Chapitre 1

Modélisation syntaxique

Ce chapitre aborde de manière pratique et pédagogique les éléments fondamentaux de la modélisation syntaxique. Il constitue un premier pas vers les techniques de compilation. Le but est de sensibiliser le lecteur à l'importance et à l'utilité de la modélisation syntaxique. D'autres notations seront ajoutées au fur et à mesure dans les chapitres suivants. Plusieurs exercices issus de la théorie des langages seront traités dans ce chapitre (tout comme dans les autres chapitres) pour familiariser le lecteur aux notions de langages formels. Les notions acquises dans le cours de théorie des langages sont nécessaires, voire même indispensables pour aborder aisément les exercices proposés dans ce chapitre.

1 Notions fondamentales

1.1 Alphabet

Un *alphabet* ou un *vocabulaire* est un ensemble fini non vide de caractères ou de symboles.

Un alphabet est un ensemble de caractères à base desquels on peut former des ensembles de mots qui eux aussi peuvent jouer le rôle de vocabulaires. Ces derniers peuvent donc, à leur tour, être utilisés pour construire par exemple des phrases. Dans un langage de programmation, les phrases peuvent être des instructions.

Par exemple, les ensembles suivants représentent des vocabulaires :

L'ensemble des lettres du latin noté V = {a, b, c …, z, A, B, C…, Z} est considéré comme l'alphabet à base duquel on peut former des mots pour les langues naturelles comme l'Anglais, le Français, l'Espagnol, etc.

L'ensemble C = {+, -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9} est le vocabulaire permettant de former des mots qui représentent des entiers comme +128, -32, 765, etc.

L'ensemble D = {0, 1} est l'alphabet de base des nombres binaires comme 0, 1, 00, 01, 10, 11, 0000, 010, etc.

Si l'ensemble V = {a, b, c …, z, A, B, C…, Z} est considéré comme l'alphabet pour former les mots des langues naturelles, il peut également être utilisé pour former des mots clés ou réservés d'un langage de programmation comme C, Pascal, etc. Ces ensembles de mots clés comme {begin, end, if, then, else,…} sont ensuite utilisés comme des vocabulaires pour coder des instructions selon la syntaxe du langage de programmation considéré.

1.2 Chaîne ou mot

Une *chaîne (*ou *mot)* sur un alphabet V est une séquence finie éventuellement vide d'éléments de V.

Par exemple, les séquences mises entre quotes suivantes correspondent à des chaines de caractères.

'ABCD', 'centrale', 'sensitive', 'samedi', '-2300', '178£', 'WR12', etc.

Pour rappel, un mot est pris ici au sens quantitatif et non au sens qualitatif. On ne s'intéresse pas à l'aspect sémantique d'un mot. Autrement dit, un mot ici est regardé du point de vue de sa composition en caractères et non du point de vue de sa signification.

1.3 Longueur d'un mot ou d'une chaine de caractères

La *longueur d'un mot* est égale au nombre de caractères qui constituent le mot.

La longueur du mot noté M est, par convention, désignée par |M|.

Par exemple, les longueurs des chaines de caractères 'conventionnellement' et 'A234' sont respectivement : |conventionnellement| = 19 et |A234 | = 4.

Désormais, on parlera indifféremment d'une chaine de caractères pour désigner un mot, et vice-versa.

1.4 Mot ou chaine vide

On appelle *mot (ou chaine) vide*, une séquence de longueur nulle. En d'autres termes, un mot vide ne comporte aucun caractère.

Par convention, on note le mot vide par le symbole ε (epsilon).
En conséquence, la longueur du mot ε est égale à zéro (|ε|=0).
Ne pas confondre la chaine vide avec l'espace blanc. Comme tout caractère, l'espace blanc est un caractère représentant une chaine de longueur = 1 (| | = 1).

