

Structure algébrique des langages

- ▶ Alphabet : Ensemble fini de symboles $A = \{s_i \mid i \in [1..n], n \in \mathbb{N}\}$
- ▶ Exemples :

Binaire : $\{0, 1\}$

Décimal : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Hexadécimal : $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Minuscules Romaine : $\left\{ \begin{array}{l} a, b, c, d, e, f, g, h, i, j, k, l, \\ m, n, o, p, q, r, s, t, u, v, w, x, y, z \end{array} \right\}$

Couleurs : $\{\blacksquare, \blacksquare, \blacksquare, \blacksquare, \blacksquare, \blacksquare, \blacksquare, \blacksquare\}$

Notes de musique : $\{\text{G}, \text{B}, \text{G}, \text{o}, \text{d}, \text{d}, \text{n}, \text{n}, \dots\}$

Figures : $\{\clubsuit, \heartsuit, \diamondsuit, \spadesuit\}$

- ▶ Mot : Séquence de symbole $s_1 s_2 \dots s_n$

- ▶ Exemples :

Nombres : 123

- ▶ Mot vide : Λ

- ▶ Langage : Ensemble de mots (Langage vide : $\emptyset = \{\} \right)$

Codage sur 3 bits : $\{000, 001, 011, 010, 110, 100, 101, 111\}$



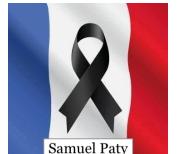
Opération de concaténation : \bullet

- ▶ Opération de concaténation, juxtaposition, séquencement, . . . :
 $\bullet : \text{Mot} \times \text{Mot} \mapsto \text{Mot}$
- ▶ $(s_1 \dots s_p) \bullet (s_{p+1} \dots s_n) = s_1 \dots s_n$
- ▶ $s_1 s_2 \dots s_n = (\dots (s_1 \bullet s_2) \bullet \dots) \bullet s_n$
- ▶ Associatif :
 $(\dots (s_1 \bullet s_2) \bullet \dots) \bullet s_n = s_1 \bullet s_2 \bullet \dots \bullet s_n = s_1 \bullet (\dots \bullet (s_{n-1} \bullet s_n) \dots)$
- ▶ Elément neutre : Suite vide Λ
- ▶ Extension à des langages :
 $X \bullet Y = \{x \bullet y \mid x \in X, y \in Y\}$
- ▶ Exemple : $\{a, bc\} \bullet \{de, f\} = \{ade, af, bcde, bcf\}$
- ▶ d'où l'opération X^n définie par :
 $X^0 = \{\Lambda\}$ et $X^{n+1} = X \bullet X^n$
- ▶ Exemple :
 $\{a, bc\}^3 = \{aaa, aabc, abca, bcaa, abcbc, bcabc, bcbca, bcbc\}$



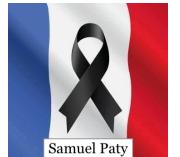
Etoile de Kleene

- ▶ Objectif : Construire l'ensemble de tous les mots possibles
- ▶ Mot vide (0 symbole) : $A^0 = \{\Lambda\}$ ($\neq \emptyset = \{\}$ langage vide)
- ▶ Mots d'1 symbole : $A^1 = A$
- ▶ Mots de 2 symboles : $A^2 = A \bullet A$
- ▶ Mots de n symboles : A^n
- ▶ d'où $A^* = \bigcup_{i=0}^{\infty} A^i$
- ▶ Un langage sur A est un sous-ensemble de A^* .
- ▶ L'ensemble des langages sur A est $\mathcal{P}(A^*)$ (donc \aleph_1 langages possibles).
- ▶ \Rightarrow Structure algébrique de A^* muni de \bullet : Monoïde libre engendré par A .



