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

Simon Picard
Arnaud Rosette

17 décembre 2014

1 Grammaire

[1]	<Program>	→	<InstructionList>
[2]	<InstructionList>	→	<IdentifierInstruction> END_OF_INSTRUCTION <InstructionList>
[3]		→	<ConstDefinition> END_OF_INSTRUCTION <InstructionList>
[4]		→	<Block> END_OF_INSTRUCTION <InstructionList>
[5]		→	<Loop> END_OF_INSTRUCTION <InstructionList>
[6]		→	<BuiltInFunctionCall> END_OF_INSTRUCTION <InstructionList>
[7]		→	<FunctionDefinition> END_OF_INSTRUCTION <InstructionList>
[8]		→	END_OF_INSTRUCTION <InstructionList>
[9]		→	EPSILON_VALUE
[10]	<IdentifierInstruction>	→	IDENTIFIER <IdentifierInstructionTail>
[11]	<IdentifierInstructionTail>	→	<AssignationTail>
[12]		→	TYPE_DEFINITION <Type>
[13]		→	<FunctionCallTail>
[14]	<AssignationTail>	→	ASSIGNATION <Expression>
[15]		→	COMMA IDENTIFIER <AssignationTail> COMMA <Expression>
[16]	<ConstDefinition>	→	CONST IDENTIFIER <AssignationTail>
[17]	<Block>	→	LET IDENTIFIER <AssignationTail> END_OF_INSTRUCTION <InstructionList> END

[18]	<Loop>	→	<If>
[19]		→	WHILE <Expression> END_OF_INSTRUCTION <InstructionList> END
[20]		→	FOR IDENTIFIER ASSIGNATION <Expression> TERNARY_ELSE <Expression> <ForTail>
[21]	<ForTail>	→	END_OF_INSTRUCTION <InstructionList> END
[22]		→	TERNARY_ELSE <Expression> END_OF_INSTRUCTION <InstructionList> END
[23]	<Type>	→	BOOLEAN_TYPE
[24]		→	REAL_TYPE
[25]		→	INTEGER_TYPE
[26]	<Expression>	→	<BinaryExpression> <TernaryIfExpression>
[27]	<TernaryIfExpression>	→	TERNARY_IF <Expression> <TernaryElseExpression>
[28]		→	EPSILON_VALUE
[29]	<TernaryElseExpression>	→	TERNARY_ELSE <Expression>
[30]	<AtomicExpression>	→	<AtomicIdentifierExpression>
[31]		→	INTEGER
[32]		→	REAL
[33]		→	BOOLEAN
[34]		→	<BuiltInFunctionCall>
[35]	<AtomicIdentifierExpression>	→	IDENTIFIER <AtomicIdentifierExpressionTail>
[36]	<AtomicIdentifierExpressionTail>	→	<FunctionCallTail>
[37]		→	EPSILON_VALUE
[38]	<UnaryExpression>	→	NEGATION <UnaryExpression>
[39]		→	<UnaryBitwiseNotExpression>
[40]	<UnaryBitwiseNotExpression>	→	BITWISE_NOT <UnaryBitwiseNotExpression>
[41]		→	<UnaryMinusPlusExpression>
[42]	<UnaryMinusPlusExpression>	→	MINUS <UnaryMinusPlusExpression>
[43]		→	PLUS <UnaryMinusPlusExpression>
[44]		→	<UnaryAtomicExpression>
[45]	<UnaryAtomicExpression>	→	<AtomicExpression>
[46]		→	LEFT_PARENTHESIS <Expression> RIGHT_PARENTHESIS
[47]	<BinaryExpression>	→	<BinaryLazyOrExpression> <BinaryExpression'>
[48]	<BinaryExpression'>	→	LAZY_OR <BinaryLazyOrExpression> <BinaryExpression'>
[49]		→	EPSILON_VALUE
[50]	<BinaryLazyOrExpression>	→	<BinaryLazyAndExpression> <BinaryLazyOrExpression'>

