

# Project INFO F 403 : Compilateur Perl

RODRIGUEZ Paul, VACCARI Eric

15 mars 2013

# Table des matières

<b>1</b>	<b>Unités lexicales</b>	<b>3</b>
1.1	Tableau . . . . .	3
1.2	Remarques . . . . .	3
<b>2</b>	<b>DFA</b>	<b>4</b>
2.1	Variables, comparateurs, blocs, littéraux . . . . .	4
2.2	Else, elsif et identifier . . . . .	5
2.3	Opérateurs et divers . . . . .	6
2.4	Remarques . . . . .	6
<b>3</b>	<b>Grammaire LL(1)</b>	<b>6</b>
3.1	Liste des règles . . . . .	6
3.2	$First_1$ . . . . .	9
3.3	$Follow_1$ . . . . .	10
<b>4</b>	<b>Table d'actions</b>	<b>13</b>

# 1 Unités lexicales

## 1.1 Tableau

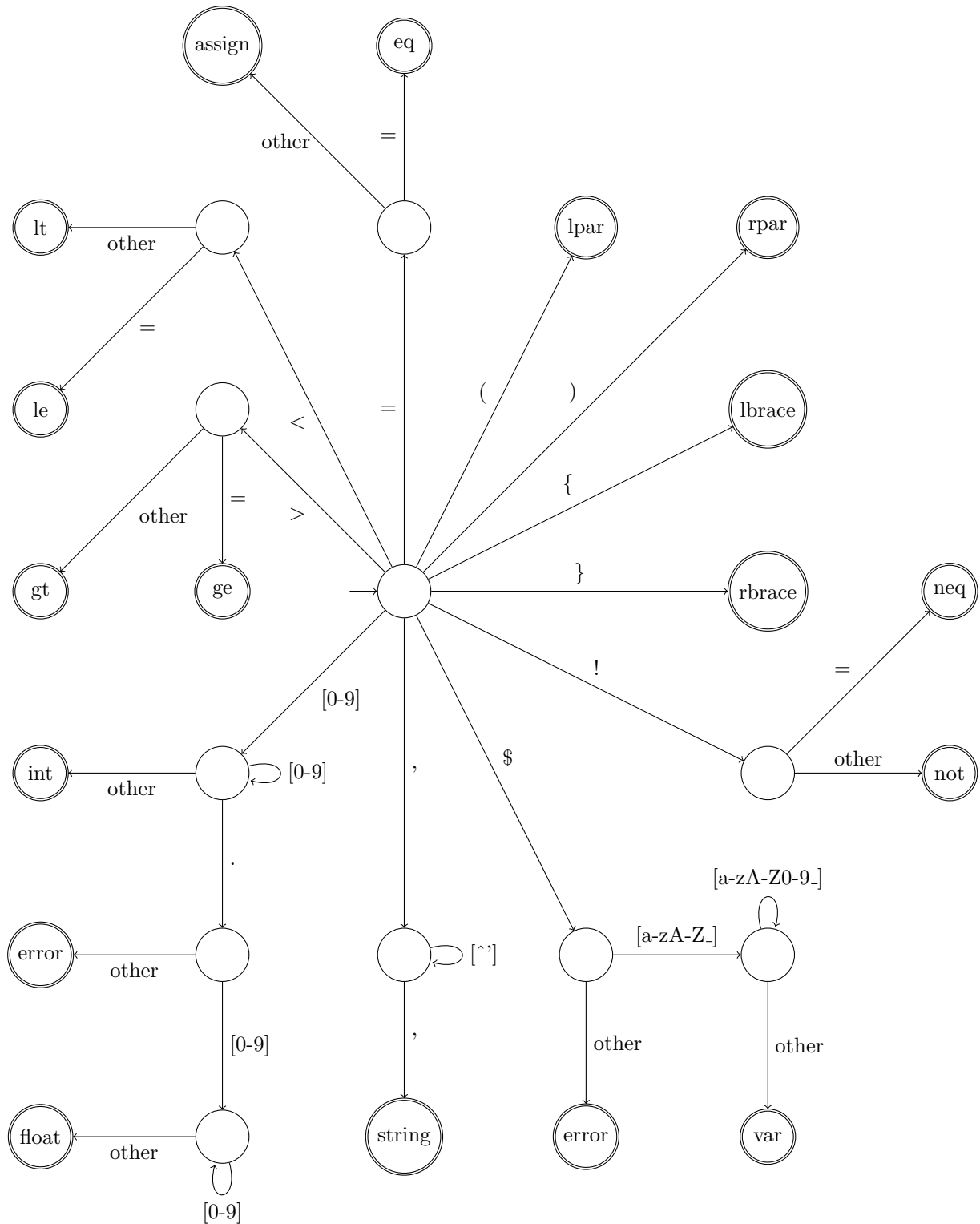
Nom	Regex
var	<code>[\$[a-zA-Z_][a-zA-Z0-9_]*]</code>
identifïer	<code>[a-zA-Z_][a-zA-Z0-9_]*</code>
integer	<code>[0-9]+</code>
float	<code>{integer}\.{integer}</code>
string	<code>'[^']*'</code>
space	<code>[\t\n ]</code>
comment	<code>#.*\n</code>
lbrace	<code>\{</code>
rbrace	<code>\}</code>
lpar	<code>\(</code>
rpar	<code>\)</code>
semicolon	<code>;</code>
call_mark	<code>&amp;</code>
plus	<code>\+</code>
minus	<code>\-</code>
times	<code>\*</code>
divide	<code>\/</code>
not	<code>!</code>
notletters	<code>not</code>
lazy_and	<code>&amp;&amp;</code>
lazy_or	<code>  </code>
equals	<code>==</code>
eq	<code>eq</code>
different	<code>!=</code>
ne	<code>ne</code>
lower	<code>&lt;</code>
lt	<code>lt</code>
greater	<code>&gt;</code>
gt	<code>gt</code>
lower_equals	<code>&lt;=</code>
le	<code>le</code>
greater_equals	<code>&gt;=</code>
ge	<code>ge</code>
comma	<code>,</code>
concat_mark	<code>\.</code>
assign_mark	<code>=</code>
sub	<code>sub</code>
if	<code>if</code>
else	<code>else</code>
elsif	<code>elsif</code>
unless	<code>unless</code>
return	<code>return</code>

