INFO-F-403 : Language theory and compiling Rapport projet partie 2 - Grammaire

Simon Picard Arnaud Rosette

17 décembre 2014

1 Transformation de la grammaire donnée

1.1 Elimination des ambiguités

La première étape pour rendre la grammaire LL(1) fut l'élimination des ambiguités de la grammaire donnée. Pour cela, nous avons commencé par fixer la iorité et l'associativité des opérateurs.

2 Grammaire

[1]	<program></program>	\rightarrow	<InstructionList $>$
[2]	<InstructionList $>$	\rightarrow	<IdentifierInstruction $>$
			END_OF_INSTRUCTION <instructionlist></instructionlist>
[3]		\rightarrow	<constdefinition> END_OF_INSTRUCTION</constdefinition>
			<InstructionList $>$
[4]		\rightarrow	<block> END_OF_INSTRUCTION</block>
[=]			<instructionlist></instructionlist>
[5]		\rightarrow	<loop> END_OF_INSTRUCTION <instructionlist></instructionlist></loop>
[6]		\rightarrow	<pre><mstructionlist> <builtinfunctioncall> END_OF_INSTRUCTION</builtinfunctioncall></mstructionlist></pre>
[O]			<pre><instructionlist></instructionlist></pre>
[7]		\rightarrow	<pre><functiondefinition> END_OF_INSTRUCTION</functiondefinition></pre>
[.]		,	<instructionlist></instructionlist>
[8]		\rightarrow	END OF INSTRUCTION <instructionlist></instructionlist>
[9]		\rightarrow	ϵ
[10]	<IdentifierInstruction $>$	\rightarrow	IDENTIFIER < IdentifierInstructionTail>
[11]	<IdentifierInstructionTail $>$	\rightarrow	<assignationtail></assignationtail>
[12]		\rightarrow	TYPE_DEFINITION < Type>
[13]		\rightarrow	<functioncalltail></functioncalltail>
[14]	<AssignationTail $>$	\rightarrow	ASSIGNATION < Expression >
[15]		\rightarrow	COMMA IDENTIFIER <assignationtail> COMMA</assignationtail>
			<expression></expression>
[16]	<constdefinition></constdefinition>	\rightarrow	${\rm CONST\ IDENTIFIER}\ < Assignation Tail >$
[17]	<block></block>	\rightarrow	LET IDENTIFIER <assignationtail></assignationtail>
[4.0]	_		END_OF_INSTRUCTION <instructionlist> END</instructionlist>
[18]	<Loop $>$	\rightarrow	<if></if>
[19]		\rightarrow	WHILE <expression> END_OF_INSTRUCTION</expression>
[00]			<instructionlist> END</instructionlist>
[20]		\rightarrow	FOR IDENTIFIER ASSIGNATION < Expression >
[91]	<fortail></fortail>		TERNARY_ELSE <expression> <fortail> END OF INSTRUCTION <instructionlist> END</instructionlist></fortail></expression>
[21] [22]	Crof fair	ightarrow	TERNARY ELSE <expression></expression>
[44]		′	END OF INSTRUCTION <instructionlist> END</instructionlist>
[23]	<type></type>	\rightarrow	BOOLEAN_TYPE

