INFO-F403 Introduction to Language Theory and Compilation

Chapeaux Thomas
Dagnely Pierre

March 4, 2013

1 Lexèmes

Définition des tokens

Lexical units	regular expressions
INT	([0-9])*
FLOAT	([0-9])*.DOT.([0-9])*
BOOL	(0+1+true+false+")
STRING	'.([A-Za-z]+[0-9])*.'
FAC	!
MUL	*
DIV	/
MINUS	-
ADD	+
LT	<
GT	>
LE	<=
GE	>=
EQUIV	==
DIF	!=
AND	&&
OR	||
NOT	not
LT-S	lt
GT-S	gt
LE-S	le
GE-S	ge
EQ-S	eq
NE-S	ne

Lexical units	regular expressions
EQUAL	=
DOT	.
SEMICOLON	;
COMA	,
OPEN-PAR	(
CLOSE-PAR	)
OPEN-BRAC	{
CLOSE-BRAC	}
OPEN-COND	IF
CLOSE-COND	ELSE
ADD-COND	ELSE IF
NEG-COND	UNLESS
RET	return
FUNCT-DEF	SUB
ID	STRING
FUNCT-CALL	&.STRING
PERL-DEF	defined
PERL-INT	int
PERL-LENG	length
PERL-SCAL	scalar
PERL-SUBS	substr
PERL-PRIN	print
COMM	#.STRING
VARIABLE	$.STRING

coma peut définir l'opérateur coma ou juste un coma entre deux param, mais même lexical unit, c'est le parser qui se charge du reste