## 1.2 Remarques

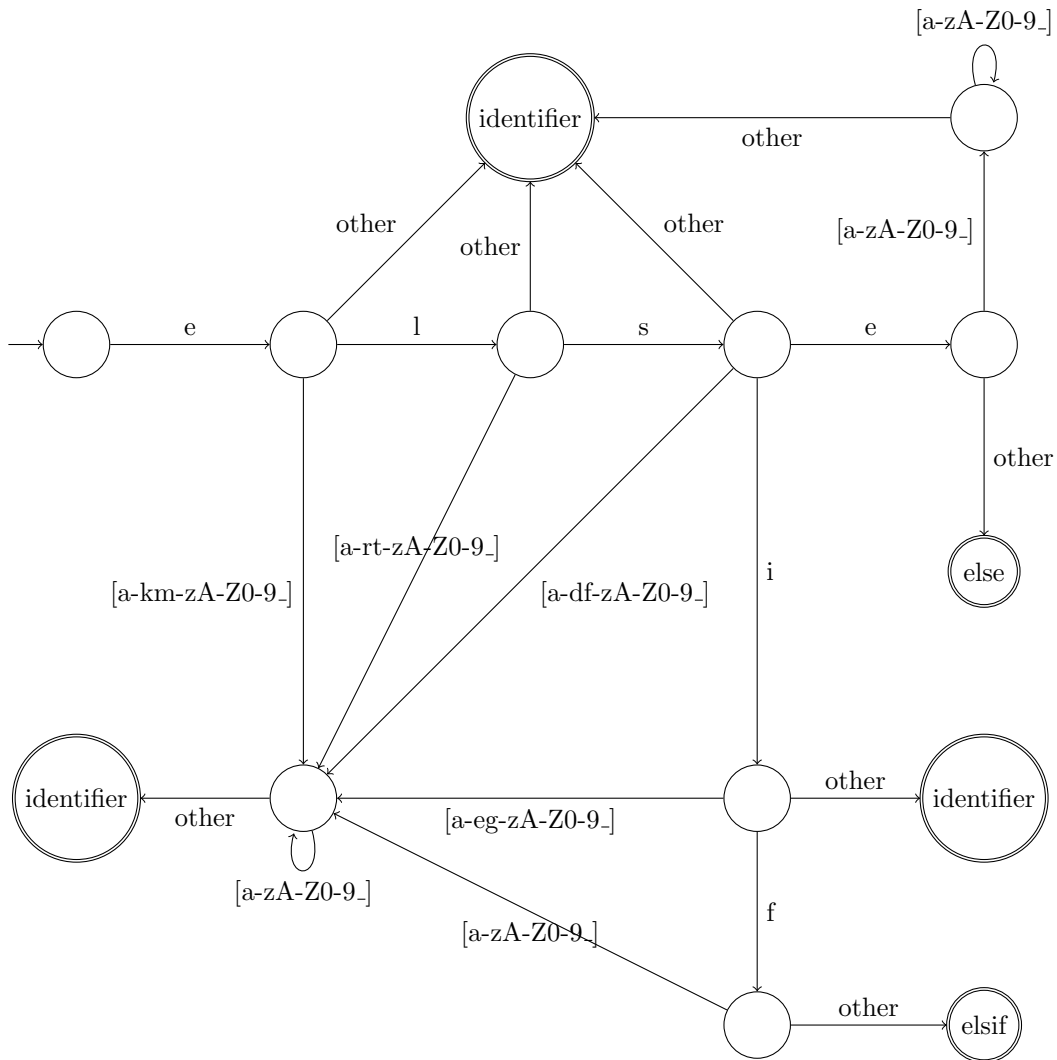
La syntaxe complète de Perl concernant les noms de variables est beaucoup plus compliquée mais concerne des fonctionnalités (packages) hors du cadre de ce projet, ce pourquoi nous nous sommes limités aux règles les plus simples.