[24]		\rightarrow	REAL TYPE
[25]		\rightarrow	INTEGER_TYPE
[26]	<Expression $>$	\rightarrow	$<\!$
[27]	<ternaryifexpression></ternaryifexpression>	\rightarrow	TERNARY_IF <expression> <ternaryelseexpression></ternaryelseexpression></expression>
[28]		\rightarrow	ϵ
[29]	<TernaryElseExpression $>$	$\stackrel{'}{ ightarrow}$	TERNARY ELSE < Expression >
[30]	<atomicexpression></atomicexpression>	\rightarrow	AtomicIdentifierExpression>
[31]	•	\rightarrow	INTEGER
[32]		\rightarrow	REAL
[33]		\rightarrow	BOOLEAN
[34]		\rightarrow	<BuiltInFunctionCall $>$
[35]	$<\!$	\rightarrow	IDENTIFIER
			$<\! Atomic Identifier Expression Tail \! >$
[36]	$<\! Atomic Identifier Expression Tail \!>$	\rightarrow	<FunctionCallTail $>$
[37]		\rightarrow	€
[38]	<UnaryExpression $>$	\rightarrow	NEGATION <unaryexpression></unaryexpression>
[39]	II D' NE	\rightarrow	<unarybitwisenotexpression></unarybitwisenotexpression>
[40]	$<\!\!\mathrm{UnaryBitwiseNotExpression}\!\!>$	\rightarrow	BITWISE_NOT <unarybitwisenotexpression></unarybitwisenotexpression>
[41]	II M' DI D	\rightarrow	<unaryminusplusexpression></unaryminusplusexpression>
[42]	$<\!\!\mathrm{UnaryMinusPlusExpression}\!\!>$	\rightarrow	MINUS <unaryminusplusexpression></unaryminusplusexpression>
[43]		\rightarrow	PLUS <unaryminusplusexpression></unaryminusplusexpression>
[44]	<Unary $AtomicExpression>$	$\overset{\rightarrow}{\rightarrow}$	<unaryatomicexpression> <atomicexpression></atomicexpression></unaryatomicexpression>
[45] $[46]$	< OnaryAtomicExpression>	\rightarrow	LEFT PARENTHESIS < Expression >
[40]			RIGHT PARENTHESIS
[47]	<binaryexpression></binaryexpression>	\rightarrow	<binarylazyorexpression></binarylazyorexpression>
[11]	(Billary Expression)	,	<binaryexpression'></binaryexpression'>
[48]	<binaryexpression'></binaryexpression'>	\rightarrow	LAZY_OR <binarylazyorexpression></binarylazyorexpression>
[-0]	(<binaryexpression'></binaryexpression'>
[49]		\rightarrow	ϵ
[50]	<binarylazyorexpression></binarylazyorexpression>	\rightarrow	<binarylazyandexpression></binarylazyandexpression>
			<binarylazyorexpression'></binarylazyorexpression'>
[51]	$<\!$	\rightarrow	$LAZY_AND < BinaryLazyAndExpression >$
			<BinaryLazyOrExpression' $>$
[52]		\rightarrow	ϵ
[53]	<BinaryLazyAndExpression $>$	\rightarrow	<binarynumericexpression></binarynumericexpression>
			<binarylazyandexpression'></binarylazyandexpression'>
[54]	<binarylazyandexpression'></binarylazyandexpression'>	\rightarrow	GREATER_THAN < BinaryNumericExpression>
[==1			<binarylazyandexpression'></binarylazyandexpression'>
[55]		\rightarrow	LESS_THAN <binarynumericexpression></binarynumericexpression>
[=0]			<binarylazyandexpression'> CREATER OR FOLIAL CONTRACT</binarylazyandexpression'>
[56]		\rightarrow	GREATER_OR_EQUALS_THAN
			<binarynumericexpression></binarynumericexpression>
[57]		,	<binarylazyandexpression'> LESS OR FOLIALS THAN</binarylazyandexpression'>
[57]		\rightarrow	LESS_OR_EQUALS_THAN <binarynumericexpression></binarynumericexpression>
			<binarynumericexpression'></binarynumericexpression'>
[58]		\rightarrow	EQUALITY <binarynumericexpression></binarynumericexpression>
[OO]		,	<pre><binarylazyandexpression'></binarylazyandexpression'></pre>
[59]		\rightarrow	INEQUALITY <binarynumericexpression></binarynumericexpression>
[00]		·	<binarylazyandexpression'></binarylazyandexpression'>
[60]		\rightarrow	ϵ
[61]	<binarynumericexpression></binarynumericexpression>	\rightarrow	<binarytermexpression></binarytermexpression>
	•		<binarynumericexpression'></binarynumericexpression'>
[62]	$<\!$	\rightarrow	PLUS <binarytermexpression></binarytermexpression>
			$<\!$