Opérateurs sur les langages

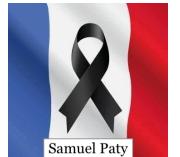
$$\begin{aligned}\overline{L} &= \{m \in A^* \mid m \notin L\} \\ L_1 \cup L_2 &= \{m \in A^* \mid m \in L_1 \vee m \in L_2\} \\ L_1 \cap L_2 &= \{m \in A^* \mid m \in L_1 \wedge m \in L_2\} = \overline{\overline{L_1} \cup \overline{L_2}} \\ L_1 \setminus L_2 &= \{m \in A^* \mid m \in L_1 \wedge m \notin L_2\} = L_1 \cap \overline{L_2} \\ L_1 \bullet L_2 &= \{m_1 \bullet m_2 \in A^* \mid m_1 \in L_1 \wedge m_2 \in L_2\} \\ L^0 &= \{\Lambda\} \neq \emptyset = \{\} \\ L^{n+1} &= L \bullet L^n \\ L^* &= \bigcup_{i=0}^{\infty} L^i \\ L^+ &= \bigcup_{i=1}^{\infty} L^i = L \bullet L^* = L^* \setminus \{\Lambda\} \text{ si } \Lambda \notin L\end{aligned}$$



Expressions régulières

- ▶ Objectif : Description formelle de langages (langage de description de langages)
- ▶ Moyen : Langage vide, mot vide, symboles de l'alphabet, opérateurs sur les langages
- ▶ Constantes :
Signification

Notation
s
$[s_m - s_{m+n}]$
Λ
\emptyset



Expressions régulières

- ▶ Opérateurs :

Signification

Notation

notation alternative

Concaténation de langages

$e_1 e_2$

$e_1 \bullet e_2$

Choix entre deux langages

$e_1 | e_2$

$e_1 + e_2$

Option

$e?$

Répétition d'un langage

e^*

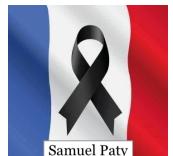
e^+

$e\{n_1, n_2\}$

Complément d'un langage

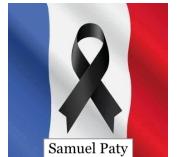
$[^s_m - s_{m+n}]$

- ▶ Les parenthèses permettent d'imposer priorité et associativité
- ▶ Les ' (apostrophes), " (guillemets) et \ banalisent les opérateurs
- ▶ Définition : identificateur = e
- ▶ Utilisation : {identificateur}
- ▶ **Attention** : Une définition ne doit pas être récursive (cela devient une grammaire EBNF)



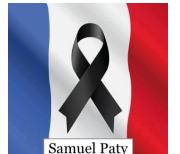
Exemples

- ▶ Nombres entiers naturels : `nombre=[0-9]+`
- ▶ Notation usuelle : `entier=([1-9][0-9]*)|0`
- ▶ Nombres entiers relatifs : `relatif=(+'|"-')?{nombre}`
- ▶ Nombres décimaux :
`decimal={relatif}(\.{nombre})?([eE]{relatif})?`



Langage associé

$$\begin{aligned} L(\emptyset) &= \{\} \\ L(\Lambda) &= \{\Lambda\} \\ L(s) &= \{s\} \\ L(e_1 \mid e_2) &= L(e_1) \cup L(e_2) \\ L(e_1 e_2) &= L(e_1) \bullet L(e_2) \\ L(e\star) &= L(e)^\star = \bigcup_{i=0}^{\infty} L(e)^i \\ L(e+) &= L(e) \bullet L(e)^\star = \bigcup_{i=1}^{\infty} L(e)^i \\ L(e?) &= \{\Lambda\} \cup L(e) \\ L(e\{n_1, n_2\}) &= \bigcup_{i=n_1}^{n_2} L(e)^i \\ L([s_m - s_{m+n}]) &= \{s_i\}_{i \in [m, \dots, m+n]} \\ L([\hat{s}_m - s_{m+n}]) &= A \setminus \{s_i\}_{i \in [m, \dots, m+n]} \end{aligned}$$



Propriétés des opérateurs

$$\emptyset e = e \emptyset = \emptyset$$

$$e | \emptyset = \emptyset | e = e$$

$$e_1 (e_2 e_3) = (e_1 e_2) e_3$$

$$e_1 (e_2 | e_3) = (e_1 e_2) | (e_1 e_3)$$

$$e_1 | e_2 = e_2 | e_1$$

$$e\star = \Lambda | e+$$

$$e\star e\star = e\star$$

$$e = e\star \Leftrightarrow e = e e$$

$$(e_1 \star e_2 \star) \star = (e_1 | e_2) \star = (e_1 \star | e_2 \star) \star$$