## 2 DFA

### 2.1 Variables, comparateurs, blocs, littéraux

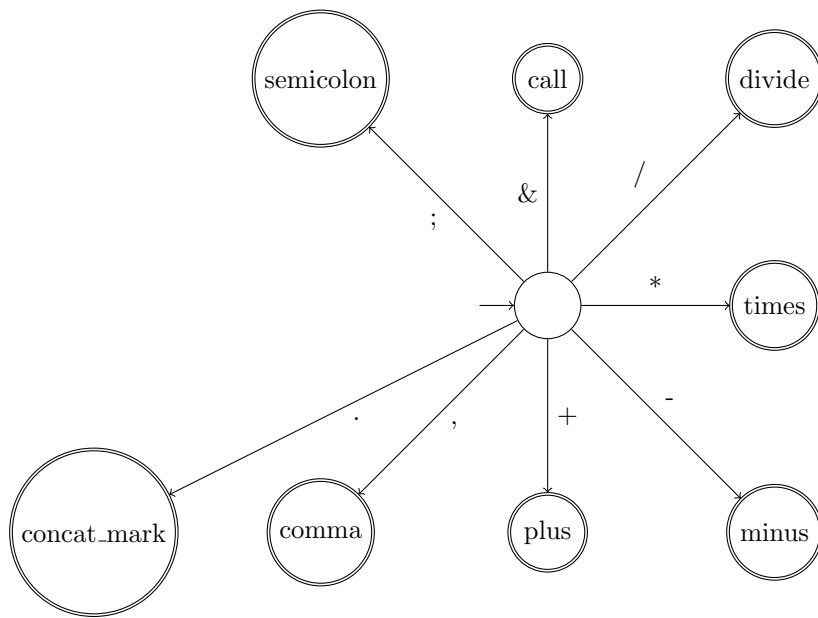


## 2.2 Else, elsif et identifier



Nous avons décidé de ne représenter que ces deux exemples, tous les mots clés fonctionnent sur le même principe.

## 2.3 Opérateurs et divers



## 2.4 Remarques

Certains tokens sont identifiables dès que leur dernier caractère a été lu (par exemple les accolades), d'autres nécessitent la lecture du caractère suivant le dernier (par exemple, pour terminer un entier il faut lire autre chose qu'un chiffre). Dans ce deuxième cas, après avoir identifié le token la lecture du dernier caractère est annulée, il servira comme premier caractère du token suivant.

