Chapitre 3 : Analyse lexicale

3.1 Le rôle de l'analyseur lexical

L'analyseur lexical constitue la première phase d'un compilateur, sa tâche principale est de lire les caractères d'entrée et de produire comme résultat, une suite d'unité lexicales que l'analyseur syntaxique va utiliser.

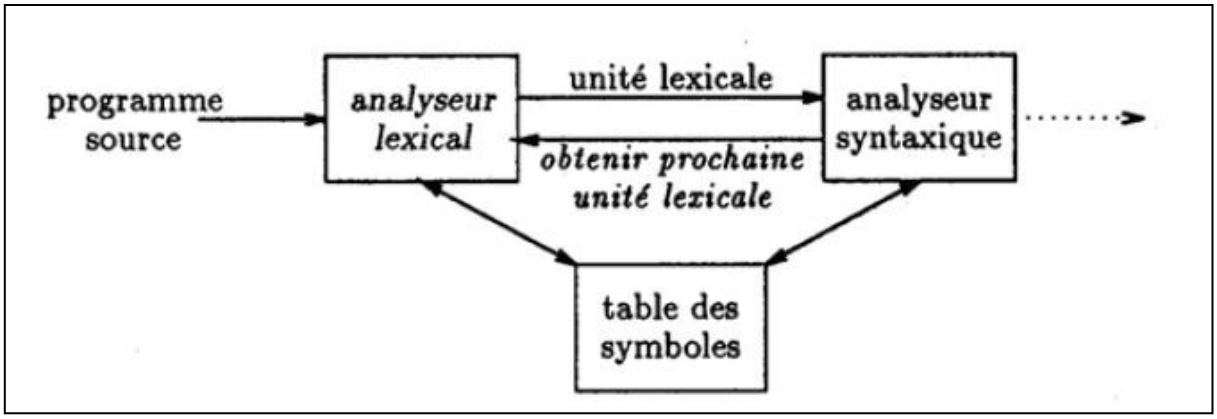


Figure 3.1 : Interaction entre un analyseur lexical et un analyseur syntaxique

A la réception d'une commande "prochaine unité lexicale" de l'analyseur syntaxique, l'analyseur lexical lit les caractères d'entrée jusqu'à ce qu'il puisse obtenir la prochaine unité lexicale.

L'analyseur lexical peut aussi réaliser certaines tâches secondaires comme l'élimination des commentaires et des espaces (caractère blanc, tabulation, fin de ligne).

Une autre tâche consiste à Tenir à jour les coordonnées (ligne, colonne) dans le programme pour fin de production des éventuels messages d'erreur.

3.2 Unités lexicales, modèles et lexèmes

Quand on parle d'analyse lexicale, on utilise les termes « unité lexicale » « modèle » et « lexème » avec des spécifications bien spécifiques. Le tableau 2.1 donne des exemples de leur utilisation. En général, il y a un ensemble de chaînes en entrée pour lesquelles la même unité lexicale est produite en sortie. Cet ensemble de chaînes est décrit par une règle appelée modèle associé à l'unité lexicale. On dit que le modèle filtre chaque chaîne de l'ensemble.

Un lexème est une suite de caractères du programme source qui concorde avec le modèle de l'unité lexicale. Par exemple, dans la déclaration Pascal : *const pi=3.1416*.

La sous-chaîne *pi* est un lexème de l'unité lexicale « identificateur ».

Les lexèmes reconnus par le modèle décrivant l'unité lexicale représentent les chaînes de caractères du langage source qui peuvent être traitées en bloc comme une unité lexicale.

Unité lexicale (Jeton)	Exemples de Lexèmes	Description informelle des modèles
const	const	const
if	if	if
oprel (opérateur de relation)	<, <=, =, <>, > , >=	< ou <= ou = ou <> ou > ou >=
id	pi compte D2	Lettre suivie de lettres ou chiffres
nb	3.1416, 0, 6.02E23	Toute constante numérique
littéral	"cours compilation"	Tous caractère entre "et " sauf "

Tableau 3.1: Exemples d'unités lexicales

Dans la plupart des langages de programmation, les constructions suivantes sont traitées comme des unités lexicales : mots clés, opérateurs, identificateurs, constantes, chaînes littérales et symbole de ponctuation (parenthèses, virgules, :). Dans l'exemple précédent, quand la suite de caractères *pi* apparaît dans le programme source, une unité lexicale représentant un identificateur est retournée à l'analyseur syntaxique (souvent en passant un entier correspondant à l'unité lexicale, c'est cet entier qui est référencé par l'unité lexicale *id*).