Remarque 1.1

L'opération fondamentale de la formation des mots ou des chaines de caractères, moyennant un alphabet, s'appelle la *concaténation*. En partant de ε (de la chaine vide), on introduit le premier caractère, ensuite par juxtaposition, on ajoute les caractères suivants jusqu'à ce qu'on ait obtenu le mot envisagé.

Par exemple, la formation du mot 'source' nécessite de concaténer d'abord 's' à la chaine vide (ε) ; ce qui donne un premier résultat intermédiaire 's'. Ensuite, ajouter par concaténation 'o' au résultat précédent ; ce qui donne 'so'. Ainsi de suite, jusqu'à former le mot 'source' en entier.

1.5 Définition formelle d'un ensemble de mots

Soit V un alphabet de base.

On note V^+, l'ensemble de tous les mots de longueur $\neq 0$ construits à partir des éléments de l'alphabet V. Mais, comme les mots de V^+ sont de longueur quelconque (mais $\neq 0$), on peut alors formuler V^+ comme suit :

$V^+ = V^1 \cup V^2 \ldots \cup V^n \ldots = \bigcup_{i \geq 1} V^i$.

On note V^*, l'ensemble de tous les mots formés à partir des éléments de V, auquel on ajoute le mot vide. Autrement dit, les mots de longueur nulle en font partie, c'est-à-dire que $V^* = \bigcup_{i \geq 1} V^i \cup \{\varepsilon\} = \bigcup_{i \geq 0} V^i$.

Par voie de conséquence, on déduit que $V^* = V^+ \cup \{\varepsilon\}$ et $V^+ = V^* - \{\varepsilon\}$.

Ainsi par exemple :

Avec le singleton $V_1 = \{b\}$, on peut construire l'ensemble V_1^* de tous les mots de longueur quelconque, représenté par $V_1^0 \cup V_1^1 \ldots \cup V_1^n \ldots = \{b\}^0 \cup \{b\}^1 \ldots \cup \{b\}^n \ldots$ c'est-à-dire $V_1^* = \{\varepsilon, b, bb, bbb, bbbb, \ldots, b^n, \ldots\} = \{b^n \mid n \geq 0\}$ qui est un ensemble infini dénombrable. Pour rappel, un ensemble est dénombrable lorsque ses éléments peuvent être énumérés sans omission ni répétition dans une liste indexée par des entiers.

Avec le vocabulaire $V_2 = \{a, b\}$, l'ensemble V_2^* est composé de toutes les chaines constituées des lettres *a* et *b*, y compris la chaine vide ε. Cet ensemble est également infini dénombrable. La liste de ses éléments est : ε, *a, b, aa, ab, ba, bb, aaa, aab, aba, baa,* …,$\{a, b\}^n$, …, avec $n \geq 0$. $V_2^* = \{ \{a, b\}^n \mid n \geq 0\}$.

Si l'on remplace les lettres *a* et *b*, respectivement par 0 et 1, et si on ignore la chaine vide on peut construire l'ensemble des chaines représentant l'ensemble B des nombres binaires, $B = \{ \{0, 1\}^n \mid n \geq 1\} = \{0, 1, 00, 01, 10, 11, 000, 001,\ldots\}$.

1.6 Concaténation de mots

Soient x et y deux mots dans l'ensemble V^*. On appelle *concaténation* de x et y dans V*, la juxtaposition de x et y. On note alors la concaténation des mots x et y par xy (ou x.y).

En fait, mathématiquement, l'ensemble V^* est le monoïde libre engendré par l'alphabet de base V (l'ensemble V est appelé aussi base de V^*).

La concaténation étant une loi de composition interne sur V^*. Donc, quels que soient les mots x et $y \in V^*$, leur juxtaposition donne lieu toujours à un mot appartenant à V^*, c'est-à-dire que $z = xy$ (ou $x.y) \in V^*$.

La concaténation est associative, car quels que soient les mots x, y et $z \in V^*$, on a toujours $(x.y).z = x.(y.z) = x.y.z$.