2 Automates

Définition des automates finis

DFA

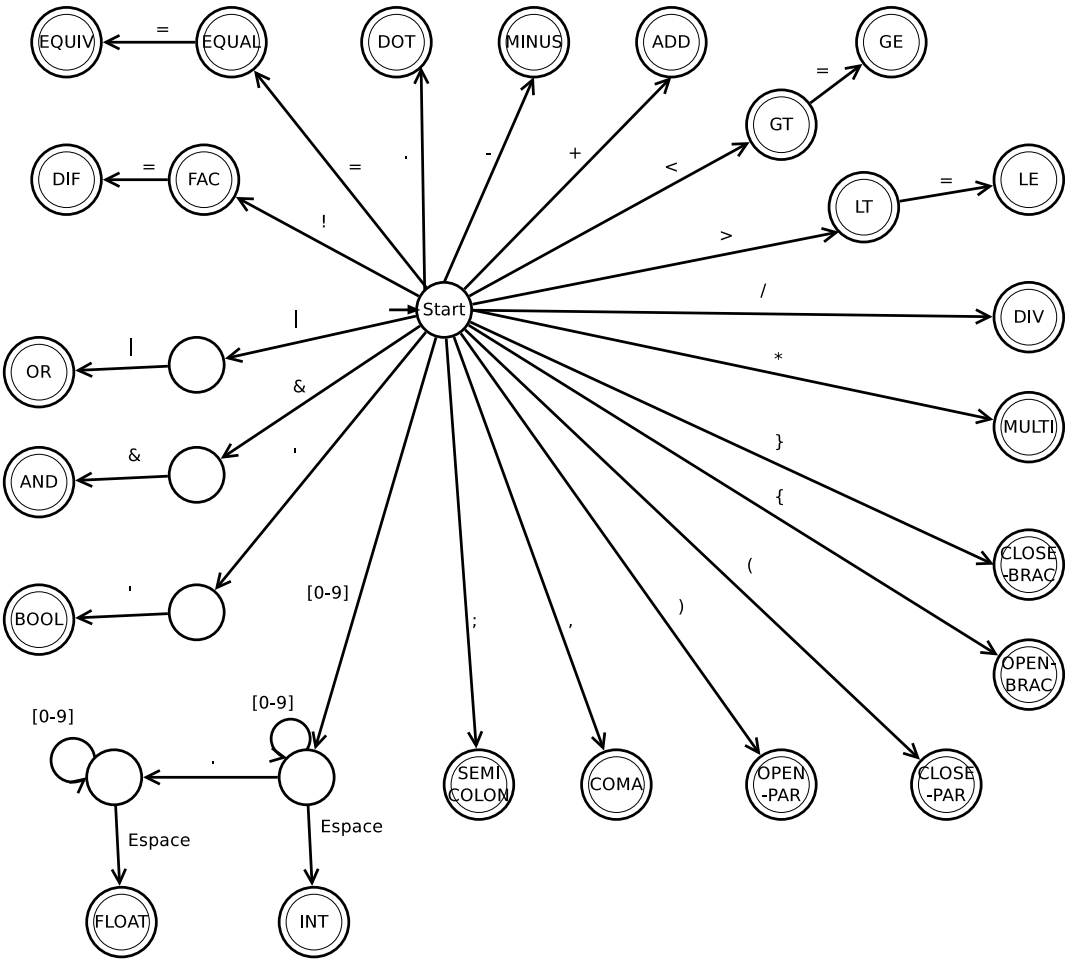

Figure 1: automate "non alphabétique"

La plupart des noeuds pointent vers le token ID, trop lourd a représenter, donc met une petite fleche bleu à la place.