[63]		\rightarrow	${\rm MINUS} < \!\!{\rm BinaryTermExpression} \!\!>$
[64]		\rightarrow	<binarynumericexpression'> BITWISE OR <binarytermexpression></binarytermexpression></binarynumericexpression'>
[04]		/	<pre><binarynumericexpression'></binarynumericexpression'></pre>
[65]		\rightarrow	BITWISE_XOR <binarytermexpression></binarytermexpression>
[66]		\rightarrow	$<$ BinaryNumericExpression' $>$ ϵ
[67]	$<\!$	\rightarrow	<BinaryShiftedExpression $>$
[60]	D: E : b		<binarytermexpression'></binarytermexpression'>
[68]	<binarytermexpression'></binarytermexpression'>	\rightarrow	ARITHMETIC_SHIFT_LEFT <binaryshiftedexpression></binaryshiftedexpression>
			<binarytermexpression'></binarytermexpression'>
[69]		\rightarrow	ARITHMETIC_SHIFT_RIGHT
			<binaryshiftedexpression></binaryshiftedexpression>
[70]		\rightarrow	$<$ BinaryTermExpression'> ϵ
[71]	<BinaryShiftedExpression $>$	\rightarrow	<binaryfactorexpression></binaryfactorexpression>
			<BinaryShiftedExpression' $>$
[72]	$<\!$	\rightarrow	${\it TIMES} < {\it BinaryFactorExpression}>$
			<binaryshiftedexpression'></binaryshiftedexpression'>
[73]		\rightarrow	DIVIDE <binaryfactorexpression></binaryfactorexpression>
[= 4]			<pre><binaryshiftedexpression'></binaryshiftedexpression'></pre>
[74]		\rightarrow	REMAINDER <binaryfactorexpression></binaryfactorexpression>
[75]		\rightarrow	<pre><binaryshiftedexpression'> BITWISE AND <binaryfactorexpression></binaryfactorexpression></binaryshiftedexpression'></pre>
[10]		7	<pre></pre>
[76]		\rightarrow	INVERSE_DIVIDE <binaryfactorexpression></binaryfactorexpression>
			<binaryshiftedexpression'></binaryshiftedexpression'>
[77]		\rightarrow	ϵ
[78]	<BinaryFactorExpression $>$	\rightarrow	<unaryexpression></unaryexpression>
[70]	< Rinary Factor Expression?		<pre><binaryfactorexpression'> POWER < UnaryExpression></binaryfactorexpression'></pre>
[79]	<binaryfactorexpression'></binaryfactorexpression'>	\rightarrow	SinaryFactorExpression'>
[80]		\rightarrow	ϵ
[81]	<if></if>	\rightarrow	$IF < Expression > END_OF_INSTRUCTION$
			<InstructionList $>$ $<$ IfEnd $>$
[82]	<IfEnd $>$	\rightarrow	ELSE_IF <expression> END_OF_INSTRUCTION</expression>
[60]		,	<instructionlist> <ifend></ifend></instructionlist>
[83] [84]		$\stackrel{ o}{ o}$	ELSE <instructionlist> END END</instructionlist>
[85]	<builtinfunctioncall></builtinfunctioncall>	\rightarrow	READ REAL LEFT PARENTHESIS
[00]	Dunini diction car	,	RIGHT PARENTHESIS
[86]		\rightarrow	READ_INTEGER LEFT_PARENTHESIS
			RIGHT_PARENTHESIS
[87]		\rightarrow	INTEGER_CAST LEFT_PARENTHESIS < Expression >
[88]		\rightarrow	RIGHT_PARENTHESIS REAL_CAST LEFT_PARENTHESIS <expression></expression>
[66]		7	RIGHT PARENTHESIS
[89]		\rightarrow	BOOLEAN_CAST LEFT_PARENTHESIS <expression></expression>
			RIGHT_PARENTHESIS
[90]		\rightarrow	PRINTLN LEFT_PARENTHESIS <expression></expression>
[01]	Eurotian Call Tail	,	RIGHT_PARENTHESIS
[91]	<functioncalltail></functioncalltail>	\rightarrow	LEFT_PARENTHESIS < Parameter > RIGHT PARENTHESIS
[92]	<parameter></parameter>	\rightarrow	<expression> <parametertail></parametertail></expression>
[93]		\rightarrow	ϵ
[94]	<ParameterTail $>$	\rightarrow	COMMA < Expression > < Parameter Tail >
[95]		\rightarrow	ϵ

[96]	<FunctionDefinition $>$	\rightarrow	FUNCTION IDENTIFIER LEFT_PARENTHESIS <argument> RIGHT_PARENTHESIS <instructionlist> <functiondefinitionend></functiondefinitionend></instructionlist></argument>
[97]	<FunctionDefinitionEnd $>$	\rightarrow	$\operatorname{RETURN} < \operatorname{Expression} > \operatorname{END}$
[98]		\rightarrow	END
[99]	<argument></argument>	\rightarrow	IDENTIFIER TYPE_DEFINITION <type> <argumenttail></argumenttail></type>
[100]		\rightarrow	ϵ
[101]	<argumenttail></argumenttail>	\rightarrow	COMMA IDENTIFIER TYPE_DEFINITION <type> <argumenttail></argumenttail></type>
[102]		\rightarrow	ϵ