$$(e_1 \star e_2) \star (e_1 \star) = (e_1 | e_2) \star = e_1 \star (e_2 (e_1 \star)) \star$$

$$\Lambda e = e \Lambda = e$$

$$e | e = e$$

$$e_1 | (e_2 | e_3) = (e_1 | e_2) | e_3$$

$$(e_1 | e_2) e_3 = (e_1 e_2) | (e_1 e_3)$$

$$\emptyset \star = \Lambda \star = \Lambda$$

$$e+ = e \quad e\star = e \star \quad e$$

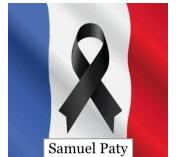
$$e\star\star = e\star$$

$$e e\star = e\star \Leftrightarrow \Lambda \in L(e)$$



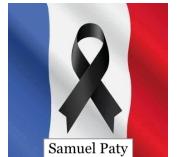
Limites des expressions régulières

- ▶ Nombres entiers naturels : `nombre=[0-9]+`
- ▶ Nombres entiers relatifs : `relatif=('+' | '-')?{nombre}`
- ▶ Nombres décimaux :
`decimal={relatif}(\.{nombre})?([eE]{relatif})?`
- ▶ Expressions arithmétiques non parenthésées :
`expression={decimal}((\"+\" | ' - ' | \"*\" | ' / ') {decimal})*`
- ▶ Expressions arithmétiques parenthésées :
`expression='(*{decimal})*'*((+' | '-' | '*' | '/')'(*{decimal})*')*`
- ▶ **Limites** : Expressions bien parenthésées (association des parenthèses – langages de Dick)



Grammaire

- ▶ Quadruplet (A, V, S, P) tel que :
 - ▶ A : ens. fini symboles terminaux (alphabet)
 - ▶ V : ens. fini symboles non-terminaux ($A \cap V = \emptyset$)
 - ▶ $S \in V$: axiome
 - ▶ $P \subseteq ((A \cup V)^* \bullet V \bullet (A \cup V)^*) \times (A \cup V)^*$: ensemble fini de règles de production notées $\alpha \rightarrow \beta$ (β est la production de α)
- ▶ Notations :
 - ▶ $a, b, \dots \in A$
 - ▶ $A, B, \dots \in V$
 - ▶ $\alpha, \beta, \dots \in (A \cup V)^*$



Exemple : Expressions Arithmétiques

$$A = \{+ * - n i, () []\}$$

$$V = \{E, L_E, S_E\}$$

$$S = E$$

$$P = \left\{ \begin{array}{l} E \rightarrow E + E \\ E \rightarrow E * E \\ E \rightarrow - E \\ E \rightarrow n \\ E \rightarrow i \\ E \rightarrow i [L_E] \\ E \rightarrow (E) \\ L_E \rightarrow E S_E \\ S_E \rightarrow, E S_E \\ S_E \rightarrow \Lambda \end{array} \right\}$$