## 3 Grammaire LL(1)

### 3.1 Liste des règles

[1]	$\langle \text{PROGRAM} \rangle$	$\rightarrow \langle \text{PROGRAM\_F} \rangle \langle \text{PROGRAM\_V} \rangle$
[2]	$\langle \text{PROGRAM\_V} \rangle$	$\rightarrow \langle \text{PROGRAM\_F} \rangle \langle \text{PROGRAM\_V} \rangle$
[3]		$\rightarrow \epsilon$
[4]	$\langle \text{PROGRAM\_F} \rangle$	$\rightarrow \langle \text{FUNCTION} \rangle$
[5]		$\rightarrow \langle \text{INSTRUCTION} \rangle$
[6]	$\langle \text{FUNCTION} \rangle$	$\rightarrow \text{SUB IDENTIFIER } \langle \text{FUNCTION\_ARGUMENT} \rangle$ $\text{LBRACE } \langle \text{INSTRUCTION\_LIST} \rangle \text{ RBRACE}$
[7]	$\langle \text{FUNCTION\_ARGUMENT} \rangle$	$\rightarrow \text{LPAR } \langle \text{ARGUMENT\_LIST} \rangle \text{ RPAR}$
[8]		$\rightarrow \epsilon$
[9]	$\langle \text{ARGUMENT\_LIST} \rangle$	$\rightarrow \text{VAR } \langle \text{ARGUMENT\_LIST\_V} \rangle$
[10]		$\rightarrow \epsilon$
[11]	$\langle \text{ARGUMENT\_LIST\_V} \rangle$	$\rightarrow \text{COMMA VAR } \langle \text{ARGUMENT\_LIST\_V} \rangle$
[12]		$\rightarrow \epsilon$
[13]	$\langle \text{INSTRUCTION\_LIST} \rangle$	$\rightarrow \langle \text{INSTRUCTION} \rangle \langle \text{INSTRUCTION\_LIST} \rangle$
[14]		$\rightarrow \epsilon$

[15]	$\langle \text{INSTRUCTION} \rangle$	$\rightarrow \langle \text{EXPRESSION} \rangle \langle \text{INSTRUCTION\_F} \rangle \text{ SEMICOLON}$
[16]		$\rightarrow \text{RETURN } \langle \text{EXPRESSION} \rangle \langle \text{INSTRUCTION\_F} \rangle \text{ SEMICOLON}$
[17]		$\rightarrow \text{LBRACE } \langle \text{INSTRUCTION\_LIST} \rangle \text{ RBRACE}$
[18]		$\rightarrow \langle \text{CONDITION} \rangle \langle \text{EXPRESSION} \rangle \text{ LBRACE } \langle \text{INSTRUCTION\_LIST} \rangle \text{ RBRACE } \langle \text{CONDITION\_END} \rangle$
[19]	$\langle \text{INSTRUCTION\_F} \rangle$	$\rightarrow \langle \text{CONDITION} \rangle \langle \text{EXPRESSION} \rangle$
[20]		$\rightarrow \epsilon$
[21]	$\langle \text{CONDITION} \rangle$	$\rightarrow \text{IF}$
[22]		$\rightarrow \text{UNLESS}$
[23]	$\langle \text{CONDITION\_END} \rangle$	$\rightarrow \text{ELSIF } \langle \text{EXPRESSION} \rangle \text{ LBRACE } \langle \text{INSTRUCTION\_LIST} \rangle \text{ RBRACE } \langle \text{CONDITION\_END} \rangle$
[24]		$\rightarrow \text{ELSE LBRACE } \langle \text{INSTRUCTION\_LIST} \rangle \text{ RBRACE}$
[25]		$\rightarrow \epsilon$
[26]	$\langle \text{EXPRESSION} \rangle$	$\rightarrow \langle \text{EXPRESSION\_TWO} \rangle \langle \text{EXPRESSION\_V} \rangle$
[27]	$\langle \text{EXPRESSION\_V} \rangle$	$\rightarrow \text{ASSIGN\_MARK } \langle \text{EXPRESSION\_TWO} \rangle \langle \text{EXPRESSION\_V} \rangle$
[28]		$\rightarrow \epsilon$
[29]	$\langle \text{EXPRESSION\_TWO} \rangle$	$\rightarrow \langle \text{EXPRESSION\_THREE} \rangle \langle \text{EXPRESSION\_TWO\_V} \rangle$
[30]	$\langle \text{EXPRESSION\_TWO\_V} \rangle$	$\rightarrow \text{LAZY\_OR } \langle \text{EXPRESSION\_THREE} \rangle \langle \text{EXPRESSION\_TWO\_V} \rangle$
[31]		$\rightarrow \epsilon$
[32]	$\langle \text{EXPRESSION\_THREE} \rangle$	$\rightarrow \langle \text{EXPRESSION\_FOUR} \rangle \langle \text{EXPRESSION\_THREE\_V} \rangle$
[33]	$\langle \text{EXPRESSION\_THREE\_V} \rangle$	$\rightarrow \text{LAZY\_AND } \langle \text{EXPRESSION\_FOUR} \rangle \langle \text{EXPRESSION\_THREE\_V} \rangle$
[34]		$\rightarrow \epsilon$
[35]	$\langle \text{EXPRESSION\_FOUR} \rangle$	$\rightarrow \langle \text{EXPRESSION\_FIVE} \rangle \langle \text{EXPRESSION\_FOUR\_V} \rangle$
[36]	$\langle \text{EXPRESSION\_FOUR\_V} \rangle$	$\rightarrow \langle \text{EXPRESSION\_FOUR\_F} \rangle \langle \text{EXPRESSION\_FIVE} \rangle$
[37]		$\rightarrow \epsilon$
[38]	$\langle \text{EXPRESSION\_FOUR\_F} \rangle$	$\rightarrow \text{DIFFERENT}$
[39]		$\rightarrow \text{EQ}$
[40]		$\rightarrow \text{EQUALS}$
[41]		$\rightarrow \text{NE}$
[42]	$\langle \text{EXPRESSION\_FIVE} \rangle$	$\rightarrow \langle \text{EXPRESSION\_SIX} \rangle \langle \text{EXPRESSION\_FIVE\_V} \rangle$
[43]	$\langle \text{EXPRESSION\_FIVE\_V} \rangle$	$\rightarrow \langle \text{EXPRESSION\_FIVE\_F} \rangle \langle \text{EXPRESSION\_SIX} \rangle$
[44]		$\rightarrow \epsilon$
[45]	$\langle \text{EXPRESSION\_FIVE\_F} \rangle$	$\rightarrow \text{GREATER}$
[46]		$\rightarrow \text{GREATER\_EQUALS}$
[47]		$\rightarrow \text{GT}$
[48]		$\rightarrow \text{GE}$
[49]		$\rightarrow \text{LOWER}$
[50]		$\rightarrow \text{LOWER\_EQUALS}$
[51]		$\rightarrow \text{LT}$
[52]		$\rightarrow \text{LE}$
[53]	$\langle \text{EXPRESSION\_SIX} \rangle$	$\rightarrow \langle \text{EXPRESSION\_SEVEN} \rangle \langle \text{EXPRESSION\_SIX\_V} \rangle$

