INFO-F403 Introduction to Language Theory and Compilation

Chapeaux Thomas
Dagnely Pierre

February 24, 2013

Lexical units	regular expressions
INT	(UNARY+ + UNARY-)(0+1+2+3+4+5+6+7+8+9)*
FLOAT	(UNARY+ + UNARY-)(0+1+2+3+4+5+6+7+8+9)*.DOT.(0+1+2+3+4+5+6+7+8+9)*
BOOL	(0+1)
STRING (?)	'Σ'
VALUE	INT + FLOAT + BOOL + STRING
VARIABLE	$STRING
OPERATOR	! + * + / + - + +
OPERATOR-COMP	< + > + <= + >= + == + != + && + || + not + lt + gt + le + ge + eq + ne
UNARY+	+
UNARY-	-
EQUAL	=
DOT	.
SEMICOLON	;
COMA	,
AND	&
OPEN-PAR	(
CLOSE-PAR	)
OPEN-BRAC	{
CLOSE-BRAC	}
DOLLAR	$
OPEN-COND	IF
CLOSE-COND	ELSE
ADD-COND	ELSE IF
NEG-COND	UNLESS
RET	return
FUNCT-ID	SUB
FUNCT-NAME (?)	string
FUNCT-CALL	&STRING
PERL-FUNCT-NAME	defined + int + length + scalar + substr + scalar + substr + print
COMM	#.STRING

EXPRESSION	→ VARIABLE
	→ EXPRESSION OPERATOR EXPRESSION
	→ EXPRESSION-COMP
EXPRESSION-COMP	→ EXPRESSION OPERATOR-COMP EXPRESSION
ASSIGNATION	→ VARIABLE EQUAL VALUE
	→ VARIABLE EQUAL EXPRESSION
CONDITION	→ OPEN-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS CLOSE-BRAC CONDITION-EN
	→ NEG-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS CLOSE-BRAC CONDITION-ENI
	→ EXPRESSION OPEN-COND EXPRESSION-COND
	→ EXPRESSION NEG-COND EXPRESSION-COND
CONDITION-END	→ ADD-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS CLOSE-BRAC
	→ ADD-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS CLOSE-BRAC CONDITION-ENI
	→ CLOSE-COND EXPRESSION-COND OPEN-BRAC INSTRUCTIONS CLOSE-BRAC
	→ EPSILON
INSTRUCTIONS	→ CONDITION SEMICOLON INSTRUCTIONS
	→ EXPRESSION SEMICOLON INSTRUCTIONS
	→ FUNCT-CALL SEMICOLON INSTRUCTIONS
	→ ASSIGNATION SEMICOLON INSTRUCTIONS
	→ CONDITION SEMICOLON
	→ EXPRESSION SEMICOLON
	→ FUNCT-CALL SEMICOLON
	→ ASSIGNATION SEMICOLON
	→ EPSILON
PARAM	→ DOLLAR VARIABLE
	→ DOLLAR VARIABLE PARAM-END
	→ EPSILON
PARAM-END	→ COMA DOLLAR VARIABLE
	→ COMA DOLLAR VARIABLE PARAM-END
	→ EPSILON
USER-FUNCT-CALL	→ AND FUNCT-NAME OPEN-PAR CLOSE-PAR SEMICOLON
	→ AND FUNCT-NAME OPEN-PAR PARAM CLOSE-PAR SEMICOLON
	→ AND FUNCT-NAME PARAM SEMICOLON
	→ AND FUNCT-NAME 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	→ FUNCT-ID FUNCT-NAME OPEN-BRAC INSTRUCTIONS RETURN SEMICOLON CLOSE-BRAC
	→ FUNCT-ID FUNCT-NAME OPEN-PAR CLOSE PAR OPEN-BRAC INSTRUCTIONS RETURN SEM
	→ FUNCT-ID FUNCT-NAME OPEN-PAR PARAM CLOSE-PAR OPEN-BRAC INSTRUCTIONS RETU
RETURN	→ RETURN EXPRESSION
	→ RETURN EXPRESSION-COND
	→ RETURN VARIABLE
	→ EPSILON
FUNCTION-LIST	→ FUNCTION
	→ FUNCTION FUNCTION-LIST 3
	→ EPSILON
PROGRAM	→ PROGRAM FUNCTION-LIST
	→ PROGRAM INSTRUCTIONS
	→ FUNCTION-LIST
	→ INSTRUCTIONS
	→ EPSILON

EXPRESSION (?)	VARIABLE OPERATOR VARIABLE EXPRESSION OPERATOR VARIABLE
EXPRESSION-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)*

(slide 13)