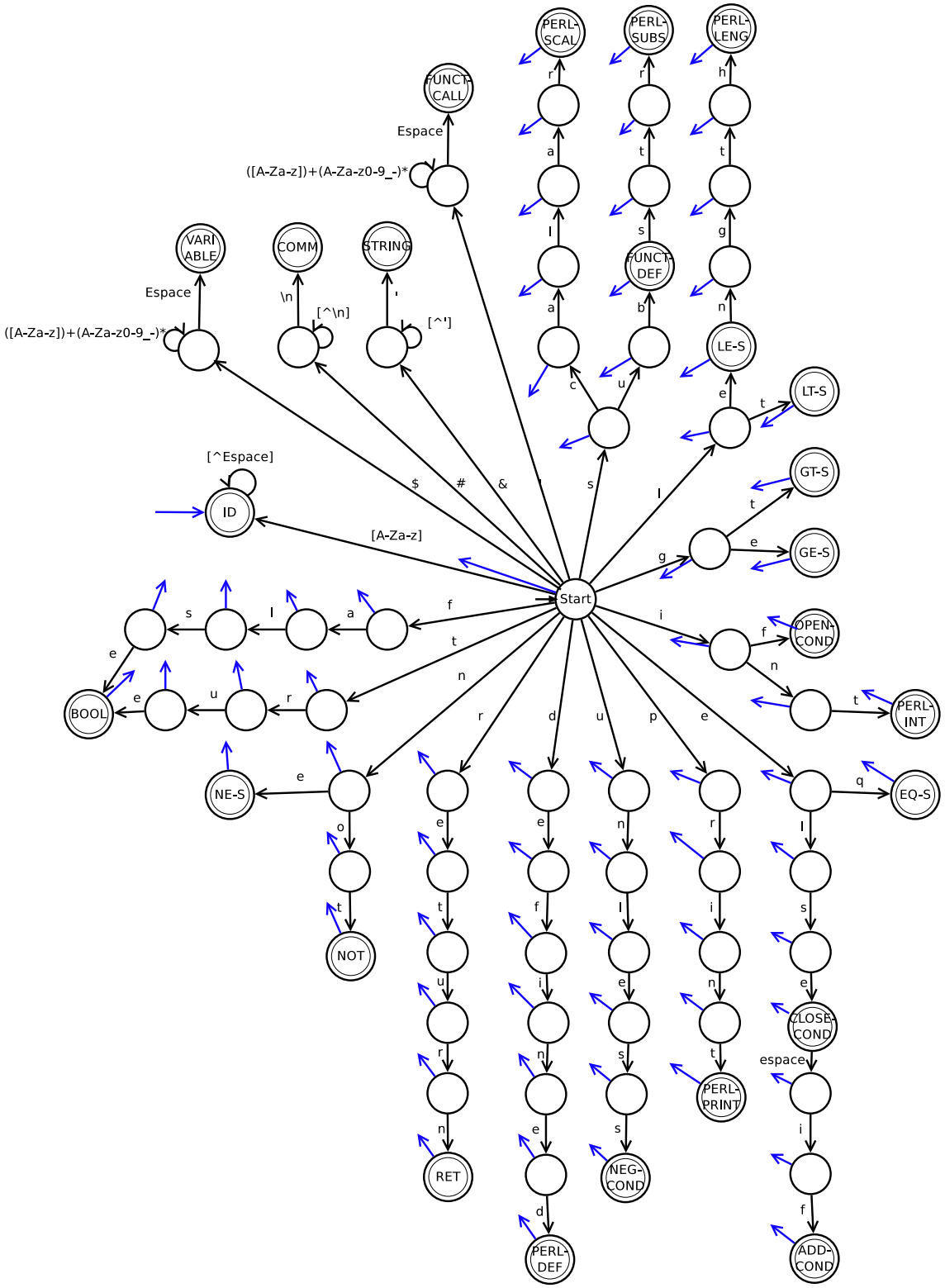

Figure 2: automate "alphabétique"

3 Gramaires

3.1 Définition de la grammaire

On se base sur la BNF donné par l'assistant et on l'adapte à notre version de perl

PROGRAM → PROGRAM FUNCT-LIST
→ PROGRAM INSTRUCT
→ FUNCT-LIST
→ INSTRUCT
→ EPSILON

FUNCT-LIST → FUNCT
→ FUNCT FUNCT-LIST
→ EPSILON

FUNCT → FUNCT-ID FUNCT-NAME OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
→ FUNCT-ID FUNCT-NAME OPEN-PAR CLOSE PAR OPEN-BRAC INSTRUCT RETURN CLOSE-BRA
→ FUNCT-ID FUNCT-NAME OPEN-PAR PARAM CLOSE-PAR OPEN-BRAC INSTRUCT RETURN CL

FUNCT-CALL → USER-FUNCT-CALL
→ PERL-FUNCT-CALL

USER-FUNCT-CALL → FUNCT-NAME OPEN-PAR CLOSE-PAR
→ FUNCT-NAME OPEN-PAR PARAM CLOSE-PAR
→ FUNCT-NAME PARAM
→ FUNCT-NAME

PERL-FUNCT-CALL → PERL-DEF EXP
→ PERL-INT EXP
→ PERL-LENG EXP
→ PERL-SCAL EXP
→ PERL-SUBS EXP COMA INT COMA INT
→ PERL-SUBS EXP COMA INT
→ PERL-PRIN LIST

LIST → STRING
→ STRING LIST
→ EPSILON

PARAM → VAR
→ VAR PARAM-END
→ EPSILON

PARAM-END → COMA VAR
→ COMA VAR PARAM-END
→ EPSILON

RETURN → RET EXP SEMICOLON
→ RET EXP-COND SEMICOLON
→ RET VAR SEMICOLON
→ EPSILON

INSTRUCT	→ COND SEMICOLON INSTRUCT
	→ EXP SEMICOLON INSTRUCT
	→ FUNCT-CALL SEMICOLON INSTRUCT
	→ ASSIGNATION SEMICOLON INSTRUCT
	→ COND SEMICOLON
	→ EXP SEMICOLON
	→ FUNCT-CALL SEMICOLON
	→ ASSIGNATION SEMICOLON
	→ EPSILON
ASSIGNATION	→ VAR EQUAL VALUE
	→ VAR EQUAL EXP
COND	→ OPEN-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ NEG-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ EXP OPEN-COND EXP-COND
	→ EXP NEG-COND EXP-COND
COND-END	→ ADD-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC
	→ ADD-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ CLOSE-COND OPEN-BRAC INSTRUCT CLOSE-BRAC
	→ EPSILON
EXP	→ VAR
	→ EXP OPERATOR EXP
	→ EXP-COMP
	→ NOT EXP
	→ FAC EXP
	→ ADD EXP
	→ MINUS EXP
EXP-COMP	→ EXP OPERATOR-COMP EXP
OPERATOR	→ MUL
	→ DIV
	→ MINUS
	→ CONC
	→ ADD
OPERATOR-COMP	→ LT
	→ GT
	→ LE
	→ GE
	→ EQUIV
	→ DIF
	→ AND-LOGIC
	→ OR
	→ LT-S
	→ GT-S
	→ LE-S
	→ GE-S
	→ COMA-LOGIC
	→ EQ-S
	→ NE-S
VALUE	→ INT
	→ FLOAT
	→ BOOL
	→ STRING
VAR	→ VALUE
	→ MINUS VAR
	→ ADD VAR
	→ OPEN-PAR EXP CLOSE-PAR