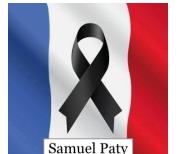


Dérivation

Langage décrit par une grammaire

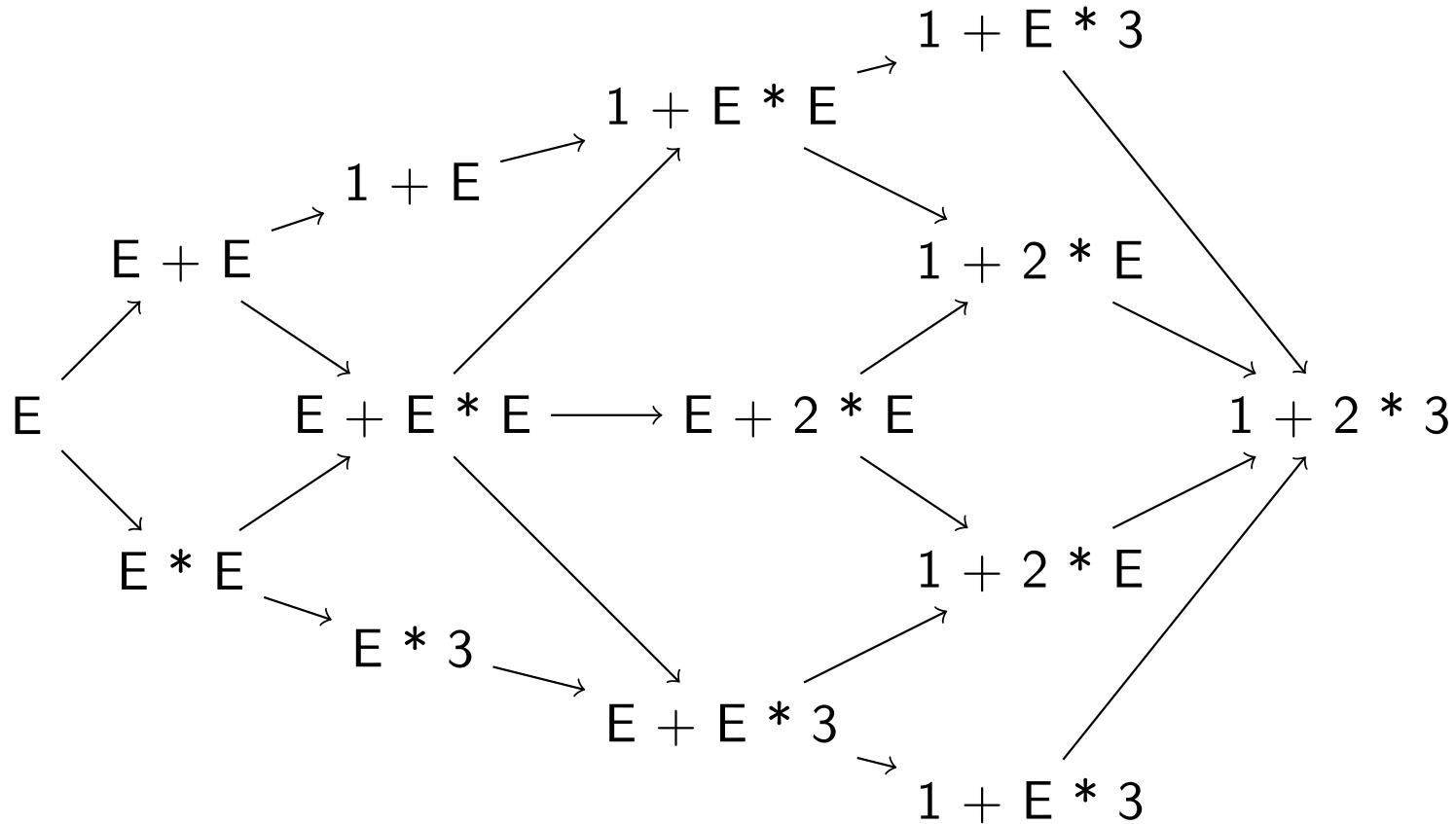
- ▶ Relation $\alpha \Rightarrow \beta$ si et seulement si $\left\{ \begin{array}{l} \alpha = \delta_1 \gamma_1 \delta_2 \\ \beta = \delta_1 \gamma_2 \delta_2 \\ \gamma_1 \rightarrow \gamma_2 \in P \\ \delta_i, \gamma_i \in (A \cup V)^* \end{array} \right.$
- ▶ Fermeture réflexive transitive : $\alpha \Rightarrow^* \beta$
- ▶ α se dérive en β en utilisant P
- ▶ Notation : $\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \dots \Rightarrow \alpha_{n-1} \Rightarrow \beta$
- ▶ $L(G) = \{m \in A^* \mid S \Rightarrow^* m\}$
- ▶ Exemple : Dérivation pour $E \Rightarrow^* 1 + 2 * 3$

$$\begin{array}{ccccccc} E & \xrightarrow{\quad E * E \quad} & \xrightarrow{\quad E + E * E \quad} & \xrightarrow{\quad 1 + E * E \quad} & \xrightarrow{\quad 1 + 2 * E \quad} & \xrightarrow{\quad 1 + 2 * 3 \quad} \\ \alpha = E & \alpha = E * E & \alpha = E + E * E & \alpha = E + E * E & \alpha = 1 + E * E & \alpha = 1 + 2 * E \\ \beta = E * E & \beta = E + E * E & \beta = 1 + E * E & \beta = 1 + E * E & \beta = 1 + 2 * E & \beta = 1 + 2 * 3 \\ \delta_1 = \Lambda & \delta_1 = \Lambda & \delta_1 = \Lambda & \delta_1 = \Lambda & \delta_1 = 1 + & \delta_1 = 1 + 2 * \\ \gamma_1 = E & \gamma_1 = E \\ \delta_2 = \Lambda & \delta_2 = *E & \delta_2 = +E * E & \delta_2 = +E * E & \delta_2 = *E & \delta_2 = \Lambda \\ \gamma_2 = E * E & \gamma_2 = E + E & \gamma_2 = 1 & \gamma_2 = 1 & \gamma_2 = 2 & \gamma_2 = 3 \end{array}$$