3 First et Follow set

Variable	First	Follow						
	BOOLEAN_CAST, PRINTLN, FOR EPSILON_VALUE							
TD.	INTEGER_CAST, FUNCTION							
<program></program>	END_OF_INSTRUCTION CONST, READ_INTEGER, LET							
	WHILE, IDENTIFIER							
	READ REAL, REAL CAST, IF							
	BOOLEAN CAST, PRINTLN, FOR							
	EPSILON VALUE							
	INTEGER CAST, FUNCTION							
<instructionlist></instructionlist>	END OF INSTRUCTION	RETURN, ELSE IF, ELSE, END						
	CONST, READ INTEGER, LET	, _ , ,						
	WHILE, IDENTIFIER							
	READ_REAL, REAL_CAST, IF							
<identifierinstruction></identifierinstruction>	IDENTIFIER	END_OF_INSTRUCTION						
	ASSIGNATION, COMMA							
<identifierinstructiontail></identifierinstructiontail>	LEFT_PARENTHESIS	END_OF_INSTRUCTION						
	TYPE_DEFINITION	907.57						
<assignationtail></assignationtail>	ASSIGNATION, COMMA	COMMA						
	, ,	END_OF_INSTRUCTION						
<constdefinition></constdefinition>	CONST LET	END_OF_INSTRUCTION						
<block></block>		END_OF_INSTRUCTION END_OF_INSTRUCTION						
<loop></loop>	FOR, WHILE, IF	END_OF_INSTRUCTION						
<fortail></fortail>	END_OF_INSTRUCTION TERNARY_ELSE	END_OF_INSTRUCTION						
<type></type>	INTEGER_TYPE	COMMA, RIGHT_PARENTHESIS						
-7 P	BOOLEAN_TYPE, REAL_TYPE	END_OF_INSTRUCTION						
	BOOLEAN_CAST							
	BITWISE_NOT, PRINTLN	COMMA						
	NEGATION, INTEGER_CAST BOOLEAN, MINUS	COMMA END OF INSTRUCTION						
<expression></expression>	LEFT PARENTHESIS, REAL	RIGHT PARENTHESIS						
	READ_INTEGER, IDENTIFIER	TERNARY ELSE, END						
	READ REAL, REAL CAST	TEIGNATUI _EEEEE, EINE						
	INTEGER, PLUS							
		COMMA						
(II) I(II)	TERNARY IF	END OF INSTRUCTION						
<ternaryifexpression></ternaryifexpression>	EPSILON_VALUE	RIGHT_PARENTHESIS						
		TERNARY_ELSE, END						

		C(O) 13 1 A
<ternaryelseexpression></ternaryelseexpression>	TERNARY_ELSE	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS TERNARY_ELSE, END
<AtomicExpression $>$	BOOLEAN_CAST, PRINTLN REAL, READ_INTEGER INTEGER_CAST, IDENTIFIER READ_REAL, REAL_CAST BOOLEAN, INTEGER	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
$<\!$	IDENTIFIER	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
< A tomic Identifier Expression Tail>	LEFT_PARENTHESIS EPSILON_VALUE	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS

<unaryexpression></unaryexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
$<\!$	BOOLEAN_CAST BITWISE_NOT, PRINTLN INTEGER_CAST, BOOLEAN MINUS, LEFT_PARENTHESIS REAL, READ_INTEGER IDENTIFIER, READ_REAL REAL_CAST, INTEGER, PLUS	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
<unaryminusplusexpression></unaryminusplusexpression>	BOOLEAN_CAST, PRINTLN INTEGER_CAST, BOOLEAN MINUS, LEFT_PARENTHESIS REAL, READ_INTEGER IDENTIFIER, READ_REAL REAL_CAST, INTEGER, PLUS	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS

$<\!$	LEFT_PARENTHESIS BOOLEAN_CAST, PRINTLN REAL, READ_INTEGER INTEGER_CAST, IDENTIFIER READ_REAL, REAL_CAST BOOLEAN, INTEGER	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
<binaryexpression></binaryexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS TERNARY_IF, TERNARY_ELSE END
<binaryexpression'></binaryexpression'>	EPSILON_VALUE, LAZY_OR	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS TERNARY_IF, TERNARY_ELSE END
<binarylazyorexpression></binarylazyorexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS TERNARY_IF, TERNARY_ELSE END, LAZY_OR
<binarylazyorexpression'></binarylazyorexpression'>	LAZY_AND, EPSILON_VALUE	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS TERNARY_IF, TERNARY_ELSE END, LAZY_OR
<binarylazyandexpression></binarylazyandexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS LAZY_AND, TERNARY_IF TERNARY_ELSE, END, LAZY_OR
<binarylazyandexpression'></binarylazyandexpression'>	LESS_OR_EQUALS_THAN GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, INEQUALITY EPSILON_VALUE, LESS_THAN	COMMA END_OF_INSTRUCTION RIGHT_PARENTHESIS LAZY_AND, TERNARY_IF TERNARY_ELSE, END, LAZY_OR