L'élément neutre pour le monoïde V^* est la chaîne vide ε , puisque quel que soit le mot $x \in V^*$, on vérifie toujours que $x\varepsilon = \varepsilon x = x$.

Notons que l'élément neutre ne fait pas partie de l'alphabet de base, sinon un mot aurait une infinité de décompositions possibles.

Enfin, la concaténation, comme on peut facilement le constater, n'est pas commutative. En effet, le contre-exemple x = *abc* et y = *baba*, montre que la chaîne xy = *abcbaba,* est bien différente de la chaîne yx = *babaabc*.

Remarque 1.2

Avant de clore ce premier volet, il convient d'ajouter quelques exemples illustratifs relatifs à la notion de vocabulaire tel que défini ci-dessus. L'idée est de s'approcher progressivement de la notion de langage. En effet, la définition de certains langages se limite à la construction d'ensembles de mots qui satisfont certaines conditions. On fait ici allusion à des langages très simples qui se résument à des ensembles de mots.

Par exemple :

- Formulation de l'ensemble des chaines représentant des entiers naturels.

Le vocabulaire de base est défini par l'ensemble $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

En appliquant l'opération de concaténation, on peut construire l'ensemble des mots représentant les entiers naturels.

$E = D \cup D^2 \cup \ldots \cup D^n \ldots = \{0,1,\ldots 9\} \cup \{0,1,\ldots 9\}^2 \ldots \cup D^n \ldots = \{0, 1, 2..9, 00, 01,\ldots 09, 10, 11,\ldots 19,\ldots,000, 001, 002,\ldots 009,\ldots\} = \{D^n \mid n \geq 1\}$ est un sur-ensemble de l'ensemble des mots qui représentent l'ensemble N des entiers naturels. En d'autres termes, ce dernier est un sous ensemble de E.

On verra sous peu quels sont les outils et techniques qui permettent de générer ce genre d'ensemble.

- Formulation de l'ensemble des chaines qui représentent des entiers naturels pairs.

On part de l'hypothèse que les nombres de cette catégorie se terminent par un chiffre pair {0, 2, 4, 6, 8}. Donc, ça doit être un sous-ensemble de l'ensemble E précédent.
On peut donc le formuler par $P = \{0, 2, 4, 6, 8\} \cup D.\{0, 2, 4, 6, 8\} \ldots \cup D^n.\{0, 2,\ldots 8\} \ldots$
On voit bien ici que la concaténation est appliquée aux ensembles de mots, comme par exemple la sous-expression {0, 2, 4, 6, 8} ∪ D.{0, 2, 4, 6, 8} de P qui produit un sous ensemble dont les éléments sont répertoriés dans le *Tableau 1*.

0	2	4	6	8					
00	10	20	30	40	50	60	70	80	90
02	12	22	32	42	52	62	72	82	92
04	14	24	34	44	54	64	74	84	94
06	16	26	36	46	56	66	76	86	96
08	18	28	38	48	58	68	78	88	98

Tableau 1 *: Nombres entiers pairs à deux positions au maximum*

Cette table contient tous les entiers naturels pairs à une ou deux positions, mais avec une redondance des nombres 0, 2, 4, 6, 8 (en première colonne). En effet, quand ces derniers sont codés sur deux positions, ils sont représentés par 00, 02, 04, 06, 08.

Mathématiquement, cette redondance ne devrait pas faire partie de l'ensemble des entiers naturels pairs, puisque ce dernier est dénombrable et n'admet ni répétition d'éléments, ni éléments superflus.