[54]	$\langle \text{EXPRESSION\_SIX\_V} \rangle$	$\rightarrow \langle \text{EXPRESSION\_SIX\_F} \rangle \langle \text{EXPRESSION\_SEVEN} \rangle$
[55]		$\langle \text{EXPRESSION\_SIX\_V} \rangle$ $\rightarrow \epsilon$
[56]	$\langle \text{EXPRESSION\_SIX\_F} \rangle$	$\rightarrow \text{PLUS}$
[57]		$\rightarrow \text{MINUS}$
[58]		$\rightarrow \text{CONCAT\_MARK}$
[59]	$\langle \text{EXPRESSION\_SEVEN} \rangle$	$\rightarrow \langle \text{EXPRESSION\_EIGHT} \rangle \langle \text{EXPRESSION\_SEVEN\_V} \rangle$
[60]	$\langle \text{EXPRESSION\_SEVEN\_V} \rangle$	$\rightarrow \langle \text{EXPRESSION\_SEVEN\_F} \rangle \langle \text{EXPRESSION\_EIGHT} \rangle$
[61]		$\langle \text{EXPRESSION\_SEVEN\_V} \rangle$ $\rightarrow \epsilon$
[62]	$\langle \text{EXPRESSION\_SEVEN\_F} \rangle$	$\rightarrow \text{TIMES}$
[63]		$\rightarrow \text{DIVIDE}$
[64]	$\langle \text{EXPRESSION\_EIGHT} \rangle$	$\rightarrow \langle \text{EXPRESSION\_NINE} \rangle$
[65]		$\rightarrow \langle \text{EXPRESSION\_EIGHT\_F} \rangle \langle \text{EXPRESSION\_EIGHT} \rangle$
[66]	$\langle \text{EXPRESSION\_EIGHT\_F} \rangle$	$\rightarrow \text{NOT}$
[67]		$\rightarrow \text{PLUS}$
[68]		$\rightarrow \text{MINUS}$
[69]	$\langle \text{EXPRESSION\_NINE} \rangle$	$\rightarrow \text{LPAR} \langle \text{EXPRESSION} \rangle \text{RPAR}$
[70]		$\rightarrow \langle \text{SIMPLE\_EXPRESSION} \rangle$
[71]	$\langle \text{SIMPLE\_EXPRESSION} \rangle$	$\rightarrow \langle \text{FUNCTION\_CALL} \rangle$
[72]		$\rightarrow \text{INTEGER}$
[73]		$\rightarrow \text{FLOAT}$
[74]		$\rightarrow \text{STRING}$
[75]		$\rightarrow \text{VAR}$
[76]	$\langle \text{FUNCTION\_CALL} \rangle$	$\rightarrow \text{CALL\_MARK IDENTIFIER LPAR} \langle \text{ARGUMENT\_CALL\_LIST} \rangle$ $\text{RPAR}$
[77]	$\langle \text{ARGUMENT\_CALL\_LIST} \rangle$	$\rightarrow \langle \text{EXPRESSION} \rangle \langle \text{ARGUMENT\_CALL\_LIST\_V} \rangle$
[78]		$\rightarrow \epsilon$
[79]	$\langle \text{ARGUMENT\_CALL\_LIST\_V} \rangle$	$\rightarrow \text{COMMA} \langle \text{EXPRESSION} \rangle \langle \text{ARGUMENT\_CALL\_LIST\_V} \rangle$
[80]		$\rightarrow \epsilon$