		T T T T T T T T T T T T T T T T T T T
<binarynumericexpression></binarynumericexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	LESS_OR_EQUALS_THAN COMMA, RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, LESS_THAN LAZY_OR END_OF_INSTRUCTION GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, LAZY_AND, END
$<\!$	BITWISE_OR EPSILON_VALUE BITWISE_XOR, PLUS, MINUS	LESS_OR_EQUALS_THAN COMMA, RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, LESS_THAN LAZY_OR END_OF_INSTRUCTION GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, LAZY_AND, END
<binarytermexpression></binarytermexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	LESS_OR_EQUALS_THAN COMMA, BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, LESS_THAN LAZY_OR, BITWISE_XOR MINUS END_OF_INSTRUCTION GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, LAZY_AND, END PLUS
<binarytermexpression'></binarytermexpression'>	ARITHMETIC_SHIFT_LEFT EPSILON_VALUE ARITHMETIC_SHIFT_RIGHT	LESS_OR_EQUALS_THAN COMMA, BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, LESS_THAN LAZY_OR, BITWISE_XOR MINUS END_OF_INSTRUCTION GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, LAZY_AND, END PLUS
$<\!$	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	LESS_OR_EQUALS_THAN COMMA ARITHMETIC_SHIFT_LEFT BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, LESS_THAN LAZY_OR, BITWISE_XOR MINUS END_OF_INSTRUCTION GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, LAZY_AND, END ARITHMETIC_SHIFT_RIGHT PLUS

$<\!$	INVERSE_DIVIDE, TIMES REMAINDER, EPSILON_VALUE BITWISE_AND, DIVIDE	LESS_OR_EQUALS_THAN COMMA ARITHMETIC_SHIFT_LEFT BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, LESS_THAN LAZY_OR, BITWISE_XOR MINUS END_OF_INSTRUCTION GREATER_THAN GREATER_OR_EQUALS_THAN EQUALITY, LAZY_AND, END ARITHMETIC_SHIFT_RIGHT PLUS
<binaryfactorexpression></binaryfactorexpression>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, INTEGER_CAST BOOLEAN, MINUS LEFT_PARENTHESIS, REAL READ_INTEGER, IDENTIFIER READ_REAL, REAL_CAST INTEGER, PLUS	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
<binaryfactorexpression'></binaryfactorexpression'>	POWER, EPSILON_VALUE	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
<if></if>	IF	END OF INSTRUCTION
<ifend></ifend>	ELSE_IF, ELSE, END	END_OF_INSTRUCTION

$<\! \rm Built In Function Call \!>$	BOOLEAN_CAST, PRINTLN READ_INTEGER INTEGER_CAST, READ_REAL REAL_CAST	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
<functioncalltail></functioncalltail>	LEFT_PARENTHESIS	LESS_OR_EQUALS_THAN INVERSE_DIVIDE BITWISE_OR RIGHT_PARENTHESIS INEQUALITY, TERNARY_IF TERNARY_ELSE, DIVIDE MINUS, GREATER_THAN LAZY_AND ARITHMETIC_SHIFT_RIGHT COMMA ARITHMETIC_SHIFT_LEFT TIMES, POWER, BITWISE_AND LESS_THAN, LAZY_OR BITWISE_XOR, REMAINDER END_OF_INSTRUCTION GREATER_OR_EQUALS_THAN EQUALITY, END, PLUS
<parameter></parameter>	BOOLEAN_CAST BITWISE_NOT, PRINTLN NEGATION, EPSILON_VALUE INTEGER_CAST, BOOLEAN MINUS, LEFT_PARENTHESIS REAL, READ_INTEGER IDENTIFIER, READ_REAL REAL_CAST, INTEGER, PLUS	RIGHT_PARENTHESIS
<parametertail></parametertail>	COMMA, EPSILON VALUE	RIGHT PARENTHESIS
<functiondefinition></functiondefinition>	FUNCTION	END OF INSTRUCTION
<functiondefinitionend></functiondefinitionend>	RETURN, END	END OF INSTRUCTION
<argument></argument>	EPSILON_VALUE IDENTIFIER	RIGHT_PARENTHESIS
	COMMA, EPSILON VALUE	RIGHT PARENTHESIS

