothing = \varnothing \quad \alpha+\varnothing = \alpha$
  - $\alpha\varepsilon = \alpha$
  - $\varnothing^* = \varepsilon$
  - $\varepsilon+ \alpha\alpha^* = \alpha^*$

Avec α, β, γ des expressions régulières

Expressions régulières - autres propriétés

- $(\alpha+\beta)^* = (\alpha^*+\beta^*)^* = (\alpha^*\beta^*)^* = (\alpha^*\beta)^*\alpha^* = \alpha^*(\beta\alpha^*)^*$
- $\alpha(\beta\alpha)^* = (\alpha\beta)^*\alpha$
- …

Expressions régulières - Exemples / Exercices

- ab*
- a*b (Exercice)
- (a+b)
- (a+b)* (Exercice)

Expressions régulières - Exemple

a*ba*(ba*ba*)*

c’est le langage des mots ayant un nombre impair de b

Expressions régulières - Exemples

- Désigner par une expression régulière:
  - L'ensemble des représentations en binaires d'entiers pairs:

    (0+1)*0
  - L'ensemble des mots ayant un nombre pair de lettres sur {a, b}:

    (aa+bb+ab+ba)*
  - L'ensemble des mots qui contiennent au moins une fois trois 'a' qui se suivent:

    (a+b)* aaa (a+b)*
  - L'ensemble des mots de 4 lettres sur {0, 1}:

    (0+1)(0+1)(0+1)(0+1)

Expressions régulières - Exercice

- Exprimer par une expression régulière la plus réduite possible la conjugaison en français, au *présent*, au *futur* et à *l'imparfait* du verbe *chanter*