### 3.2 *First*<sub>1</sub>

⟨PROGRAM⟩	: SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, IF, UNLESS, NOT, PLUS, MINUS
⟨PROGRAM_V⟩	: SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, IF, UNLESS, NOT, PLUS, MINUS, $\epsilon$
⟨PROGRAM_F⟩	: SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, IF, UNLESS, NOT, PLUS, MINUS
⟨FUNCTION⟩	: SUB
⟨FUNCTION_ARGUMENT⟩	: LPAR, $\epsilon$
⟨ARGUMENT_LIST⟩	: VAR, $\epsilon$
⟨ARGUMENT_LIST_V⟩	: COMMA, $\epsilon$
⟨INSTRUCTION_LIST⟩	: RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, IF, UNLESS, NOT, PLUS, MINUS, $\epsilon$
⟨FUNCTION_CALL⟩	: CALL_MARK
⟨ARGUMENT_CALL_LIST⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS, $\epsilon$
⟨ARGUMENT_CALL_LIST_V⟩	: COMMA, $\epsilon$
⟨INSTRUCTION⟩	: RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, IF, UNLESS, NOT, PLUS, MINUS
⟨INSTRUCTION_F⟩	: IF, UNLESS, $\epsilon$
⟨CONDITION⟩	: IF, UNLESS
⟨CONDITION_END⟩	: ELSIF, ELSE, $\epsilon$
⟨EXPRESSION⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_V⟩	: ASSIGN_MARK, $\epsilon$
⟨EXPRESSION_TWO⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_TWO_V⟩	: LAZY_OR, $\epsilon$
⟨EXPRESSION_THREE⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_THREE_V⟩	: LAZY_AND, $\epsilon$
⟨EXPRESSION_FOUR⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_FOUR_V⟩	: DIFFERENT, EQ, EQUALS, NE, $\epsilon$
⟨EXPRESSION_FOUR_F⟩	: DIFFERENT, EQ, EQUALS, NE
⟨EXPRESSION_FIVE⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_FIVE_V⟩	: GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, $\epsilon$
⟨EXPRESSION_FIVE_F⟩	: GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT

$\langle \text{EXPRESSION\_SIX} \rangle$	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, NOT, PLUS, MINUS
$\langle \text{EXPRESSION\_SIX\_V} \rangle$	: PLUS, MINUS, CONCAT\_MARK, $\epsilon$
$\langle \text{EXPRESSION\_SIX\_F} \rangle$	: PLUS, MINUS, CONCAT\_MARK
$\langle \text{EXPRESSION\_SEVEN} \rangle$	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, NOT, PLUS, MINUS
$\langle \text{EXPRESSION\_SEVEN\_V} \rangle$	: TIMES, DIVIDE, $\epsilon$
$\langle \text{EXPRESSION\_SEVEN\_F} \rangle$	: TIMES, DIVIDE
$\langle \text{EXPRESSION\_EIGHT} \rangle$	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, NOT, PLUS, MINUS
$\langle \text{EXPRESSION\_EIGHT\_F} \rangle$	: NOT, PLUS, MINUS
$\langle \text{EXPRESSION\_NINE} \rangle$	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK
$\langle \text{SIMPLE\_EXPRESSION} \rangle$	: INTEGER, FLOAT, STRING, VAR, CALL\_MARK