En informatique, il y a deux solutions à considérer :

Soit on opte pour la solution du *Tableau 1* et laisser le compilateur s'occuper du reste. Dans ce cas, ce n'est pas difficile de déduire que les 0 non significatifs (comme dans les nombres 0023, 023, 00023, …) sont tout simplement ignorés dans le calcul de la valeur du nombre. Les nombres 0023, 023, 00023 ne représentent qu'un seul nombre à savoir 23. Cela peut se vérifier facilement dans un programme en Pascal, C ou autre.
D'autre part, il faut noter aussi que même si on peut avoir un sur-ensemble, comme c'est le cas des deux exemples présentés ci-dessus, les limites physiques des machines feront qu'il n'y aura qu'un nombre fini de nombres entiers. En d'autres termes, la notion de sur-ensemble n'a de sens que dans le cas de la représentation formelle de ces nombres.

Soit on empêche les redondances de 0 non significatifs d'avoir lieu. Dans ce cas, la solution est une formalisation de l'expression du calcul de l'ensemble des chaines qui représentent des entiers pairs. Il suffit alors d'éliminer le caractère 0 à un certain niveau dans l'expression $P = \{0, 2, 4, 6, 8\} \cup D.\{0, 2, 4, 6, 8\} \ldots \cup D^n.\{0, 2,\ldots 8\} \ldots$Ce qui devient alors $P_0 = \{0, 2, 4, 6, 8\} \cup D'.\{0, 2, 4, 6, 8\} \ldots \cup D'^n.\{0, 2,\ldots 8\} \ldots$, sachant que $D' = D - \{0\}$. La sous-expression produit donc les éléments qui sont répertoriés dans le

Tableau 2. Cette solution donne exactement tous les entiers naturels pairs à une ou deux positions, sans redondance.

Reste à savoir si cette solution est intéressante. La solution précédente, en tout cas, est intéressante, car elle laisse la liberté à l'utilisateur d'écrire ses nombres comme il l'entend, sans aucune contrainte. D'ailleurs, c'est l'approche qui est adoptée dans quasiment tous les compilateurs actuels en raison de sa facilité d'implémentation et des avantages qu'elle offre en permettant une bonne marge de manœuvre à l'utilisateur.

0	2	4	6	8				
10	20	30	40	50	60	70	80	90
12	22	32	42	52	62	72	82	92
14	24	34	44	54	64	74	84	94
16	26	36	46	56	66	76	86	96
18	28	38	48	58	68	78	88	98

***Tableau 2** : Entiers pairs à une ou deux positions sans redondance*

La deuxième solution, quant à elle, est réalisable mais moyennant un petit ajustement de la part du compilateur qui va devoir empêcher qu'un entier puisse admettre des 0 non significatifs. Un test, à la rencontre d'un 0 comme premier caractère d'un nombre, permet de gérer la situation. Cette solution est évidemment moins intéressante que la première.

A présent, on va passer à la notion de langage formel qui est très importante dans le contexte de la modélisation syntaxique. Les vocabulaires, comme on vient de le voir, sont fondamentaux, dans le concept des langages formels. Mais, il va falloir également définir les outils et techniques nécessaires à la formulation et la génération de ces langages. C'est l'objet des différentes sections suivantes.

2 Langages formels

2.1 Notion de langage

Un *langage* est un ensemble de mots construits à base d'un alphabet. Plus précisément, on appelle langage L sur un alphabet V, un sous-ensemble de mots *de* V^* ($L \subseteq V^*$).

Par exemple, soient V = {0, 1, 2, 3, …, 9}, le vocabulaire de base et $V^* = \{0, 1, 2, 3, \ldots 9\}^*$ l'ensemble de toutes les combinaisons des mots formés à partir de V. Ces derniers, à l'exception de la chaine vide (ε), représentent des entiers naturels.

Les langages L_i suivants sont inclus dans V^* ; ($L_i \subseteq V^*$) :

- L_0 est le langage des chaines représentant des nombres entiers naturels multiples de 10. L_0 est donc un sous-ensemble de V^*. $L_0 = \{\{1, 2, 3, \ldots 9\}^+ 0\}$.

Si l'on considère que les multiples de 10 sont ≥ 10, alors L_0 correspond bien à l'expression L_0 présentée ci-dessus.