3.2 Gestion des priorités

On doit adapter la grammaire pour respecter les priorité et les associativité gauche/-droite.
Cela ne modifie que les règles EXP, EXP-COND, OPERATOR et OPERATOR-COND qui deviennent :

<EXP> → <EXP> LT <EXP2>
→ <EXP> LT-S <EXP2>
→ <EXP> GT <EXP2>
→ <EXP> LT-S <EXP2>
→ <EXP> LE <EXP2>
→ <EXP> LE-S <EXP2>
→ <EXP> GE <EXP2>
→ <EXP> GE-S <EXP2>
→ <EXP> EQUIV <EXP2>
→ <EXP> DIF <EXP2>
→ <EXP2> EQUAL <EXP>
→ NOT <EXP>
→ FAC <EXP>
→ <EXP2>
<EXP2> → <EXP> ADD <EXP3>
→ <EXP> MINUS <EXP3>
→ <EXP> OR <EXP3>
→ <EXP3>
<EXP3> → <EXP> MUL <VAR>
→ <EXP> DIV <VAR>
→ <EXP> AND <VAR>
→ <VAR>

3.3 Suppression des symboles inutiles

On doit virer non-productifs et inaccessibles.
Ici ok, rien à faire

3.4 left-factoring

On vire ce qui commence pareillement

On obtient la gram suivante :

PROGRAM	→ PROGRAM PROG-END
	→ FUNCT-LIST
	→ INSTRUCT
	→ EPSILON
PROG-END	→ FUNCT-LIST
	→ INSTRUCT
FUNCT-LIST	→ FUNCT FUNCT-LIST-END
	→ EPSILON
FUNCT-LIST-END	→ FUNCT-LIST
	→ EPSILON
FUNCT	→ FUNCT-ID FUNCT-NAME FUNCT-END
FUNCT-END	→ OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
	→ OPEN-PAR FUNCT-END2
FUNCT-END2	→ CLOSE PAR OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
	→ PARAM CLOSE-PAR OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
FUNCT-CALL	→ USER-FUNCT-CALL
	→ PERL-FUNCT-CALL
USER-FUNCT-CALL	→ FUNCT-NAME USER-FUNCT-CALL-END
USER-FUNCT-CALL-END	→ OPEN-PAR USER-FUNCT-CALL-END2
	→ PARAM
	→ EPSILON
USER-FUNCT-CALL-END2	→ CLOSE-PAR
	→ PARAM CLOSE-PAR
PERL-FUNCT-CALL	→ PERL-DEF EXP
	→ PERL-INT EXP
	→ PERL-LENG EXP
	→ PERL-SCAL EXP
	→ PERL-SUBS PERL-SUBS-END
	→ PERL-PRIN LIST
PERL-SUBS-END	→ EXP COMA INT COMA INT
	→ EXP COMA INT
LIST	→ STRING LIST-END
LIST-END	→ LIST
	→ EPSILON
PARAM	→ PARAM2
	→ EPSILON
PARAM2	→ PARAM-END
	→ EPSILON
PARAM-END	→ COMA VAR PARAM-END2
	→ EPSILON
PARAM-END2	→ PARAM-END
	→ EPSILON
RETURN	→ RET RETURN-END
	→ EPSILON
RETURN-END	→ EXP SEMICOLON
	→ EXP-COND SEMICOLON
	→ VAR SEMICOLON