### 3.3 *Follow*<sub>1</sub>

$\langle \text{PROGRAM} \rangle$	: $\phi$
$\langle \text{PROGRAM\_V} \rangle$	: $\phi$
$\langle \text{PROGRAM\_F} \rangle$	: SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, IF, UNLESS, NOT, PLUS, MINUS
$\langle \text{FUNCTION} \rangle$	: SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, IF, UNLESS, NOT, PLUS, MINUS
$\langle \text{FUNCTION\_ARGUMENT} \rangle$	: LBRACE
$\langle \text{ARGUMENT\_LIST} \rangle$	: RPAR
$\langle \text{ARGUMENT\_LIST\_V} \rangle$	: RPAR
$\langle \text{INSTRUCTION\_LIST} \rangle$	: RBRACE
$\langle \text{FUNCTION\_CALL} \rangle$	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN\_MARK, LAZY\_OR, LAZY\_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER\_EQUALS, GT, LE, LOWER, LOWER\_EQUALS, LT, PLUS, MINUS, CONCAT\_MARK, TIMES, DIVIDE
$\langle \text{ARGUMENT\_CALL\_LIST} \rangle$	: RPAR
$\langle \text{ARGUMENT\_CALL\_LIST\_V} \rangle$	: RPAR
$\langle \text{INSTRUCTION} \rangle$	: RBRACE, SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, IF, UNLESS, NOT, PLUS, MINUS
$\langle \text{INSTRUCTION\_F} \rangle$	: SEMICOLON
$\langle \text{CONDITION} \rangle$	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, NOT, PLUS, MINUS
$\langle \text{CONDITION\_END} \rangle$	: RBRACE, SUB, RETURN, LBRACE, LPAR, INTEGER, FLOAT, STRING, VAR, CALL\_MARK, IF, UNLESS, NOT, PLUS, MINUS
$\langle \text{EXPRESSION} \rangle$	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON

⟨EXPRESSION_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON
⟨EXPRESSION_TWO⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK
⟨EXPRESSION_TWO_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK
⟨EXPRESSION_THREE⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND
⟨EXPRESSION_THREE_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND
⟨EXPRESSION_FOUR⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE
⟨EXPRESSION_FOUR_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE
⟨EXPRESSION_FOUR_F⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_FIVE⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT
⟨EXPRESSION_FIVE_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT
⟨EXPRESSION_FIVE_F⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_SIX⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK
⟨EXPRESSION_SIX_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK
⟨EXPRESSION_SIX_F⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_SEVEN⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK, TIMES, DIVIDE
⟨EXPRESSION_SEVEN_V⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK, TIMES, DIVIDE
⟨EXPRESSION_SEVEN_F⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS
⟨EXPRESSION_EIGHT⟩	: IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK, TIMES, DIVIDE
⟨EXPRESSION_EIGHT_F⟩	: LPAR, INTEGER, FLOAT, STRING, VAR, CALL_MARK, NOT, PLUS, MINUS

$\langle \text{EXPRESSION\_NINE} \rangle$	:	IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK, TIMES, DIVIDE
$\langle \text{SIMPLE\_EXPRESSION} \rangle$	:	IF, UNLESS, COMMA, LBRACE, RPAR, SEMICOLON, ASSIGN_MARK, LAZY_OR, LAZY_AND, DIFFERENT, EQ, EQUALS, NE, GE, GREATER, GREATER_EQUALS, GT, LE, LOWER, LOWER_EQUALS, LT, PLUS, MINUS, CONCAT_MARK, TIMES, DIVIDE

## 4 Table d'actions

	if	unless	else	elsif	sub	return	=	{	}	(	)	,	;	&		&&	!=	eq	==	ne	>	>=	gt	ge	<	<=	lt	le
<PROGRAM>	1	1			1		1	1	1	1	1			1														
<PROGRAM_V>	2	2			2		2	2	2	2	2			2														
<PROGRAM_F>	5	5			4		5	5	5	5	5			5														
<FUNCTION>					6																							
<FUNCTION_ARGUMENT>							8	8	7	7																		
<ARGUMENT_LIST>										10	10																	
<ARGUMENT_LIST_V>										12	11																	
<INSTRUCTION_LIST>	13	13				13	13	13	14	13				13														
<INSTRUCTION>	18	18				16	17	17	15	15				15														
<INSTRUCTION_F>	19	19											20															
<CONDITION>	21	22																										
<CONDITION_END>			24	23	25	25	25	25	25	25				25														
<SIMPLE_EXPRESSION>														71														
<FUNCTION_CALL>														76														
<ARGUMENT_CALL_LIST>										77	78			77														
<ARGUMENT_CALL_LIST_V>										80	79																	