Mais, si l'on parle de nombres entiers multiples de 10 (ou divisibles par 10) qui incluent le 0, on le désigne par L_1, et on écrit $L_1 = \{\{1, 2, 3, \ldots, 9\}^*.0\}$.

Dans le 1er cas on a : 10, 20, …90, 100, 110,…, 950, 1000,…, …, 10500,…, etc. Les nombres sont au minimum sur 2 positions, l'essentiel qu'ils se terminent par 0 pour qu'ils soient multiples de 10.

Dans le 2eme cas, on introduit le 0. Ce qui donne la même série de nombres à laquelle on ajoute 0. On a en tout cas $L_1 = L_0 \cup \{0\}$.

- L_2 est le langage de toutes les chaînes de V^* représentant des nombres entiers divisibles par 5 de longueur vérifiant $1 \leq$ longueur ≤ 2. Ce langage est symboliquement noté V_5^{+2}. L'indice 5 pour la divisibilité par 5, et la puissance $^{+2}$ pour la condition $1 \leq$ longueur ≤ 2. Les nombres seront codés sur 1 position (pour 0 et 5) et 2 positions pour les autres jusqu'à 95.

Les nombres divisibles par 5 se terminent par 0 ou 5. Donc, $V_5^{+2} = \{1, 2,\ldots, 9\}^{*1}.\{0, 5\}$.

Dans *Tableau 3* sont récapitulés les éléments de l'ensemble $V_5^{+2} = L_2$.

0	5	10	15	20	25	30	35
40	45	50	55	60	65	70	75
80	85	90	95				

Tableau 3 *: Multiples de 5 à deux positions au maximum*

Remarque 2.1

Dans les exemples ci-dessus, on a introduit de nouvelles notations pour exprimer certaines situations. On a utilisé la notation V_5^{+2} où la puissance $^{+2}$ indique que la longueur des mots de $L_2 = V_5^{+2}$ est au moins égale à 1 et au plus égale à 2 ($1 \leq$ longueur ≤ 2).

On peut généraliser en disant que :

La longueur $|V^{*n}|$ des éléments de V^{*n} vérifie $0 \leq |V^{*n}| \leq n$.

La longueur $|V^{+n}|$ des éléments de V^{+n} vérifie $1 \leq |V^{+n}| \leq n$.

Avec $V_5^{*2} = \{1, 2,\ldots, 9\}^{*1}.\{0, 5\}$, les éléments de $\{1, 2,\ldots, 9\}^{*1}$ sont au plus sur 1 position. Donc, ces éléments concaténés avec $\{0, 5\}$, devraient donner des éléments de longueur $1 \leq$ longueur ≤ 2. Le résultat est donc $V_5^{+2} = \{0, 5, 10, 20, 30, 40, \ldots, 95\}$.
En effet, en s'inspirant des formules vues plus haut, l'ensemble $\{1, 2,\ldots, 9\}^{*1} =$
$\{1, 2, 3, \ldots 9\}^0 \cup \{1, 2, 3, \ldots 9\}^1 = \{\varepsilon\} \cup \{1, 2, 3, \ldots 9\}$.
Donc, en concaténant $\{0, 5\}$ à cet ensemble, on aura $(\{\varepsilon\} \cup \{1, 2, 3, \ldots 9\}).\{0,5\}$. Ensuite, en développant les calculs, on obtient l'ensemble $\{0,5\} \cup \{10, 15, 20, 25, \ldots 90, 95\}$ qui correspond exactement aux éléments du *Tableau 3*.

L'opération de concaténation, définie, à la base, pour former des mots, peut être étendue aisément aux ensembles de mots, donc aux langages. On définira dans ce qui suit, la concaténation ainsi que toutes les opérations réalisables sur les langages formels.

2.2 Opérations usuelles sur les langages formels

Etant donnés deux langage L_1 et L_2 sur un alphabet V, c'est-à-dire que L_1 et L_2 sont inclus dans V^* ($L_1 \subseteq V^*$et $L_2 \subseteq V^*$).