INSTRUCT	→ COND INSTRUCT-END
	→ EXP INSTRUCT-END
	→ FUNCT-CALL INSTRUCT-END
	→ ASSIGNATION INSTRUCT-END
	→ EPSILON
INSTRUCT-END	→ SEMICOLON INSTRUCT-END2
INSTRUCT-END2	→ INSTRUCT
	→ EPSILON
ASSIGNATION	→ VAR EQUAL ASSIGNATION-END
ASSIGNATION-END	→ VALUE
	→ EXP
COND	→ OPEN-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ NEG-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ EXP COND-END2
COND-END2	→ OPEN-COND EXP-COND
	→ NEG-COND EXP-COND
COND-END	→ ADD-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END3
	→ CLOSE-COND OPEN-BRAC INSTRUCT CLOSE-BRAC
	→ EPSILON
COND-END3	→ COND-END
	→ EPSILON
<EXP>	→ <EXP> <EXP-END>
	→ <EXP2> EQUAL <EXP>
	→ NOT <EXP>
	→ FAC <EXP>
	→ <EXP2>
<EXP-END>	→ LT <EXP2>
	→ LT-S <EXP2>
	→ GT <EXP2>
	→ LT-S <EXP2>
	→ LE <EXP2>
	→ LE-S <EXP2>
	→ GE <EXP2>
	→ GE-S <EXP2>
	→ EQUIV <EXP2>
	→ DIF <EXP2>
<EXP2>	→ <EXP> <EXP2-END>
	→ <EXP3>
<EXP2-END>	→ ADD <EXP3>
	→ MINUS <EXP3>
	→ OR <EXP3>
<EXP3>	→ <EXP> <EXP3-END>
	→ <VAR>
<EXP3-END>	→ MUL <VAR>
	→ DIV <VAR>
	→ AND <VAR>
VALUE	→ INT
	→ FLOAT
	→ BOOL
	→ STRING
VAR	→ VALUE
	→ MINUS VAR
	→ ADD VAR
	→ OPEN-PAR EXP CLOSE-PAR

3.5 récursion gauche

3.6 Grammaire finale

VERSION PROVISOIRE !!!!!!!!!!!!!!!!!!!!!!!!!!!

PROGRAM	→ PROGRAM PROG-END
	→ FUNCT-LIST
	→ INSTRUCT
	→ EPSILON
PROG-END	→ FUNCT-LIST
	→ INSTRUCT
FUNCT-LIST	→ FUNCT FUNCT-LIST-END
	→ EPSILON
FUNCT-LIST-END	→ FUNCT-LIST
	→ EPSILON
FUNCT	→ FUNCT-ID FUNCT-NAME FUNCT-END
FUNCT-END	→ OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
	→ OPEN-PAR FUNCT-END2
FUNCT-END2	→ CLOSE PAR OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
	→ PARAM CLOSE-PAR OPEN-BRAC INSTRUCT RETURN CLOSE-BRA
FUNCT-CALL	→ USER-FUNCT-CALL
	→ PERL-FUNCT-CALL
USER-FUNCT-CALL	→ FUNCT-NAME USER-FUNCT-CALL-END
USER-FUNCT-CALL-END	→ OPEN-PAR USER-FUNCT-CALL-END2
	→ PARAM
	→ EPSILON
USER-FUNCT-CALL-END2	→ CLOSE-PAR
	→ PARAM CLOSE-PAR
PERL-FUNCT-CALL	→ PERL-DEF EXP
	→ PERL-INT EXP
	→ PERL-LENG EXP
	→ PERL-SCAL EXP
	→ PERL-SUBS PERL-SUBS-END
	→ PERL-PRIN LIST
PERL-SUBS-END	→ EXP COMA INT COMA INT
	→ EXP COMA INT
LIST	→ STRING LIST-END
LIST-END	→ LIST
	→ EPSILON
PARAM	→ PARAM2
	→ EPSILON
PARAM2	→ PARAM-END
	→ EPSILON
PARAM-END	→ COMA VAR PARAM-END2
	→ EPSILON
PARAM-END2	→ PARAM-END
	→ EPSILON
RETURN	→ RET RETURN-END
	→ EPSILON
RETURN-END	→ EXP SEMICOLON
	→ EXP-COND SEMICOLON
	→ VAR SEMICOLON