	if	unless	else	elsif	sub	return	=	{	}	(	)	,	;	&		&&	!=	eq	==	ne	>	>=	gt	ge	<	<=	lt	le
<EXPRESSION>										26				26														
<EXPRESSION_V>	28	28					27	28		28	28	28		26														
<EXPRESSION_TWO>										29				29														
<EXPRESSION_TWO_V>	31	31					31	31		31	31	31			30													
<EXPRESSION_THREE>										32				32														
<EXPRESSION_THREE_V>	34	34					34	34		34	34	34			34	33												
<EXPRESSION_FOUR>										35				35														
<EXPRESSION_FOUR_V>	37	37					37	37		37	37	37			37	37	36	36	36	36								
<EXPRESSION_FOUR_F>																	38	39	40	41								
<EXPRESSION_FIVE>										42				42														
<EXPRESSION_FIVE_V>	44	44					44	44		44	44	44			44	44	44	44	44	44	43	43	43	43	43	43	43	43
<EXPRESSION_FIVE_F>																					45	46	47	48	49	50	51	52
<EXPRESSION_SIX>										53				53														
<EXPRESSION_SIX_V>	55	55					55	55		55	55	55			55	55	55	55	55	55	55	55	55	55	55	55	55	55
<EXPRESSION_SIX_F>																												
<EXPRESSION_SEVEN>										59				59														
<EXPRESSION_SEVEN_V>	61	61					61	61		61	61	61			61	61	61	61	61	61	61	61	61	61	61	61	61	
<EXPRESSION_SEVEN_F>																												
<EXPRESSION_EIGHT>										64				64														
<EXPRESSION_EIGHT_F>																												
<EXPRESSION_NINE>										69				70														

	+	-	.	*	/	!	id	integer	float	string	var
⟨PROGRAM⟩	1	1				1		1	1	1	1
⟨PROGRAM_V⟩	2	2				2		2	2	2	2
⟨PROGRAM_F⟩	5	5				5		5	5	5	5
⟨FUNCTION⟩											
⟨FUNCTION_ARGUMENT⟩											
⟨ARGUMENT_LIST⟩											9
⟨ARGUMENT_LIST_V⟩											
⟨INSTRUCTION_LIST⟩	13	13				13		13	13	13	13
⟨INSTRUCTION⟩	15	15				15		15	15	15	15
⟨INSTRUCTION_F⟩											
⟨CONDITION⟩											
⟨CONDITION_END⟩	25	25				25		25	25	25	25
⟨SIMPLE_EXPRESSION⟩								72	73	74	75
⟨FUNCTION_CALL⟩											
⟨ARGUMENT_CALL_LIST⟩	77	77				77		77	77	77	77
⟨ARGUMENT_CALL_LIST_V⟩											

	+	-	.	*	/	!	id	integer	float	string	var
<EXPRESSION>	26	26				26		26	26	26	26
<EXPRESSION_V>											
<EXPRESSION_TWO>	29	29				29		29	29	29	29
<EXPRESSION_TWO_V>											
<EXPRESSION_THREE>	32	32				32		32	32	32	32
<EXPRESSION_THREE_V>											
<EXPRESSION_FOUR>	35	35				35		35	35	35	35
<EXPRESSION_FOUR_V>											
<EXPRESSION_FOUR_F>											
<EXPRESSION_FIVE>	42	42				42		42	42	42	42
<EXPRESSION_FIVE_V>											
<EXPRESSION_FIVE_F>											
<EXPRESSION_SIX>	53	53				53		53	53	53	53
<EXPRESSION_SIX_V>	54	54	54								
<EXPRESSION_SIX_F>	56	57	58								
<EXPRESSION_SEVEN>	59	59				59		59	59	59	59
<EXPRESSION_SEVEN_V>	61	61	61	60	60						
<EXPRESSION_SEVEN_F>				62	63						
<EXPRESSION_EIGHT>	65	65				65		64	64	64	64
<EXPRESSION_EIGHT_F>	68	67				66					
<EXPRESSION_NINE>								70	70	70	70



remarques : 1) on a enlevé la possibilité d'omettre les () autour des listes d'arguments a l'appel d'une fonction, c'est pas LL(1) car le follow de expression prend le follow de toutes les expressions

2) On a supprimé le "not", car sa priorité faible engendre un comportement non-LL(1), le follow de expression prend le follow de toutes les expressions

3) La grammaire autorise l'assignation de n'importe quelle expression a une autre, c'est lors de l'analyse semantique qu'on determine ce qui est lvalue

4) On ne respecte pas les specificités du genre "0 but true"