- *Concaténation*

La concaténation de L_1 et L_2 est définie par $L_1.L_2 = \{xy \in V^* \mid x \in L_1 \text{ et } y \in L_2\}$.
Par exemple, $L_1 = \{HT, BT\}$ et $L_2 = \{00, 01, 10, 11\}$.
$L_1.L_2 = \{HT00, HT01, HT10, HT11, BT00, BT01, BT10, BT11\}$.

- *Union*

On définit l'union de deux langages L_1 et L_2 par $L_1 \cup L_2 = \{x \in V^* \mid x \in L_1 \text{ ou } x \in L_2\}$.

Par exemple, $L_1 = \{000, 010, 111\}$ et $L_2 = \{111, 010, 101, 100, 110\}$.
$L_1 \cup L_2 = \{000, 010, 111, 101, 100, 110\}$.

- *Intersection*

On définit l'intersection de L_1 et L_2 par $L_1 \cap L_2 = \{x \in V^* \mid x \in L_1 \text{ et } x \in L_2\}$.
Par exemple, $L_1 = \{\text{begin, end, if, then, else}\}$ et $L_2 = \{\text{repeat, until, begin, end}\}$.
$L_1 \cap L_2 = \{\text{begin, end}\}$.

- *Reflet ou effet miroir*

On définit l'effet (image) miroir du langage L par $L^R = \{x \mid x = y^R \text{ avec } y \in L\}$; y^R est le mot miroir de y, ou l'image miroir de y, c'est-à-dire que si $y = a_1 \ldots a_n$, alors $y^R = a_n \ldots a_1$.
Par exemple, l'image miroir du mot y ='exercice' est $x = y^R$ = 'ecicrexe'.
Les images miroirs des mots 'elle', 'ici' et 'ere' sont respectivement 'elle', 'ici' et 'ere', c'est-à-dire que les mots miroirs sont égaux à leurs antécédents respectifs.
Lorsqu'un mot est égal à son image miroir, il est appelé mot palindrome.

- *Complémentaire*

On définit le complémentaire d'un langage L sur V par $L_C = \{x \in V^* \mid x \notin L\}$.
Par exemple, le langage des nombres binaires pairs noté L_P est le complémentaire du langage des nombres binaires impairs L_I dans le système de numération binaire pur.
$L_P = \{0, 1\}^*.0$ est l'ensemble de toutes les chaines constituées des caractères 0 ou 1 se terminant par 0. Donc, cela correspond nécessairement à l'ensemble des nombres binaires pairs.
$L_I = \{0, 1\}^*.1$ est l'ensemble de toutes les chaines constituées de 0 ou 1, se terminant par 1. Donc, cela correspond nécessairement à l'ensemble des nombres binaires impairs.
Aperçu sur $L_P = \{0, 00, 10, 000, 010, 100, 110, \ldots, \{0, 1\}^n.0\ldots\}$.
Aperçu sur $L_I = \{1, 01, 11, 001, 011, 101, 111, \ldots, \{0, 1\}^n.1\ldots\}$.
$L_I \cup L_P = \{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101,\ldots, \{0, 1\}^n\{0,1\}\ldots\} = \{0,1\}^+$ qui est exactement l'ensemble de tous les nombres binaires.

- *Différence*

La différence entre deux langages L_1 et L_2 est l'ensemble noté $L_1 - L_2$ composé des mots qui sont dans L_1 mais qui ne sont pas dans L_2. $L_1 - L_2 = \{x \in V^* \mid x \in L_1 \text{ et } x \notin L_2\}$.
Par exemple, si L_1 est l'ensemble des nombres binaires divisibles par 2 (pairs) et L celui de tous les nombres binaires, alors en appliquant la formule de la différence $L - L_1$, on obtient $L - L_1 = \{0,1\}^+ - \{0,1\}^*.0 = \{0,1\}^*.(\{0,1\} - \{0\}) = \{0,1\}^* (\{1\}) = \{0,1\}^*.1$ qui correspond exactement à l'ensemble des nombres binaires impairs.