[51]	<BinaryLazyOrExpression'>	→	LAZY_AND <BinaryLazyAndExpression> <BinaryLazyOrExpression'>
[52]		→	EPSILON_VALUE
[53]	<BinaryLazyAndExpression>	→	<BinaryNumericExpression> <BinaryLazyAndExpression'>
[54]	<BinaryLazyAndExpression'>	→	GREATER_THAN <BinaryNumericExpression> <BinaryLazyAndExpression'>
[55]		→	LESS_THAN <BinaryNumericExpression> <BinaryLazyAndExpression'>
[56]		→	GREATER_OR_EQUALS_THAN <BinaryNumericExpression> <BinaryLazyAndExpression'>
[57]		→	LESS_OR_EQUALS_THAN <BinaryNumericExpression> <BinaryLazyAndExpression'>
[58]		→	EQUALITY <BinaryNumericExpression> <BinaryLazyAndExpression'>
[59]		→	INEQUALITY <BinaryNumericExpression> <BinaryLazyAndExpression'>
[60]		→	EPSILON_VALUE
[61]	<BinaryNumericExpression>	→	<BinaryTermExpression> <BinaryNumericExpression'>
[62]	<BinaryNumericExpression'>	→	PLUS <BinaryTermExpression> <BinaryNumericExpression'>
[63]		→	MINUS <BinaryTermExpression> <BinaryNumericExpression'>
[64]		→	BITWISE_OR <BinaryTermExpression> <BinaryNumericExpression'>
[65]		→	BITWISE_XOR <BinaryTermExpression> <BinaryNumericExpression'>
[66]		→	EPSILON_VALUE
[67]	<BinaryTermExpression>	→	<BinaryShiftedExpression> <BinaryTermExpression'>
[68]	<BinaryTermExpression'>	→	ARITHMETIC_SHIFT_LEFT <BinaryShiftedExpression> <BinaryTermExpression'>
[69]		→	ARITHMETIC_SHIFT_RIGHT <BinaryShiftedExpression> <BinaryTermExpression'>
[70]		→	EPSILON_VALUE
[71]	<BinaryShiftedExpression>	→	<BinaryFactorExpression> <BinaryShiftedExpression'>

[72]	<BinaryShiftedExpression'>	→	TIMES <BinaryFactorExpression> <BinaryShiftedExpression'>
[73]		→	DIVIDE <BinaryFactorExpression> <BinaryShiftedExpression'>
[74]		→	REMAINDER <BinaryFactorExpression> <BinaryShiftedExpression'>
[75]		→	BITWISE_AND <BinaryFactorExpression> <BinaryShiftedExpression'>
[76]		→	INVERSE_DIVIDE <BinaryFactorExpression> <BinaryShiftedExpression'>
[77]		→	EPSILON_VALUE
[78]	<BinaryFactorExpression>	→	<UnaryExpression> <BinaryFactorExpression'>
[79]	<BinaryFactorExpression'>	→	POWER <UnaryExpression> <BinaryFactorExpression'>
[80]		→	EPSILON_VALUE
[81]	<If>	→	IF <Expression> END_OF_INSTRUCTION <InstructionList> <IfEnd>
[82]	<IfEnd>	→	ELSE_IF <Expression> END_OF_INSTRUCTION <InstructionList> <IfEnd>
[83]		→	ELSE <InstructionList> END
[84]		→	END
[85]	<BuiltInFunctionCall>	→	READ_REAL LEFT_PARENTHESIS RIGHT_PARENTHESIS
[86]		→	READ_INTEGER LEFT_PARENTHESIS RIGHT_PARENTHESIS
[87]		→	INTEGER_CAST LEFT_PARENTHESIS <Expression> RIGHT_PARENTHESIS
[88]		→	REAL_CAST LEFT_PARENTHESIS <Expression> RIGHT_PARENTHESIS
[89]		→	BOOLEAN_CAST LEFT_PARENTHESIS <Expression> RIGHT_PARENTHESIS
[90]		→	PRINTLN LEFT_PARENTHESIS <Expression> RIGHT_PARENTHESIS
[91]	<FunctionCallTail>	→	LEFT_PARENTHESIS <Parameter> RIGHT_PARENTHESIS
[92]	<Parameter>	→	<Expression> <ParameterTail>
[93]		→	EPSILON_VALUE
[94]	<ParameterTail>	→	COMMA <Expression> <ParameterTail>
[95]		→	EPSILON_VALUE

[96]	<FunctionDefinition>	→	FUNCTION IDENTIFIER LEFT_PARENTHESIS <Argument> RIGHT_PARENTHESIS <InstructionList> <FunctionDefinitionEnd>
[97]	<FunctionDefinitionEnd>	→	RETURN <Expression> END
[98]		→	END
[99]	<Argument>	→	IDENTIFIER TYPE_DEFINITION <Type> <ArgumentTail>
[100]		→	EPSILON_VALUE
[101]	<ArgumentTail>	→	COMMA IDENTIFIER TYPE_DEFINITION <Type> <ArgumentTail>
[102]		→	EPSILON_VALUE