Un modèle est une règle qui décrit l'ensemble des lexèmes pouvant représenter une unité lexicale particulière dans les programmes sources. Le modèle de l'unité lexicale *const* est limité à la simple chaîne *const* qui orthographie le mot clé. Le modèle de l'unité lexicale *oprel* est l'ensemble des six opérateurs de relation de pascal. Pour décrire précisément les modèles d'unités lexicales plus complexes, comme *id* et *nb* nous utiliserons la notation des expressions régulières.

3.3 Attributs des unités lexicales

L'analyseur lexical réunit les informations sur les unités lexicales dans les attributs qui leur sont associés.

Les unités lexicales influent sur les décisions de l'analyse syntaxique les attributs influent sur la traduction des unités lexicales. En pratique, une unité lexicale a en général un seul attribut un pointeur vers l'entrée de la table des symboles dans laquelle l'information sur l'unité lexicale est conservée.

Exemple

Les unités lexicales et les valeurs d'attributs associées pour l'instruction Fortran *E = M*C**2* sont données ici sous la forme de couples :

<i**d**, pointeur vers l'entrée de la table des symboles associés à E>

<**op-affectation**, > /* aucune valeur d'attribut n'est nécessaire, le premier composant suffit pour identifier le lexème*/.

<**id**, pointeur vers l'entrée de la table des symboles associés à M>

<**op-multiplication**,>

<**id**, pointeur vers l'entrée de la table des symboles associés à C>

<**op-exponentiation**, >

<**nb**, valeur entière 2> /* Nous avons associé à cette unité un attribut à valeur entière. Le compilateur peut ranger la chaîne de caractères composant le nombre dans la table des symboles et faire en sorte que l'attribut de l'unité lexicale nb soit un pointeur vers la table des symboles.*/

3.4 Reconnaissance des unités lexicales (jetons)

Le but principal de l'analyse lexicale est de reconnaitre les jetons pour une certaine grammaire.

Exemple

instr → **if** expr **then** instr

| **if** expr **then** instr **else** instr

| ε

exp → terme **relop** terme

| terme

terme → **id**

| **num**

Où les terminaux **if**, **then**, **else**, **relop**, **id** et **num** engendrent les ensembles de chaînes données par les définitions régulières suivantes :

if → if

then → then

else → else

relop → < |<= | = | <>| > | >=

id → lettre (lettre | digit)*

num → digit +(. digit +)? (E(+|-)? digit+)?

lettre et digit sont définis comme suit :