A présent, on va présenter l'itération d'un langage. Cette dernière est analogue à l'opération de construction des ensembles des mots construits à partir d'un alphabet.

- *Itération et étoile d'un langage*

L'itération d'un langage L est définie par $L^i = L.L^{i-1}$ avec $i \geq 1$. Par convention, on pose $L^0 = \{\varepsilon\}$.
Par conséquent, l'étoile d'un langage s'écrit $L^* = \bigcup_{i \geq 0} L^i = L^0 \cup L^1 \ldots$

$L^+ = L^1 \cup L^2 \cup \ldots = \bigcup_{i \geq 1} L^i$ alors $L^* = \{\varepsilon\} \cup L^+ = L^0 \cup L^+$ et donc $L^+ = L\,L^* = L^*\,L$.

Par exemple, soit $L = \{00, 01, 10, 11\}$
$L^2 = LL^1 = LL = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$.
Le langage L représente les entiers de 0 à 3 en binaire codés sur 2 positions.

Le langage L^2 représente les entiers de 0 à 15 en binaire codés sur 4 positions.
Le langage L^i représente les entiers de 0 à $2^i - 1$ en binaire codés sur i positions.

L'étoile $L^* = \bigcup_{i \geq 0} L^i = L^0 \cup L^1 + \ldots = \{\varepsilon\} \cup \{00, 01, 10, 11\} \cup \ldots L^i \ldots \cup \ldots$

L^* est l'ensemble constitué de toutes les chaines binaires codées respectivement sur 2, 3, 4,.. 8,…, …positions, et de la chaine vide {ε}. C'est un ensemble infini dénombrable. Le *Tableau 4* donne quelques séquences, mais non exhaustives de ces chaines binaires.

nombre entiers	nombres correspondants en binaire
0, 1, 2, 3	00, 01, 10, 11
0 - 7	000, 001, 010,…111
0 - 15	0000, 0101, 0110,…, 1000, 1001, 1010, 1011,…1100, 1101, 1110, 1111
$0 - (2^i\text{-}1)$ …	0000000…000, 0000000…001,0000000…010…… 00000000…111
	0000001…000, 0000001…001,00000001…010……
…….	……. …….. …….

Tableau 4 *: Nombres entiers naturels et leurs représentations binaires associées*

2.3 Modes d'utilisation d'un langage

En général, un langage peut être utilisé de deux manières différentes :

- En mode *générateur*

Le langage peut être décrit par une grammaire basée sur des règles syntaxiques. L'application de ces dernières permet de générer le langage en question.

- En mode *accepteur*

Le mode accepteur sous-entend 'analyseur'. Cette analyse est généralement assurée par des outils comme les automates qu'on nomme également 'accepteurs'.

Dans le premier cas, étant donné un flot d'entrée basé à priori sur la syntaxe d'un langage, l'application des règles syntaxiques permet de générer une séquence de mots qui sera comparée au flot présenté en entrée.

Dans le deuxième cas, étant donné un flot d'entrée basé à priori sur la syntaxe d'un langage, un automate a pour rôle d'approuver (accepter) ou de désapprouver (rejeter) le flot présenté en entrée.

Quand on manipule manuellement une grammaire ou un automate, on ne voit pas de différence. La seule différence qu'il y a, concerne uniquement leurs configurations ou leurs descriptions. On pourra en dire plus lorsqu'on étudiera amplement ces deux outils.

2.4 Grammaire formelle

Une *grammaire formelle* est définie par le quadruplet $G = (V_N, V_T, S, P)$ dans lequel :

- V_T est un vocabulaire terminal (alphabet de base) qui consiste en un ensemble fini non vide ($\neq\varnothing$) de symboles terminaux.