Stratégies de dérivation

- ▶ Un même mot peut être engendré par plusieurs dérivations distinctes



- ▶ Analyse déterministe requiert une stratégie lors de la reconnaissance de ce mot

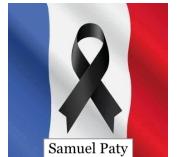
Stratégies de dérivation

- ▶ Dérivation la plus à gauche (Leftmost) : Dérivation du non-terminal le plus à gauche (Analyseur LL : Left to right/Leftmost)

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow 1 + E * E \Rightarrow 1 + 2 * E \Rightarrow 1 + 2 * 3$$

- ▶ Dérivation la plus à droite (Rightmost) : Dérivation du non-terminal le plus à droite (Analyseur LR : Left to right/Rightmost)

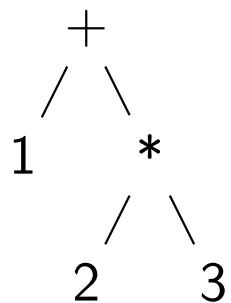
$$E \Rightarrow E * E \Rightarrow E * 3 \Rightarrow E + E * 3 \Rightarrow E + 2 * 3 \Rightarrow 1 + 2 * 3$$



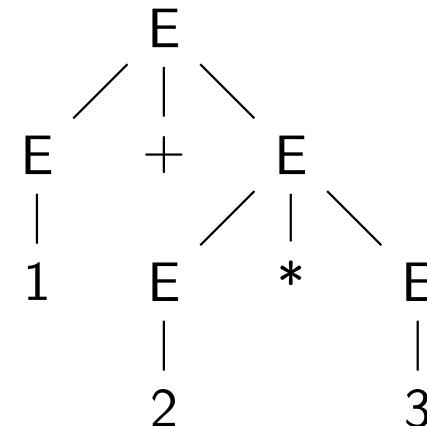
Arbres de dérivation (syntaxique)

- ▶ Objectif : Représenter la structure du mot induite par les règles de production lors d'une dérivation
- ▶ Feuilles de l'arbre : Terminaux composant le mot
- ▶ Racine de l'arbre : Axiome
- ▶ Nœuds de l'arbre : Non-terminaux apparaissant dans la dérivation
- ▶ Branches de l'arbre : Règles de production

Attention : \neq Arbre abstrait

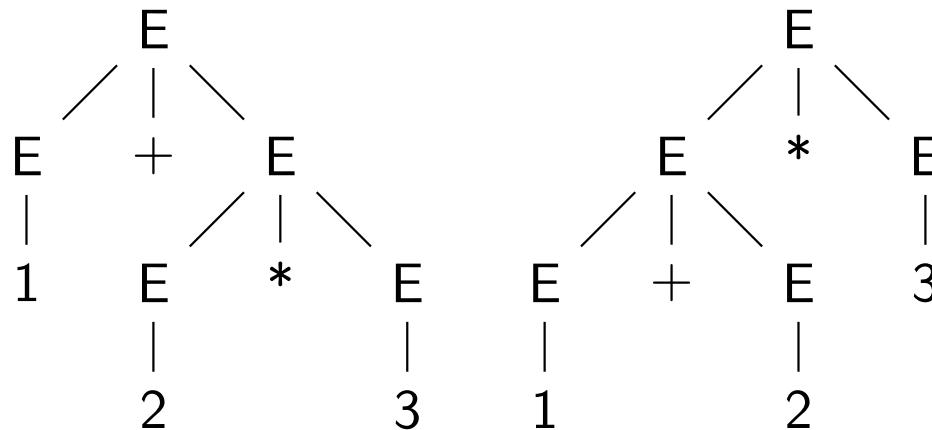


Exemple :
Arbre de dérivation
pour $1 + 2 * 3$

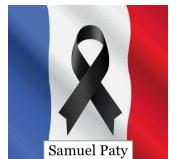


Grammaire et langage ambigu

- ▶ Une grammaire est ambiguë s'il existe plusieurs arbres de dérivation distincts pour un même mot
- ▶ Exemple : Arbres de dérivation pour $1 + 2 * 3$



