Cartouche du document

Année : ING 1 - Matière : Théorie des langages - Activité : Travail dirigé

Objectifs

Ce travail dirigé a pour but :

- L'introduction à la théorie des langages
  - Grammaire
  - Dérivation
- La décomposition d'un problème en analyse lexicale, syntaxique et sémantique

On rappelle que le mot langage s'écrit sans *u* en français.

Sommaire des exercices

1 - Dérivation de règles
2 - Analyse lexicale et syntaxique
3 - Le langage SQL

Corps des exercices

1 - Dérivation de règles

Énoncé :

En théorie des langages, quand les symboles sont déjà des suites de symboles, on parlera de vocabulaire plutôt que d'alphabet et de phrases plutôt que de mots.

On considère le petit langage naturel suivant :

- le vocabulaire A = {le, la, fille, raisin, cueille}
- Une phrase est de la forme groupe nominal verbe groupe nominal.
- Un groupe nominal est de la forme déterminant nom.
- Les déterminants sont la ou le.
- Les noms sont fille ou raisin.
- Le verbe est cueille.

Question 1)

Énoncé de la question

Etablir la grammaire de ce langage

Solution de la question

T = {le, la, fille, raisin, cueille }
N = {PH, GN, N, D, V }
S = PH
P = {

PH —> GN V GN

GN —> D N

N —> fille | raisin
// équivalent à la réunion des 2 règles suivantes :

// N —> fille

// N —> raisin

D —> la | le

V —> cueille
}

Question 2)

Énoncé de la question

Montrer par dérivation que *la fille cueille le raisin* est une phrase du langage.

Solution de la question

PH —> GN V GN —> D N V D N —> la fille cueille le raisin.

la fille cueille le raisin n'est composé que d'éléments terminaux. Il s'agit donc d'une phrase de notre langage.

Note : L'ensemble des dérivations peut également être représenté par un arbre.

Question 3)

Énoncé de la question

Montrer par dérivation que *le raisin cueille la fille* est une phrase du langage.

Solution de la question

PH —> GN V GN —> D N V D N —> le raisin cueille la fille

Question 4)

Énoncé de la question

Que pensez-vous de la deuxième phrase ?

Solution de la question

Elle n'est pas sémantiquement correcte

2 - Analyse lexicale et syntaxique

Énoncé :

On considère le mini-langage de programmation suivant :

- 01) L'alphabet Alp du langage L est formé des lettres {a, ..., z, A, ..., Z}, des chiffres {0, ..., 9}, du point {.}, de la quote {'} et les caractères {+, -, *, /, =}.
- 02) Un identificateur est composé de lettres et de chiffres et commence par une lettre.
- 03) Un nombre peut être un entier (composé de chiffres) ou un décimal (composé d'une partie entière et d'une partie fractionnaire séparées par un point).
- 04) Un opérateur est l'une des 4 opérations arithmétiques.
- 05) Une expression numérique est une suite de nombres ou d'identificateurs reliés par des opérateurs.
- 06) Une affectation numérique associe un identificateur à une expression numérique (par le symbole =).

- 07) Un littéral est une suite de symboles entre simple quotes.
- 08) Une expression alphanumérique est une suite de littéraux ou d'identificateurs reliés par l'opérateur de concaténation (symbole &).
- 09) Une affectation alphanumérique associe un identificateur à une expression alphanumérique.
- 10) Une expression syntaxiquement correcte du langage défini sur Alp est une affectation numérique ou alphanumérique.

On cherche les expressions syntaxiquement correctes.

Question 1)

Énoncé de la question

Décomposer ce problème en analyse lexicale et analyse syntaxique.

Solution de la question

Il peut exister plusieurs décompositions lexicales/syntaxiques en fonction des lexèmes que l'analyse lexicale reconnaît.

Ici, on suppose que les lexèmes reconnus par l'analyse lexicale sont: identificateur, nombre, litteral, operateur numérique, concaténation et affectation.

L'analyse lexicale

Les phrases suivantes définissent les règles pour fabriquer les lexèmes.

- 01) L'alphabet Alp du langage L est formé des lettres {a, ..., z, A, ..., Z}, des chiffres {0, ..., 9}, du point {.}, de la quote {'} et les caractères {+, -, *, /, =}.
- 02) Un identificateur est composé de lettres et de chiffres et commence par une lettre.
- 03) Un nombre peut être un entier (composé de chiffres) ou un décimal (composé d'une partie entière et d'une partie fractionnaire séparées par un point).
- 04) Un opérateur est l'une des 4 opérations arithmétiques.
- 07) Un littéral est une suite de symboles entre simple quotes.

