Révisions Master IMA

Axel PIGEON

8 juillet 2025

Table des matières

Chapitre 1

Mots et Langages

Contents

Si on connaît plusieurs langages de programmation, on remarque que chaque langage, ou plutôt chaque paradigme de langage est spécialisé pour la résolution d'une carégorie dde problèmes. On pourrait se demander s'il est possible de créer un langage permettant de résoudre tous les problèmes. Pour cela, il nous faudrait d'abord être capable de définir formellement un langage, des mots, etc...

1.1 Alphabets et Mots

1.1.1 Premières Définitions

Commençons tout d'abord par redéfinir correctement la notion d'alphabet et de mot.

Définition (**Alphabet**) . Un alphabet est simplement un ensemble fini, noté Σ. On nomme "lettre" ou "symbole" les éléments d'un alphabet.

Exemple Quelques exemples d'alphabets :

- $\Sigma = \{a, b\}$
- $\Sigma = \{a, b, \ldots, \%, \$\}$

Définition (**Mot**) . Un mot est une suite finie de lettres d'un alphabet.

Proposition Le mot vide est noté ε. On note l'ensemble des mots d'un alphabet Σ^*.

Définition (**Longueur d'un mot**) . On note $|w|$ la longueur d'un mot $w \in \Sigma^*$ qui correspond au nombre de lettres, avec répétition du mot w.

Définition (**Egalité de mots**) . On dit que deux mots $u, v \in \Sigma^*$ son égaux ssi :
- $|v| = |u|$
- $\forall i \in [\![1, |v|]\!], \quad u[i] = v[i]$ où $u[i]$ est la ième lettre du mot u.

Deux mot sont égaux ssi ils sont de même longueur et sont composés des mêmes lettres dans le même ordre.

1.1.2 Opérations sur les mots

Sur les mots, on ne définit qu'une seule opération, la **concaténation**.

Définition (**Concaténation**) . Soient $u, v \in \Sigma^*$ deux mots définis sur un même alphabet. On appelle la concaténation l'application :

$$\begin{cases} \Sigma^* \times \Sigma^* \longrightarrow \Sigma^* \\ (u, v) \longmapsto w = u.v \end{cases}$$

Elle est définie telle que $\forall u, v \in \Sigma^*$ de longueur $n, p \in \mathbb{N}$, on ait :
- $|u.v| = |u| + |v| = n + p$
- $\forall i \in [\![1, n]\!] u.v[i] = u[i]$ et $\forall i \in [\![1, p]\!], u.v[n + i] = v[i]$

On parlera identiquement de concaténation ou de produit.

Remarque Cette définition reprend bien la caractérisation de deux mots égaux.

Proposition (Propriétés de la concaténation) La concaténation est une application :
- **Associative :** $\forall u, v, w \in \Sigma^*, \quad w.(u.v) = (w.u).v$
- **Pas commutative** pour un alphabet de plus d'une lettre.
- admet pour **élément neutre** le mot vide ε.

1.1.3 Puissance d'un mot

Une fois la concaténation définie pour un mot, on peut alors parler de puissance de mot. Définissons celle-ci par récurrence.

Définition (**Puissance d'un mot**) . Soit Σ un alphabet et $u \in \Sigma^*$, on a :
- $u^0 = \varepsilon$
- $u^1 = u$
- $\forall n \in \mathbb{N}, u^{n+1} = u^n.u$

Exemple $(ba)^3 = bababa$

Proposition Soient $u \in \Sigma^*$ on peut appliquer les règles "connues" des puissances d'où :

$$\forall n, p \in \mathbb{N}, \quad u^{n+p} = u^n.u^p = u^p.u^n$$

On remarque que l'on peut effectuer des simplifications sur les égalités de mots.

Propriété (**Simplifications**) . L'ensemble Σ^* est simplifiable à gauche et à droite.

- $\forall u, v, w \in \Sigma^*, \quad u.w = v.w \Longrightarrow u = v$
- $\forall u, v, w \in \Sigma^*, \quad w.u = w.v \Longrightarrow u = v$

Ici, pas besoin d'inverse, la démonstration repose sur la définition de l'égalité entre deux mots.