- ▶ Un langage est ambigu si toutes les grammaires le représentant sont ambiguës

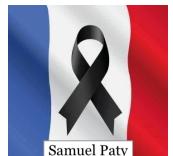


Typologie des grammaires

Travaux de Noam Chomsky

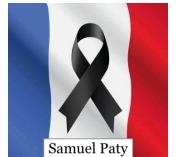
Type	Productions	Nom
0	$\alpha \rightarrow \beta$	
1	$\gamma X \delta \rightarrow \beta$ $\alpha \rightarrow \beta, \alpha < \beta , \alpha \rightarrow \Lambda$	Contextuel Taille croissante
2	$X \rightarrow \beta$	Non contextuel Algébrique
$LR(k)$	$X \rightarrow \beta$ déterministe	
3	$X \rightarrow aY, X \rightarrow \Lambda$ $X \rightarrow Ya, X \rightarrow \Lambda$	Régulier à droite Régulier à gauche

Type	Analyse	Isomorphe
0	Indécidable, Coûteuse	Machine de Turing
1	Décidable, Coûteuse	
2	Décidable, Indéterministe	Automate fini à pile indéterministe
$LR(k)$	Décidable, Performante	Automate fini à pile déterministe
3	Décidable, Performante	Automate fini



Notation BNF

- ▶ Introduction du choix dans les productions (factorisation)
- ▶ Non-terminaux : $\langle X \rangle$
- ▶ Productions : $X \rightarrow \alpha$ notée : $\langle X \rangle ::= \alpha$
- ▶ Choix : $\{X \rightarrow \alpha_i\}_{i \in [1, \dots, n]}$ noté : $\langle X \rangle ::= \alpha_1 \mid \dots \mid \alpha_n$

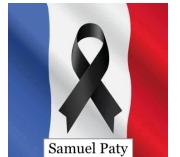


Notation EBNF

- ▶ Introduction des répétitions dans les productions
- ▶ Répétition 0 ou plus de α : $\{\alpha\}$
- ▶ α optionnel : $[\alpha]$
- ▶ Associativité/Priorité : (α)



RÉPUBLIQUE FRANÇAISE



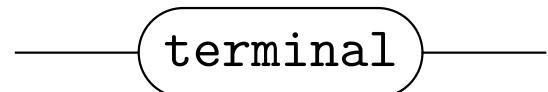
Exemple : Expressions Arithmétiques

```
<Expression> ::=  
  (  
    <Expression> ( + | * ) <Expression>  
  |  
    - <Expression>  
  |  
    CONSTANTE  
  |  
    IDENTIFICATEUR  
    [  
      \  
      <Expression> { VIRGULE <Expression> }  
      \  
    ]  
  |  
    \( <Expression> \)  
)
```



Notation de Conway

- ▶ Forme graphique plus intuitive
- ▶ Terminal :



- ▶ Non terminal :



- ▶ Production $X \rightarrow \alpha$:

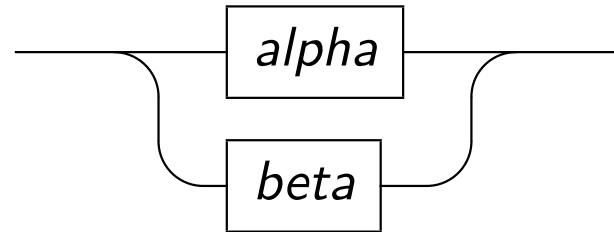


- ▶ Séquence $\alpha\beta$:

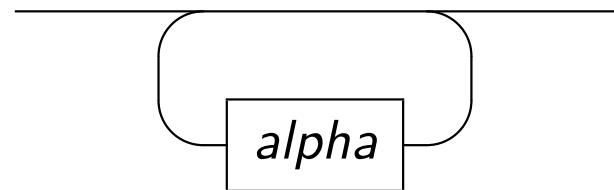


Notation de Conway (bis)