4 Action Table

RETURN	1	6		1				_		П		_	Т				П		П	_	П			_	Т	1		\top	Т	П	_	26	
FUNCTION	_	~								Н				П																	8	3 0,	
PRINTLN	-	9				П			26	П	č	ئ ⁴		33	4;	44 44 44 44 44 44 44 45 45 45 45 45 45 4	47	55		3	61	67	5 1	7	28			18		92	Ť		
BOOLEAN CAST	-	9				П		T	56	П	-	,		39	ij.	4 3	47,		i c	3	91	12	5 1	₹	182			68	T	928	\top		\Box
REAL CAST	-	9						\top	26 26	П	-	54 54		39	₫;	4 3	7	50.50		3	91	77	5 1	1/1/	182			8		92			
INTEGER_CAST	-	9							26 26	П	-	44 44 44		39	4;	4 5	47	- 12	[2	3	61 61 61 61	79797979	5 1	₹	120			8687888990		92 92 92 92 92			
READ_INTEGER	_	9							26	П	2	4		33	₽;	44	47	20		3	61	2	5 1	7	200			98		92			
READ_REAL	-	9							26	П	-	4		33	4	44	47	55		3	61	2	5 1	7	22			23		92			
INTEGER TYPE								25		П					Ì										1			1					
REAL TYPE								24		П					П																		
BOOLEAN TYPE								ध		П					П																		
LET	-	4				17				П					П																		
ELSE		6						\top		П					П												o o	3					
ELSE IF		6				П		T		П				П	П		П	\top			П		\top				300	ĭ					\Box
IF	-	ಬ					<u>∞</u>			П					П												8						
POWER										П			37		П											62							
INVERSE DIVIDE						П		T		П			37.		П									192		8							
BITWISE AND			\top					1		П			3737	П			П			1	П		\top	75		8080			T		\top		\Box
REMAINDER	T					П		\top		Н			<u> </u>	П	H			\top		1			\top	4		18 S			T				
DIVIDE								+		Н			3737	Н	H			\top						1		8080							
TIMES	\vdash							+		Н			373	Н	H					1		+	+	2		808			+		_		
ARITHMETIC SHIFT RIGHT	+	\vdash	+	+	\vdash	H	+	+	1	Н	+	+	73	Н	\vdash	+	\forall	+	++	+	\vdash	+	<u> 5</u>	777	+	180	+	+	+	\vdash	+	+	+
ARITHMETIC SHIFT LEFT	+	\vdash	+	+	\vdash	\vdash	+	+		\vdash	+	+	373737	\vdash	\vdash	+	\vdash	+	++	+	\vdash	+	698902	777	+	180	+	+	+	\vdash	+	+	++
BITWISE XOR	+		+	+	\vdash	\vdash	+	+	1	Н	+	+	73	\vdash	\vdash	+	\vdash	+		+	1	ᆉ	190	777	+	8	\vdash	+	+	\vdash	+	+	++
BITWISE OR	+	\vdash	+	+	\vdash	\vdash	+	+	\vdash	$\vdash \vdash$	+	+	73	\vdash	\vdash	+	\vdash	+	++	+	 	유	707	4	+	180	+	+	+	\vdash	+	+	+
INEQUALITY	-	\vdash	+	-	+	\vdash	+	+	-	Н		+	737	Н	\vdash	+	\vdash	+	++	16		<u>e </u>	0	4	+	100	\vdash	+	+	\vdash	+		++
EQUALITY EQUALITY	\vdash	\vdash	+	+	\vdash	\vdash	+	+	-	$\vdash \vdash$	+	+	73	Н	\vdash	+	\vdash	+	++	8 59	- 3	9	12	777	+	180	\vdash	+	\vdash	\vdash	+	+	++
	+		+	+	\vdash	H	+	+		Н	+	+	73	\vdash	\vdash	+	\vdash	+	++	57 58		<u>ĕ</u>	70 70 70	777	+	18	\vdash	+	+	H	+	+	++
LESS_OR_EQUALS_THAN	T		+	+	\vdash	\vdash	+	+	-	$\vdash \vdash$	+	+	37373737	Н	\vdash	\perp		+	++	35		00 00 00 00 00 00 00 00 00 00 00	12	777	+	80808080808080808080	\vdash	+	+	\vdash	+	+	++
GREATER_OR_EQUALS_THAN			-				_	+		Ш		_	3,				\vdash	_		55 56		<u>ğ</u>	2	1		$ \infty $		-	-		_		\vdash
LESS_THAN										Ш			37							<u> </u>	3	<u>ğ </u>	70 70	77	_	$\frac{8}{8}$							
GREATER_THAN								\perp		Ш			373737		Ш					60 60 54	8	8	2	12	1	8			╙				
LAZY_AND	╙		_					\perp		Ш			37		Ш				52 51	18		월	70 70	12	:	18			╙		_		
LAZY_OR										Ш			37					84	22	_ 8		8 _	2	12		8080							
RIGHT_PARENTHESIS										8			37					49	52	09	8	ရှု	70	12	:	18				93	18		100
LEFT PARENTHESIS	+		~	,				+	56				9	6	٠,	44 44 44 42 42 43 44 45 45 45 45	<u>-</u>	25			=	1	200000000000000000000000000000000000000	= -	100				91		-		