1.2 Relations d'Ordre

Dans l'alphabet dit "classique" on possède un ordre lexicographique des mots permettant de les classer en fonction de leurs lettres et de la position de leurs lettres. Ici, nous allons définir deux types de relations d'ordre sur les mots.
Commençons par rappeler la définition de relation d'ordre.

Définition (**Relation d'Ordre**) . Soit $\lhd$ une relation sur un ensemble E. On dit que $\lhd$ est une **relation d'ordre** ssi pour tout $x, y, z \in E$, $\lhd$ est :

- **Réflexive :** $x \lhd x$
- **Anti-Symétrique :** $x \lhd y$ et $y \lhd x \Longrightarrow x = y$
- **Transitive :** $x \lhd z$ et $z \lhd y \Longrightarrow x \lhd y$

1.2.1 Ordre Préfixe

Naturellement, on munitΣ^* d'un ordre préfixe permettant de classer les mots en fonction de leur préfixe. Cette relation peut être vue comme un forme d'inclusion de mots.

Définition (**Ordre Préfixe**) . Soient $u, w \in \Sigma^*$, on définit la relation d'ordre préfixe $\sqsubseteq$ telle que :

$$\boxed{u \sqsubseteq w \iff \exists v \in \Sigma^*, w = u.v}$$

Autrement dit, u est un préfixe de w ssi il existe un mot v tel que w soit composé de la concaténation de u et v.

Remarque L'ordre préfixe ne nécessite pas de relation d'ordre directement sur l'alphabet Σ.

Propriété (**Ordre Préfixe et égalité**) . Soient $u, v \in \Sigma^*$ on a :

$$u \sqsubseteq v \text{ et } v \sqsubseteq u \Longrightarrow u = v$$

Démonstration Soient $u, v \in \Sigma^*$ tels que $\exists x, y \in \Sigma^*$ tels que

$$u = v.x \quad \text{et} \quad v = u.y$$

On a alors que :

$$\begin{cases} v = v.y.x \\ yx = \varepsilon \end{cases} \Longrightarrow \begin{cases} y = \varepsilon \\ x = \varepsilon \end{cases} \Longrightarrow u = v$$

Remarque Attention : l'ordre préfixe est une relation d'ordre partielle. Autrement dit, tous les éléments d'un même alphabet de sont pas comparables.

1.2.2 Ordre Lexicographique

Définition (**Ordre Lexicographique**) . Soit Σ un alphabet que l'on muni d'une relation d'ordre $\leqslant$. L'ordre lexicographique $\leqslant$ est une relation d'ordre totale sur Σ^*.

Remarque Ici, nous avons bien besoin de définir au préalable un ordre sur notre alphabet Σ.

Propriété (**Compatibilité**) . L'ordre lexicographique est compatible avec l'ordre préfixe. Plus formellement,

$$\forall u, v \in \Sigma^*, \quad u \sqsubseteq v \Longrightarrow u \leqslant v$$

1.3 Langage

Maintenant que nous sommes au clair sur la définition de lettre et de mot, on peut enfin définir l'objet principal de ce chapitre, les langages.

1.3.1 Definition

Définition (**Langage**) . Soit Σ un alphabet, on appelle langage sur Σ toute partie de Σ^*.

Remarque L'ensemble de tous les langages d'un alphabet Σ est donc $\mathcal{P}(\Sigma^*)$, l'ensemble de toutes les parties de Σ^*.

Définition (**Complémentaire**) . Soit L un langage sur Σ. On définit le complémentaire de L dans Σ^* le langage :

$$\overline{L} = \{w,\ w \notin L\}$$

1.3.2 Opérations

De même que pour les alphabets et les mots, on peut définir des opérations sur les langages.