- ▶ Choix entre α et β :

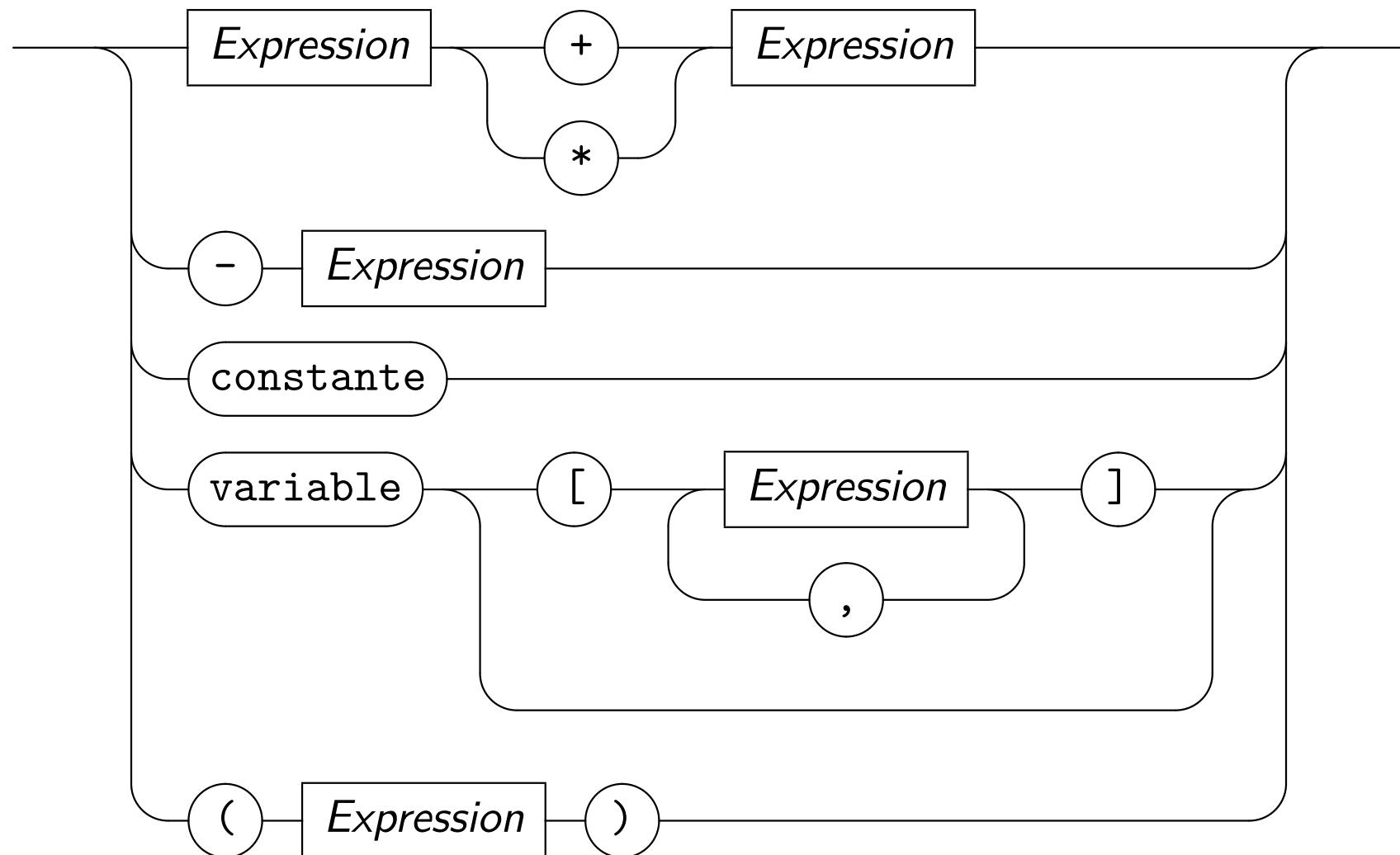


- ▶ Répétition 0 ou plus de α :