INSTRUCT	→ COND INSTRUCT-END
	→ EXP INSTRUCT-END
	→ FUNCT-CALL INSTRUCT-END
	→ ASSIGNATION INSTRUCT-END
	→ EPSILON
INSTRUCT-END	→ SEMICOLON INSTRUCT-END2
INSTRUCT-END2	→ INSTRUCT
	→ EPSILON
ASSIGNATION	→ VAR EQUAL ASSIGNATION-END
ASSIGNATION-END	→ VALUE
	→ EXP
COND	→ OPEN-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ NEG-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
	→ EXP COND-END2
COND-END2	→ OPEN-COND EXP-COND
	→ NEG-COND EXP-COND
COND-END	→ ADD-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END3
	→ CLOSE-COND OPEN-BRAC INSTRUCT CLOSE-BRAC
	→ EPSILON
COND-END3	→ COND-END
	→ EPSILON
<EXP>	→ <EXP> <EXP-END>
	→ <EXP2> EQUAL <EXP>
	→ NOT <EXP>
	→ FAC <EXP>
	→ <EXP2>
<EXP-END>	→ LT <EXP2>
	→ LT-S <EXP2>
	→ GT <EXP2>
	→ LT-S <EXP2>
	→ LE <EXP2>
	→ LE-S <EXP2>
	→ GE <EXP2>
	→ GE-S <EXP2>
	→ EQUIV <EXP2>
	→ DIF <EXP2>
<EXP2>	→ <EXP> <EXP2-END>
	→ <EXP3>
<EXP2-END>	→ ADD <EXP3>
	→ MINUS <EXP3>
	→ OR <EXP3>
<EXP3>	→ <EXP> <EXP3-END>
	→ <VAR>
<EXP3-END>	→ MUL <VAR>
	→ DIV <VAR>
	→ AND <VAR>
VALUE	→ INT
	→ FLOAT
	→ BOOL
	→ STRING
VAR	→ VALUE
	→ MINUS VAR
	→ ADD VAR
	→ OPEN-PAR EXP CLOSE-PAR

4 Poubelle

EXPRESSION (?)	VARIABLE OPERATOR VARIABLE EXPRESSION OPERATOR VARIABLE
XPRESSION-COND (?)	VARIABLE OPERATOR-COMP VARIABLE EXPRESSION OPERATOR-COMP VARIABLE
ASSIGNATION	VARIABLE EQUAL VALUE
CONDITION (?)	((OPEN-COND+NEG-COND)EXPRESSION-COND OPEN-BRAC INSTRUCTIONS* CLOSE-BRAC (ADD-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS* CLOSE-BRAC)* (CLOSE-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS* CLOSE-BRAC)) + EXPRESSION (OPEN-COND + NEG-COND) EXPRESSION-COND
INSTRUCTIONS	((CONDITION SEMICOLON)* + (EXPRESSION SEMICOLON)* + (FUNCTION-CALL SEMICOLON)* + (ASSIGNATION SEMICOLON)*)*
PARAM	DOLLAR VARIABLE (COMA DOLLAR VARIABLE)*
USER-FUNCT-CALL	AND FUNCTION-NAME (OPEN-PAR CLOSE-PAR + OPEN-PAR PARAM CLOSE-PAR + PARAM) SEMICOLON
PERL-FUNCT-CALL	defined EXPRESSION + int EXPRESSION + length EXPRESSION scalar EXPRESSION + substr EXPRESSION COMA INT COMA INT scalar EXPRESSION + substr EXPRESSION COMA INT + print (?liste de string)
FUNCTION-CALL	USER-FUNCT-CALL + PERL-FUNCT-CALL
FUNCTION	FUNCTION-ID FUNCTION-NAME (OPEN-PAR CLOSE PAR + OPEN-PAR PARAM CLOSE-PAR) OPEN-BRAC INSTRUCTIONS (RETURN EXPRESSION + RETURN EXPRESSION-COND + RETURN VARIABLE) SEMICOLON CLOSE-BRAC
FUNCTION-LIST	FUNCTION*
PROGRAM	(FUNCTION-LIST + INSTRUCTIONS)*