Il faut également ajouter des règles permettant de reconnaître les lexèmes de concaténation et d'affectation.

- 11) L'opérateur de concaténation est le signe &.
- 12) L'opérateur d'affectation est le symbole =.

L'analyse syntaxique

Pour l'analyse syntaxique, on parle de vocabulaire plutôt que d'alphabet.

Le vocabulaire est formé des lexèmes obtenues après l'analyse lexicale. Ces lexèmes sont ceux décrits précédemment : identificateur, nombre, litteral, operateur numérique, plus et affectation.

Les phrases suivantes définissent les règles pour valider la suite de catégories de lexèmes obtenue après l'analyse lexicale.

- 05) Une expression numérique est une suite de nombres ou d'identificateurs reliés par des opérateurs.
- 06) Une affectation numérique associe un identificateur à une expression numérique (par le symbole =).

• 08) Une expression alphanumérique est une suite de littéraux ou d'identificateurs reliés par l'opérateur de concaténation (symbole &).
• 09) Une affectation alphanumérique associe un identificateur à une expression alphanumérique.
• 10) Une expression syntaxiquement correcte du langage défini sur Alp est une affectation numérique ou alphanumérique.

3 - Le langage SQL

Énoncé :

Soit le sous-langage de SQL, permettant d'écrire des requêtes simplifiées de projection et de sélection sur une seule table ayant des champs de type chaîne et/ou de type entier:

SELECT $champ_1$,...,$champ_n$ FROM table WHERE condition

où condition peut prendre l'une des formes suivantes :

• $champ_i$ comp constante
• (condition oper condition)

avec

• oper ∈ {OR,AND}
• comp ∈ {=,>,<}
• constante ∈ {'chaine',entier} où chaine est une suite finie de lettres ou de chiffres et entier une suite finie de chiffres.
• $champ_i$ et table sont des identificateurs (un identificateur étant composé de lettres et de chiffres et commençant par une lettre).

Question 1)

Énoncé de la question

Nous nous plaçons ici dans le cadre de l'analyse lexicale permettant de reconnaître les lexèmes utilisés dans une requête SQL. Donner une grammaire permettant d'identifier les différents lexèmes nécessaires pour les requêtes de ce langage SQL.

Solution de la question

Encore une fois, il y a plusieurs réponses possible en fonction des lexèmes que vous avez choisis.
Ici, nous supposons que les lexèmes reconnus lors de l'analyse lexicales sont : select, from, where, champ, table, condition et virgule.
Pour simplifier, on pose les 2 ensembles suivants :
lettre = {a, ..., z, A, ..., Z }
chiffre = {0, ..., 9 }
G = {

T = { SELECT, FROM, WHERE, OR, AND, ,, ', <, =, >, (,) } ∪ lettre ∪ chiffre
N = { lexemeSQL, select, from, where, comparaison, oper, identificateur, constante, chaîne, entier, condition }
S = lexemeSQL

P = {

lexemeSQL —> select | from | where | champ | table | condition | virgule

virgule —> ,

identificateur —> lettre | identificateur lettre | identificateur chiffre

entier —> chiffre | entier chiffre

chaîne —> lettre | chiffre | chaîne lettre | chaîne chiffre

constante —> 'chaîne' | entier

oper —> AND | OR

comparaison —> < | = | >

condition —> identificateur comparaison constante | (condition oper condition)

table —> identificateur

champ —> identificateur

select —> SELECT

from —> FROM

where —> WHERE

}

}

Question 2)

Énoncé de la question

Cette fois-ci, nous nous allons procéder à l'analyse syntaxique d'une requête SQL. En fonction des lexèmes obtenus lors de la question précédente, proposer une grammaire hors-contexte qui reconnait les requêtes syntaxiquement bien formées.

Les lexèmes obtenus précédemment deviennent donc les éléments terminaux de cette grammaire.

Solution de la question

G = {
T = { select, from, where, champ, table, condition, virgule }
N = { requête, listeChamps }
S = requête
P = {

requête —> select listeChamps from table where condition

listeChamps —> champ | listeChamps virgule champ

}

}