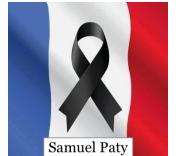
Exemple : Expressions Arithmétiques

Expression



XML : Document Type Definition

- ▶ Objectif : Définir la structure arborescente d'un document
- ▶ Définition des éléments (balises) et de leur contenu (structure)
- ▶ Inspiré de la notation EBNF : mélange règles de production
expression régulière
- ▶ Typage d'un document :
- ▶ DTD externe : `<!DOCTYPE ident SYSTEM "url">`
- ▶ DTD interne : `<!DOCTYPE ident [...]>`
 éléments
- ▶ Définition éléments : `<!ELEMENT ident (...) >`
 structure

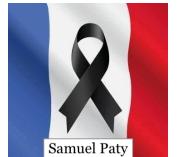


Structure d'un élément

- ▶ Séquence d'autres éléments

	Répétition	* , + , ?
	Choix
▶ Avec :	Associativité/Priorité	(...)
	Feuille texte	#PCDATA
	Feuille vide	EMPTY
	Feuille quelconque	ANY

- ▶ Problème : Récursivité entre éléments interdites par certains outils (utilisation d'entités)



Exemples

```
<!DOCTYPE PERSONNEL [  
    <!ELEMENT enseignant (identité,disipline+)>  
    <!ELEMENT identité ((prénom,nom) | (nom,prénom))>  
    <!ELEMENT prénom #PCDATA>  
    <!ELEMENT nom #PCDATA>  
    <!ELEMENT discipline #PCDATA>  
>  
<enseignant>  
    <identite>  
        <prenom>Marc</prenom>  
        <nom>Pantel</nom>  
    </identite>  
    <discipline>Traduction des langages</discipline>  
    <discipline>Sémantique formelle</discipline>  
</enseignant>
```



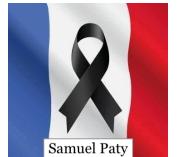
Structure d'un attribut

- Déclaration : `<!ATTLIST ident ident type mode >`

élément répétition

Obligatoire : #REQUIRED

- Mode d'attributs : Optionnel : #IMPLIED
Constant : #FIXED



Structure d'un attribut (bis)

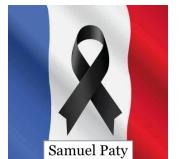
► Type d'attributs :

CDATA	texte brut
NMTOKEN	identificateur XML
NMTOKENS	suite d'identificateurs XML
ENUMERATION	valeurs possibles
ID	unique
IDREF	référence ID
IDREFS	suite de références ID
ENTITY	nom d'entité
ENTITIES	suite de noms d'entité
NOTATION	type MIME



Exemple

```
<!DOCTYPE PERSONNEL [  
    <!ELEMENT enseignant (identité,discipline+)>  
    <!ELEMENT identité #EMPTY>  
    <!ELEMENT discipline #EMPTY>  
  
    <!ATTLIST identité  
        nom CDATA #REQUIRED  
        prénom CDATA #REQUIRED  
    >  
    <!ATTLIST discipline  
        nom CDATA #REQUIRED  
    >  
]>  
  
<enseignant>  
    <identite prenom="Marc" nom="Pantel"/>  
    <discipline nom="Traduction des langages"/>  
    <discipline nom="Sémantique formelle"/>  
</enseignant>
```

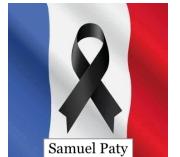


Entité

- ▶ Macro-définitions substituées à la demande (récursivité possible)
- ▶ Utilisation externe DTD :
 - ▶ Définition simple :

```
<!ENTITY ident ' ...'>
<!ENTITY ident " ...">
```
 - ▶ Accès simple : `&ident ;`
- ▶ Utilisation interne DTD :
 - ▶ Définition paramétrée :

```
<!ENTITY % ident ' ...'>
<!ENTITY % ident " ...">
```
 - ▶ Accès paramétré : `%ident ;`



Exemple

```
<!DOCTYPE ARBRE [
    <!ENTITY %arbre; "(noeud|feuille)">
    <!ELEMENT noeud (%arbre; , étiquette , %arbre; )>
    <!ELEMENT feuille (étiquette)>
    <!ELEMENT étiquette #PCDATA>
]>
```

- ▶ Problème : Entités paramétrées mal prise en compte par certains outils pour les DTD internes