lettre → A | B | … | Z | a | b | … |z

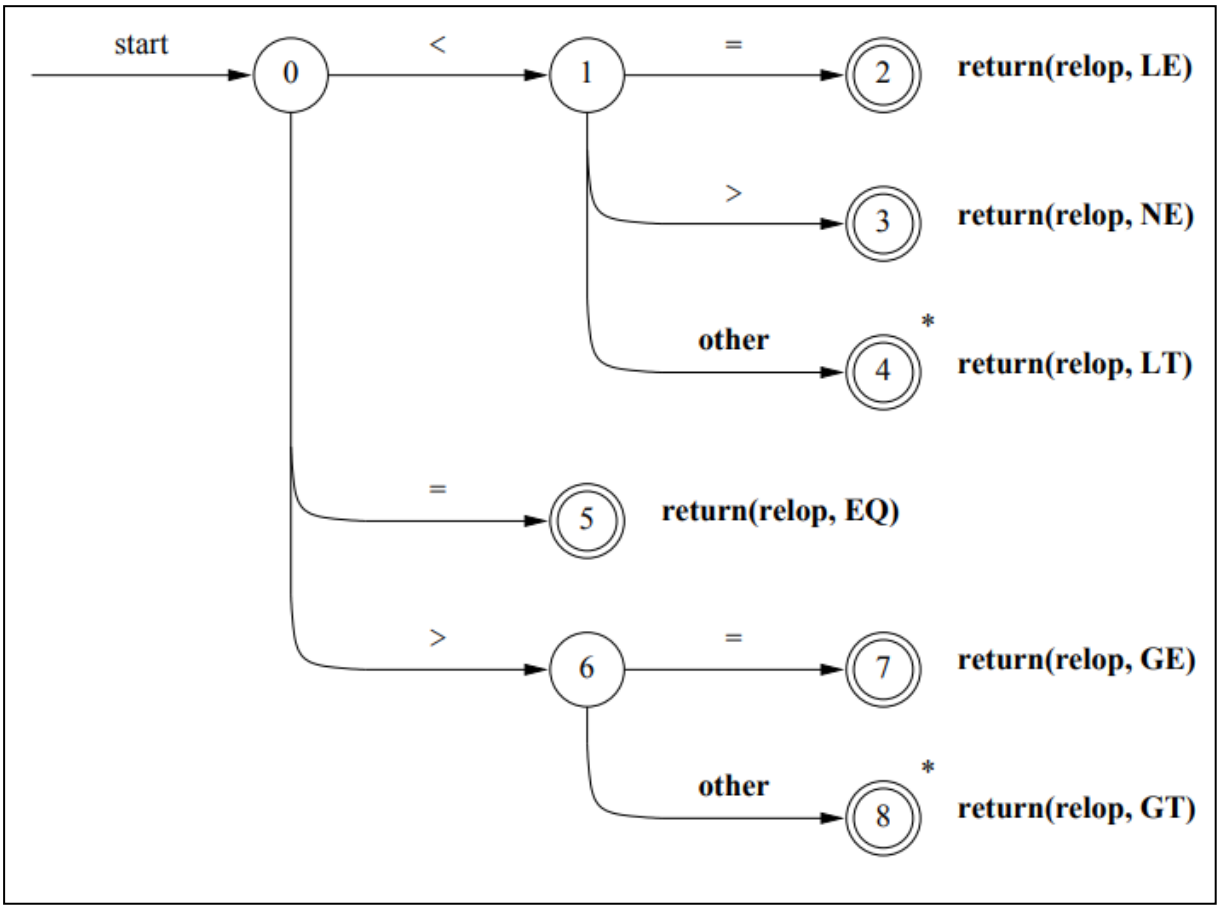
digit → 0 | 1 | … |9

Pour ce fragment de langage l'analyseur lexical reconnaîtra les mots clé si, alors sinon au même titre que les lexèmes dénotés par oprel, id, nb.

On suppose que les lexèmes sont séparés par un espace consistant en une suite non vide de blancs, tabulations et fin de ligne.

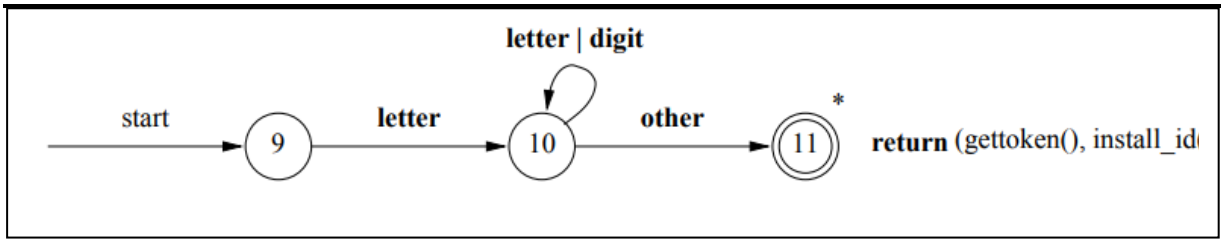
L'analyseur lexical isole le lexème associé à la prochaine unité lexicale du tampon d'entrée et qui produise en sortie un couple composé de l'unité lexicale appropriée et d'une valeur d'attribut. Les attributs pour les opérateurs de relation (oprel) sont donnés par les constantes symboliques LE, NE, LT, EQ, GE, GT.