PLUS	+		+	+-			+	+	62	Н	-	+	12	93	1.4	ঠ ক ক	7	-15	52 53 53 53 53 53 53 53 53 53 53 53 53 53	3	61 61 61 61	7 1		<u> </u>	. 00	0		-	100	92929292929292	+		++
MINUS	\vdash		_	+	\vdash	Н		+	26 26	Н	+		12	6	4	21	7	7. 07.	1 2	-	9 5	2 5	0	1/2	8	180		+	+	29	_	_	
BITWISE NOT	-		-			H		+	262	Н		-	100	93	0	4	7	50	, c	3	16	2 0	- 1	- 1	00	00			\vdash	29	-		
NEGATION	\vdash		-				-	+	262	Н		_	+	8	4		7	50.5	C	-	16	12	-	+	2				+	29	+		++
BOOLEAN	+		_	-	\vdash	Н	_	+	62	Н	c	2	-	93	_,	4 10	7	50.2	, c	-	616161	12	- -	+	8			+	+	29	-	-	
REAL	+		_	_				+	26 26	Н	0	5255	+	93	14,	4 70 4 4	7	50.5	1 20	5	16	12	- !	+	2			-	+	29	-	_	
	\vdash		-				-	+	262	Н	- 0	<u>3</u> -	+	33	4	4 4	7	-15	1 2	5	9	<u> </u>	<u> </u>	_	787				\vdash	5	-		++
INTEGER TERNARY IF			_		\vdash		_	+			Ċ	٠ <u>۲</u>		ಣ	4	4 4	4	<u> </u>			9	6	را اد	<u> </u>	1			-	-	6	_		
	-		_				_	+		27		-	ಣ	Н	\vdash			49	52	09		8	12	4	_	80			\vdash		_		Н.
COMMA			F	15						8			37					49 49	25	8	8	8	2	4		8					8		101
TERNARY_ELSE							2	1		8	23		37				!	4	25	8	18	8	2	1		8							
FOR.	-	ಬ					2																										
WHILE INTEGER_TYPE REAL_TYPE BOOLEAN_TYPE	-	ಬ					61			П																							
INTEGER TYPE								23 24 25		П																							
REAL TYPE			\top					4		П				П	П		\Box				\Box		\top						T		\top		\top
BOOLEAN TYPE								<u> </u>		П					\Box																		
END OF INSTRUCTION	-	∞					21	: -		28			37				!	49	52	99	1	8	2	1		8							
END_OF_INSTRUCTION ASSIGNATION	T	\vdash	=	14		\vdash	1,	+			1	\top	1.0	Н	H		H	4.	1	1	Ħ	+	1.	1	+	1	\vdash		T	\Box	+		+
END	+	6	+	_	\vdash	\vdash	+	+		82	\dashv	+	37	\vdash	\vdash		<u> </u>	64	52	199	9	8	12	12	+	08	2	<u>, </u>	+	\forall	+	86	+
CONST	+	က	+	+	16	\vdash	+	+		104	+	+	100	Н	\vdash	+	H	4	120	+9	H	+	1-	+	+	100	H°	+	+	\vdash	+	100	++
TYPE_DEFINITION	+		5	1	Н	\vdash	+	+	1	$\vdash \vdash$	+	+	+	Н	\vdash	+	\vdash	+	++	+	\vdash	+	++	+	+		+	+	+	\vdash	+	+	++
IDENTIFIER	-	2		1	\vdash	H	+	+	56	Н		35.5	+	39	 	44	47	55	, L	+	19	67	. ,	7	2/2		+	+	+	92	+	+	
EDGILON VALUE			7					+			- c	<u>ی ا د</u>	- 1	<u>س</u>	4,	4 4									-	_			+	8	,_		181:
EPSILON_VALUE	П	6								82			37					49	52	99	Ì	8	2	12		80				93	95		100 99
												١.																					
			- 17								^		la:		덻					되슬		/ا2	$ _{\Lambda} $	۸ إ ۲	J۸	Λ							
				3								^ :월		_	·ឆ្គ		$ \cdot $	⊼ I.5			ļ. <u>ģ</u> .			5 5	l E	'n.			١,		- 17	ړا⊊	
		A	.티틴.							<u>[</u>	Sic	= S	Š		re lie	res SSi	[달]	i S	iss	es es	ess	SS	Sic.	SS . E	SSi	SSi		<u></u>			\wedge	引卤	1 1,
		ist	<u>5</u> .	[E]	ioi					SS	res	ğ 🖁	i e	isi	윘	Z N	<u> </u>		pre 2	취취	ם		i es l	<u> </u>	ile	ire		n U	La		;; <u>5</u> ;		والما
	Ħ	<pre><instructionlist< pre=""></instructionlist<></pre>	<identifierinstruction></identifierinstruction>	<assignation tail=""></assignation>	< Const Definition	ا ي	Ş - <u>-</u>		<expression< td=""><td>TernaryIfExpression></td><td>TernaryElseExpression</td><td><atomicexpression> omicIdentifierExpress</atomicexpression></td><td></td><td><unaryexpression></unaryexpression></td><td>티핓</td><td>퓕뚾</td><td><binaryexpression></binaryexpression></td><td>< BinaryExpression? inaryLazyOrExpress</td><td></td><td>10</td><td>原</td><td>binaryivumencexpressior / Binary:TermExpression</td><td>Sinary TermExpression'</td><td></td><td>< BinaryFactorExpression</td><td><binaryfactorexpression?