ANALYSE LEXICALE

cf Legendre Schwarzentruber chap. 2.

Idée L'analyse lexicale prépare l'analyse syntaxique.
Elle repère les mots, les lexèmes d'un texte et leur type. C'est donc un peu plus qu'un découpage puisqu'en plus de savoir que de là à là il y a un mot, on sait de quel type il est, et on a pu s'abstraire de la mise en forme du texte, d'éliminer les commentaires.

NB · Le découpage se fait de manière linéaire, il n'y a pas de chevauchements.
· Les types des lexèmes correspondent aux terminaux de la grammaire du langage.

Déf Une règle d'analyse lexicale sur l'alphabet Σ, et sur l'ensemble de type T est un couple (e, t)
où e est une expression rationnelle sur Σ tq $\varepsilon \notin \mathcal{L}(e)$
t est un élément de T

NB → $\varepsilon \notin \mathcal{L}(e)$ pour être sûr que la lecture avance

Pour analyser un texte on peut se donner plusieurs règles. Si les langages de chaques types ne sont pas disjoints on utilisera un ordre de priorité sur les règles.

De plus on appliquera le principe du plus long préfixe, cela signifie que si on a deux lexèmes qui commencent au même endroit on prendra le plus long.

ex

Sur Σ l'ensemble des caractères romains
$T = \{$mot-clé-procédure ; mot-clé-proc, mot-clé prog$\}$
et pour les règles (proc → mot-clé-proc
prog → mot-clé-prog
procédure → mot-clé-procédure .)

procès ⇝ [proc, mot-clé-proc]

progrès ⇝ [prog, mot-clé-prog]

procédure un ⇝ [procédure, mot-clé-procédure]

Considérons un ensemble de règles d'analyses syntaxiques sur un alphabet Σ, numérotées de $j=1$ à m par ordre croissant de priorité. On imagine connaître les expressions rationnelles e_j par des automates $\mathcal{A}_j$ les reconnaissant, et plus précisément des automates

- à états entiers
- d'état initial 1
- de fonction de transition δ_j
- complet et déterministe
- d'état puits 0

entrée : texte t
sortie : liste des lexèmes reconnus dans t.

```
AL(t)
    Si t = ""
    alors retourner []
    sinon (l, r) = AL_aux(t) ; retourner
          retourner l :: AL(r)
```

entrée : texte $t = t_1 \text{ —— } t_n$
sortie : (le premier lexème, $t_{>l}$)

```
AL_aux(t)
    iPLP ← 0
    jPLP ← 0
    n ← length(t)
    i ← 1
    q ← tableau indicé de 1 à m, initialisé à 1
    NbB ← 0

    tant que NbB<m   i ≤ n
    NbB = 0
        Pour j allant de 1 à m
            q[j] ← δ_j(q[j], t_i)
            Si q[j] est final
            alors iPLP ← i ; jPLP ← j
            Si q[j] = 0
            alors NbB ← NbB + 1
        i ← i+1
    Si jPLP = 0   alors erreur
    Sinon renvoyer ( [t_1 —— t_iPLP ; type j_PLP] ; t_iPLP+1 —— t_n )
```