Reconnaissance des opérateurs relationnels

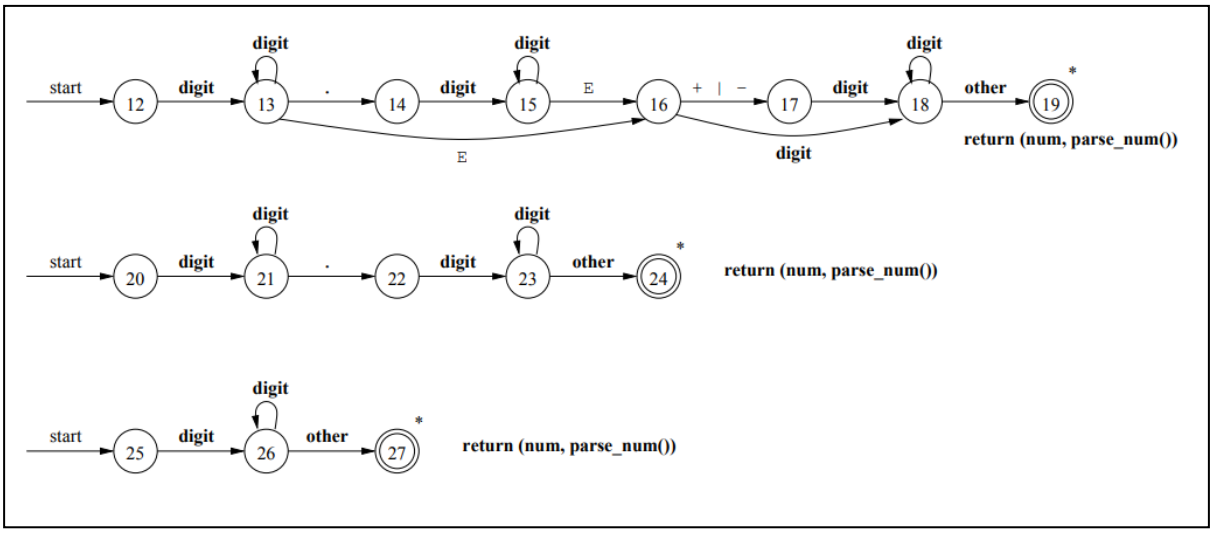


Reconnaissance des identificateurs et mots-clés

Les mots-clés sont préinscrits dans la table des symboles. Celle-ci contient la catégorie lexicale de chacun.

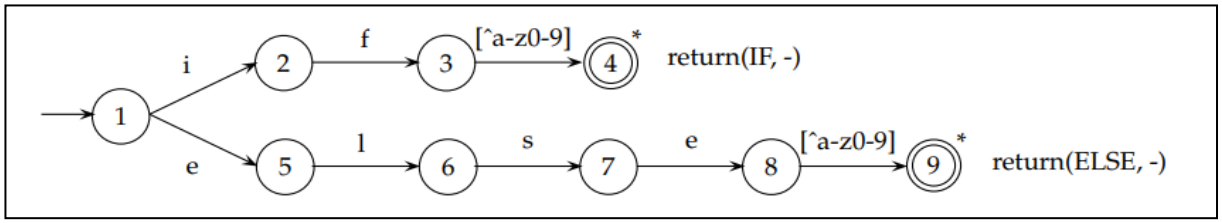


Reconnaissance des constantes numériques

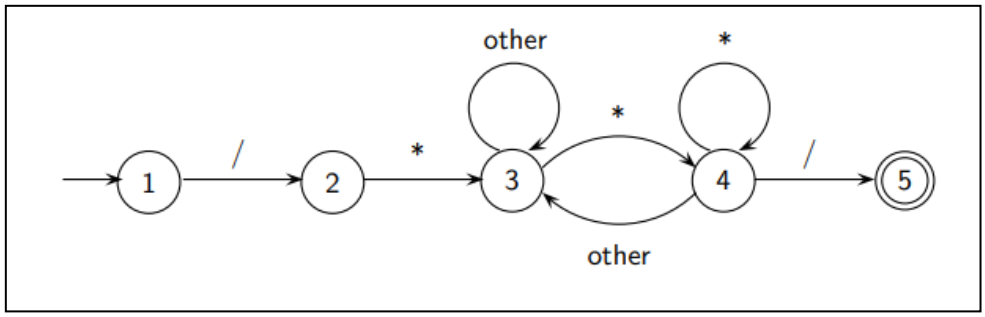


Exemples supplémentaires

Traitement des mots-clés (if et else) par des automates.



Commentaires du langage C.



3.5 Gestion des erreurs

Certaines erreurs sont véritablement de nature lexicale. Par exemple, rencontrer le caractère ASCII numéro 14, qui ne doit jamais apparaitre dans les programmes sources d'un langage donné.

Plusieurs erreurs ne peuvent pas être géré au niveau de l'analyseur lexical. Par exemple, on retrouve dans le programme source la chaine :

fi (a == f(x)) . . .

Cette erreur nous semble de nature lexicale à cause de la faute d'orthographe mais 'fi' constitue un identificateur tout à fait valide.

Il y a plusieurs moyens de gérer une erreur lexicale :

- Arrêter toute la compilation avec un message d'erreur.
- Abandonner des caractères de l'entrée jusqu'à ce qu'un jeton bien formé soit disponible.
- Effectuer une ou plusieurs opérations d''édition comme :
  – Effacer un caractère. (Assure la terminaison.)
  – Insérer un caractère [adéquat].
  – Remplacer un caractère par un autre.
  – Inverser l'ordre de deux caractères consécutifs.
- Trouver une distance d'édition minimale pour obtenir un programme lexicalement valide à partir du programme source.

Un avantage à tenter de recouvrer un état stable après une erreur lexicale consiste à possiblement fournir une liste d'erreurs plus complète pour le programme.

Un inconvénient consiste à possiblement confondre les étapes suivantes de la compilation et ainsi générer des erreurs de compilation incongrues.