Définition (**Opérations sur les langages**) . Soit Σ un alphabet et $L_1, L_2 \subseteq \Sigma^*$ deux langages de Σ. On définit 4 principales opérations sur des langages :

- **Somme :** notée $+$, la somme de deux langages d'apparente à l'union des ensembles.

$$\boxed{L_1 + L_2 = \{w,\ w \in L_1 \text{ ou } w \in L_2\}}$$

  C'est une opération :
  — Commutative
  — Associative
  — dont $\emptyset$ est le neutre.
- **Intersection :** de même que pour les ensembles :

$$\boxed{L_1 \cap L_2 = \{w,\ w \in L_1 \text{ et } w \in L_2\}}$$

  C'est une opération :
  — Commutative
  — Associative
  — dont Σ^* est le neutre
- **Différence :** comme les ensembles, on définit la différence de langages :

$$\boxed{L_1/L_2 = \{w,\ w \in L_1 \text{ et } w \notin L_2\} = L_1 \cap \overline{L_2}}$$

- **Produit de concaténation :** de même que pour les mots, on peut généraliser le produit de concaténation aux langages :

$$\boxed{L_1.L_2 = \{u.v,\ u \in L_1, v \in L_2\}}$$

  C'est une opération :
  — Associative
  — Distributive par rapport à l'union
  — D'élément neutre $\{\varepsilon\}$.

Définition (**Puissance de langage**) . Soit L un langage, on définit **par récurrence** la puissance de L par :

- $L^0 = \{\varepsilon\}$
- $L^1 = L$
- $\forall n \in \mathbb{N}^*,\ L^n = L^{n-1}.L$

Une fois définies des opérations "simples" sur les langages, on peut en définir des plus complexes, permettant de "générer" un langage infini à partir d'un langage fini ou infini.

Définition (**Langage plus et étoile**) . Soit L un langage sur un alphabet Σ. On définit le langage plus de L comme le langage :

$$L^+ = L^1 + L^2 + \dots$$

De même le langage étoile de L est défini par :

$$L^* = \{\varepsilon\} + L^1 + L^2 + \dots$$

1.3.3 Propriétés

Voyons quelques propriétés des langages...

Proposition Soient L_1 et L_2 deux langages sur un alphabet Σ, on a les propriétés suivantes :

- $\forall p \in \mathbb{N}$, on a :
$$(L_1)^p.(L_2)^p \subseteq (L_1)^*.(L_2)^*$$
- L'opération étoile est idempotente :
$$(L^*)^* = L^*$$
- $L^* = \{\varepsilon\} + L^+$
- $\varepsilon \in L \iff L^+ = L^*$

1.3.4 Expressions Régulières

Lorsque l'on manipule des langages infinis, il serait appréciable d'avoir une expression pratique pour un langage permettant de directement voir la forme des mots qu'il contient. On définit ainsi les expressions régulières.

Définition (**Expression Régulière**) . On définit récurisevement une expression régulière sur un alphabet Σ :

- ε est une expression régulière.
- $\forall w \in \Sigma$ est une expression régulière.
- Si E est un expression régulière alors (E) l'est aussi.
- Si E_1 et E_2 sont des expressions régulières, alors $E_1 + E_2$ l'est aussi.
- Si E_1 et E_2 sont des expressions régulières, alors $E_1.E_2$ l'est aussi.
- Si E est une expression régulière, alors E^* l'est aussi.

Exemple Voyons quelques exemples d'expressions régulières sur un alphabet $\Sigma = \{a, b\}$:

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

Définition (**Langage Régulier**) . On dit qu'un langage est régulier si il peut s'écrire sous la forme d'une expression régulière.

Il sera donc préférable de travailler avec des langages réguliers.

1.4 Langage Décidable

L'objectif de ce cours est bien entendu de comprendre comment fonctionne un compilateur, pour pouvoir en créer un par nous même. Pour rappel, on doit d'abord bien comprendre les notions de langage et de mot pour pouvoir ensuite déterminer si un ensemble de mots est syntaxiquement corrects lors de la compilation.
Lors de la compilation, il faut d'abord commencer par savoir si un mot traité appartient au langage défini ou pas. Pour des langages finis, l'opération n'est pas compliquée, pour chaque mot il suffit de vérifier si il appartient à un ensemble fini. Pour des langages infinis, l'opération semble plus complexe, il va falloir trouver une manière systématique et efficace de définir si un mot appartient au langage ou pas.
On appelle ce genre de problème un problème de **décision**.

Définition (**Langage Décidable**) . Un langage L est dit décidable si il existe un algorithme permettant de dire si un mot w appartient ou pas au langage L.

Théorème (**Nombre de Langages Décidables**) . Il existe un nombre fini de langages décidables.

Autrement dit, il existe un nombre infini de langages non décidables...