Grammaire après gestion des priorité et associativité :

PROGRAM	→ PROGRAM FUNCT-LIST
	→ PROGRAM INSTRUCT
	→ FUNCT-LIST
	→ INSTRUCT
	→ EPSILON
FUNCT-LIST	→ FUNCT
	→ FUNCT FUNCT-LIST
	→ EPSILON
FUNCT	→ FUNCT-ID FUNCT-NAME OPEN-BRAC INSTRUCT RETURN CLOSE-BRAC
	→ FUNCT-ID FUNCT-NAME OPEN-PAR CLOSE PAR OPEN-BRAC INSTRUCT RETURN CLOSE-BRA
	→ FUNCT-ID FUNCT-NAME OPEN-PAR PARAM CLOSE-PAR OPEN-BRAC INSTRUCT RETURN CL
FUNCT-CALL	→ USER-FUNCT-CALL
	→ PERL-FUNCT-CALL
USER-FUNCT-CALL	→ FUNCT-NAME OPEN-PAR CLOSE-PAR
	→ FUNCT-NAME OPEN-PAR PARAM CLOSE-PAR
	→ FUNCT-NAME PARAM
	→ FUNCT-NAME
PERL-FUNCT-CALL	→ PERL-DEF EXP
	→ PERL-INT EXP
	→ PERL-LENG EXP
	→ PERL-SCAL EXP
	→ PERL-SUBS EXP COMA INT COMA INT
	→ PERL-SUBS EXP COMA INT
	→ PERL-PRIN LIST
LIST	→ STRING
	→ STRING LIST
	→ EPSILON
PARAM	→ VAR
	→ VAR PARAM-END
	→ EPSILON
PARAM-END	→ COMA VAR
	→ COMA VAR PARAM-END
	→ EPSILON
RETURN	→ RET EXP SEMICOLON
	→ RET EXP-COND SEMICOLON
	→ RET VAR SEMICOLON
	→ EPSILON

INSTRUCT → COND SEMICOLON INSTRUCT
→ EXP SEMICOLON INSTRUCT
→ FUNCT-CALL SEMICOLON INSTRUCT
→ ASSIGNATION SEMICOLON INSTRUCT
→ COND SEMICOLON
→ EXP SEMICOLON
→ FUNCT-CALL SEMICOLON
→ ASSIGNATION SEMICOLON
→ EPSILON

ASSIGNATION → VAR EQUAL VALUE
→ VAR EQUAL EXP

COND → OPEN-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
→ NEG-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
→ EXP OPEN-COND EXP-COND
→ EXP NEG-COND EXP-COND

COND-END → ADD-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC
→ ADD-COND EXP-COND OPEN-BRAC INSTRUCT CLOSE-BRAC COND-END
→ CLOSE-COND OPEN-BRAC INSTRUCT CLOSE-BRAC
→ EPSILON

<EXP> → <EXP> LT <EXP2>
→ <EXP> LT-S <EXP2>
→ <EXP> GT <EXP2>
→ <EXP> LT-S <EXP2>
→ <EXP> LE <EXP2>
→ <EXP> LE-S <EXP2>
→ <EXP> GE <EXP2>
→ <EXP> GE-S <EXP2>
→ <EXP> EQUIV <EXP2>
→ <EXP> DIF <EXP2>
→ <EXP2> EQUAL <EXP>
→ NOT <EXP>
→ FAC <EXP>
→ <EXP2>

<EXP2> → <EXP> ADD <EXP3>
→ <EXP> MINUS <EXP3>
→ <EXP> OR <EXP3>
→ <EXP3>

<EXP3> → <EXP> MUL <VAR>
→ <EXP> DIV <VAR>
→ <EXP> AND <VAR>
→ <VAR>

VALUE → INT
→ FLOAT
→ BOOL
→ STRING

VAR → VALUE
→ MINUS VAR
→ ADD VAR
→ OPEN-PAR EXP CLOSE-PAR