< td=""><td>_</td><td>SultInFunctionCall</td><td><functioncalltail< p=""></functioncalltail<></td><td><parameter< td=""><td><parameter tail=""></parameter></td><td>Function Definition End</td><td><argument></argument></td></parameter<></td></binaryfactorexpression?<></td></expression<>	TernaryIfExpression>	TernaryElseExpression	<atomicexpression> omicIdentifierExpress</atomicexpression>		<unaryexpression></unaryexpression>	티핓	퓕뚾	<binaryexpression></binaryexpression>	< BinaryExpression? inaryLazyOrExpress		10	原	binaryivumencexpressior / Binary:TermExpression	Sinary TermExpression'		< BinaryFactorExpression	<binaryfactorexpression?< td=""><td>_</td><td>SultInFunctionCall</td><td><functioncalltail< p=""></functioncalltail<></td><td><parameter< td=""><td><parameter tail=""></parameter></td><td>Function Definition End</td><td><argument></argument></td></parameter<></td></binaryfactorexpression?<>	_	SultInFunctionCall	<functioncalltail< p=""></functioncalltail<>	<parameter< td=""><td><parameter tail=""></parameter></td><td>Function Definition End</td><td><argument></argument></td></parameter<>	<parameter tail=""></parameter>	Function Definition End	<argument></argument>
	< Program	iti(lns H	žį į)eff	<block></block>	< Loop>	<type></type>	i së	岡	ĘĘ.		J.E			뤼팅	헸			nd	ļχį	티면	B	뒭튢	밥	ΞĮ	~IfFnd√		$\frac{n}{n}$	μĘ		를 I램	<u> </u>
	Įį̃.	밀	<u>-</u>	ilis	녣	岡	귀녆	Œ	ρľ	뗈	<u>a</u> [3 E	ifi	M	se	힘물	E	된[2		γ	lue			<u> </u>	ict	ctc	∨ ≝		Įį.	ara			rgı
		St		Sig	Suc	$ \vee $	× 7	T^{\vee}	囹	[평	ury.		ät	ar	[t]	¥III	lar		La	jazi j	<u> </u> -			2 5	Fa.	Fa		/ [략	nG	βį	ar i		4
		븬	<u></u>	¥					Γ^{\vee}		rn3		ĬĞ	띮				: []		Y.			[g]		g 5	ary ,			Fu		ئِ اِ اِ	1Ct	
		\ \	린분	} ^	$ \vee $					뗈] <u>F</u>	7 8	11:21	ľV	띪	ng ng	[🗸 [⊽I.≌	na	ıar	na	<u>ئا ا</u> ية	ij <u>ij</u> .	≝ ≗	ij.	Įį.		ΙÄ	IV		* 	기골	
							- 1																										
			_ \ \f	7							V	1	Ö			518		14				[]	7 7			\mathbb{R}		\				ľV	
				7								AtomicIdentifierExpression>	<atomicidentifierexpressiontail></atomicidentifierexpressiontail>		<unarybitwisenotexpression< td=""><td><unaryminusplusexpression <unaryatomicexpression></unaryatomicexpression></unaryminusplusexpression </td><td></td><td>< BinaryExpression > < BinaryLazyOrExpression</td><td><binarylazyorexpression></binarylazyorexpression></td><td>< Binary Lazy And Expression?</td><td><binarynumericexpression></binarynumericexpression></td><td>< binaryivumencexpression / RinaryTermExpression /</td><td></td><td>< SinaryShiftedExpression?</td><td></td><td></td><td></td><td> \</td><td></td><td></td><td></td><td></td><td></td></unarybitwisenotexpression<>	<unaryminusplusexpression <unaryatomicexpression></unaryatomicexpression></unaryminusplusexpression 		< BinaryExpression > < BinaryLazyOrExpression	<binarylazyorexpression></binarylazyorexpression>	< Binary Lazy And Expression?	<binarynumericexpression></binarynumericexpression>	< binaryivumencexpression / RinaryTermExpression /		< SinaryShiftedExpression?				\					