FINAL PROJECT

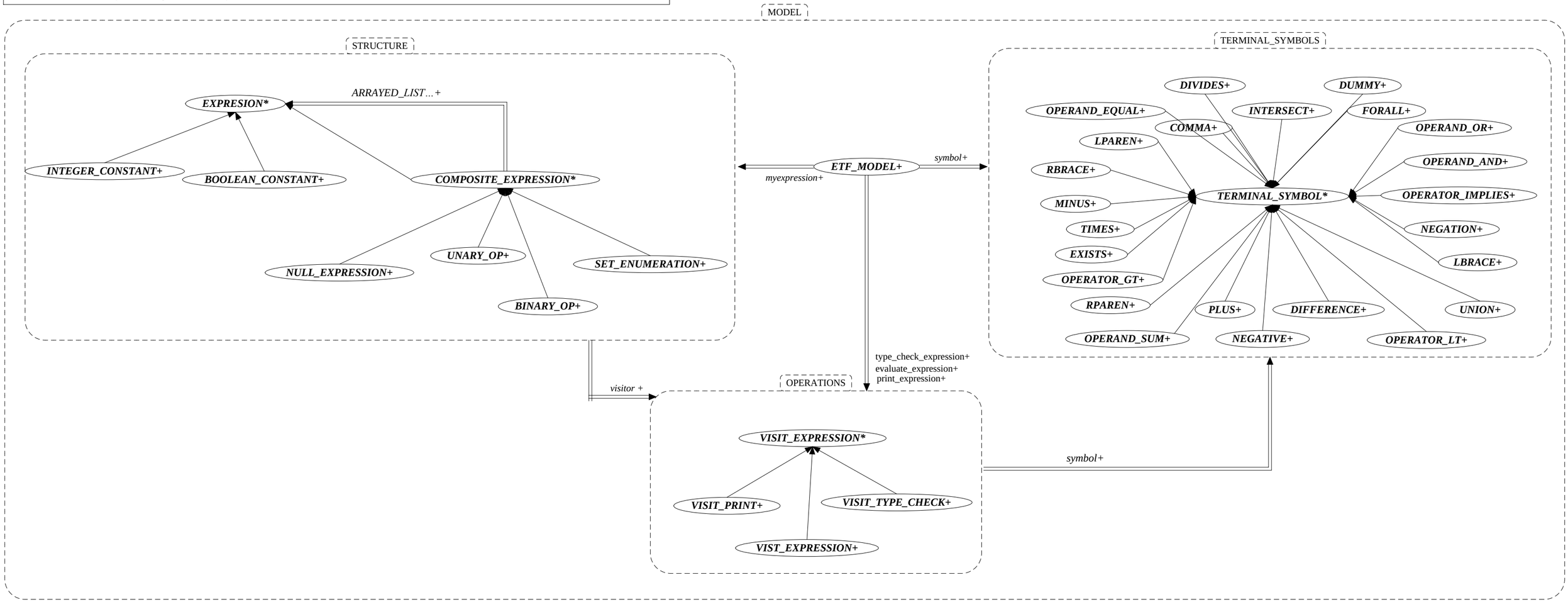
"A TYPE CHECKER FOR SMALL EXPRESSION LANGUAGE"

Names:	CSE Logins
Omar Abid	omarabid
Ashfaq Abdullah	shfb0019
Nidhi Sarathmohan	cse22011

CSE Login Of Submitting Account: *omarabid*

EECS 3311
Summer 2015
Jackie Wang
August 13, 2015

SUPPLIER/CLIENT AND INHERITANCE RELATIONS WITH COMPOSITE PATTERN, VISITOR PATTERN AND ETF MODEL
"A TYPE CHECKER FOR SMALL EXPRESSION LANGUAGE"
SUMMER 2015
EECS 3311
BY: OMAR ABID, ASHFAQ ABDULLAH, NIDHI SARATH



COMPOSITE PATTERN CONNECTED TO ETF MODEL
"A TYPE CHECKER FOR SMALL EXPRESSION LANGUAGE"
SUMMER 2015
EECS 3311

STRUCTURE

ETF_MODEL+

reset
-- Reset model state.
? not is_newly_initialized
! message.is_equal (status_ok)

start_set_enumeration
-- end the given set enumeration
? not expression_fully_specified
! message.is_equal (status_ok)

end_set_enumeration
-- end the given set enumeration
? not expression_fully_specified and is_set_enum_being_specified and not is_set_enum_empty
! message.is_equal (status_ok)

evaluate_expression: VISIT_EVALUATE
type_check_expression: VISIT_TYPE_CHECK
print_expression: VISIT_PRINT

is_new: BOOLEAN
-- is it a new expression
is_new_set: BOOLEAN
-- is it a new set enumeration

my_stack: ARRAYED_LIST
--checks the state of the current expression (not yet fully specified, etc.)

add_addition
-- add binary operation 'addition'
add_conjunction
--add binary operation 'conjunction'
add_boolean_constant (b: BOOLEAN)
? not expression_fully_specified
! message.is_equal (status_ok)
add_integer_constant (i: INTEGER_32)
? not expression_fully_specified
! message.is_equal (status_ok)

evaluate
--evaluate the expression
? no_expression_on_stack and is_type_correct and not is_divisor_zero
pretty_print
--print the expression in readable form
type_check
--check if the expression meets the criteria of type for the operation
? no_expression_on_stack

status_completed_exp: STRING
--error display message
status_divisor_zero: STRING
--error display message
status_enum_must_not_be_empty: STRING
--error display message

message: STRING
report: STRING

out: STRING
--New string containing printable version of current object

NONE

myexpression : EXPRESSION

myexpression +

terminal_symbol +

EXPRESSION*

accept (visit : VISIT_EXPRESSION)*
--accept the expression

ARRAYED_LIST...+

COMPOSITE_EXPRESSION*

add(expression : EXPRESSION)
-- add the given expression to the list
set_first_null : BOOLEAN
--sets the first 'Null_Expression' to current and updates it's string representation
end_set_enumeration:
--ends the current enumeration that is open
add_operation*
--adds the given operation

NONE

expression_list : ARRAYED_LIST[EXPRESSION]

NULL_EXPRESSION +

Make
-- creates a 'nil' symbol
make_first
-- creates a '?' symbol
output : STRING
-- returns the terminal symbol used
accept (visitor : VISIT_EXPRESSION)
-- performs the given operation

INTEGER_CONSTANT +

output : STRING
-- string representation
set_integer_constant(I : INTEGER)
-- set's the 'value' string to the integer constant
accept (visitor: VISIT_EXPRESSION)
--accept the expression

NONE

value : STRING

BOOLEAN_CONSTANT +

output : STRING
-- string representation
set_boolean_constant(I : INTEGER)
-- set's the 'value' string to the boolean constant
accept (visitor: VISIT_EXPRESSION)
--accept the expression

NONE

value : STRING

SET_ENUMERATION +

make
-- creates an empty expression
set_expression_state (I : INTEGER)
-- sets the expression state
? begin_expression <= I and I <= end_expression
add_operation
-- begins, extends or closes enumeration
end_set_enumeration
-- calls correct sub expression to close enumeration
add (expression : EXPRESSION)
-- adds given expression to set enumeration
Accept (visitor: VISITOR_EXPRESSION)
-- performs the given operation

BINARY_OP+

make
-- creates an empty binary operation
left : EXPRESSION
-- returns the left expression
right: EXPRESSION
-- returns the right expression
accept (visitor : VISITOR_EXPRESSION)
--performs the given operation

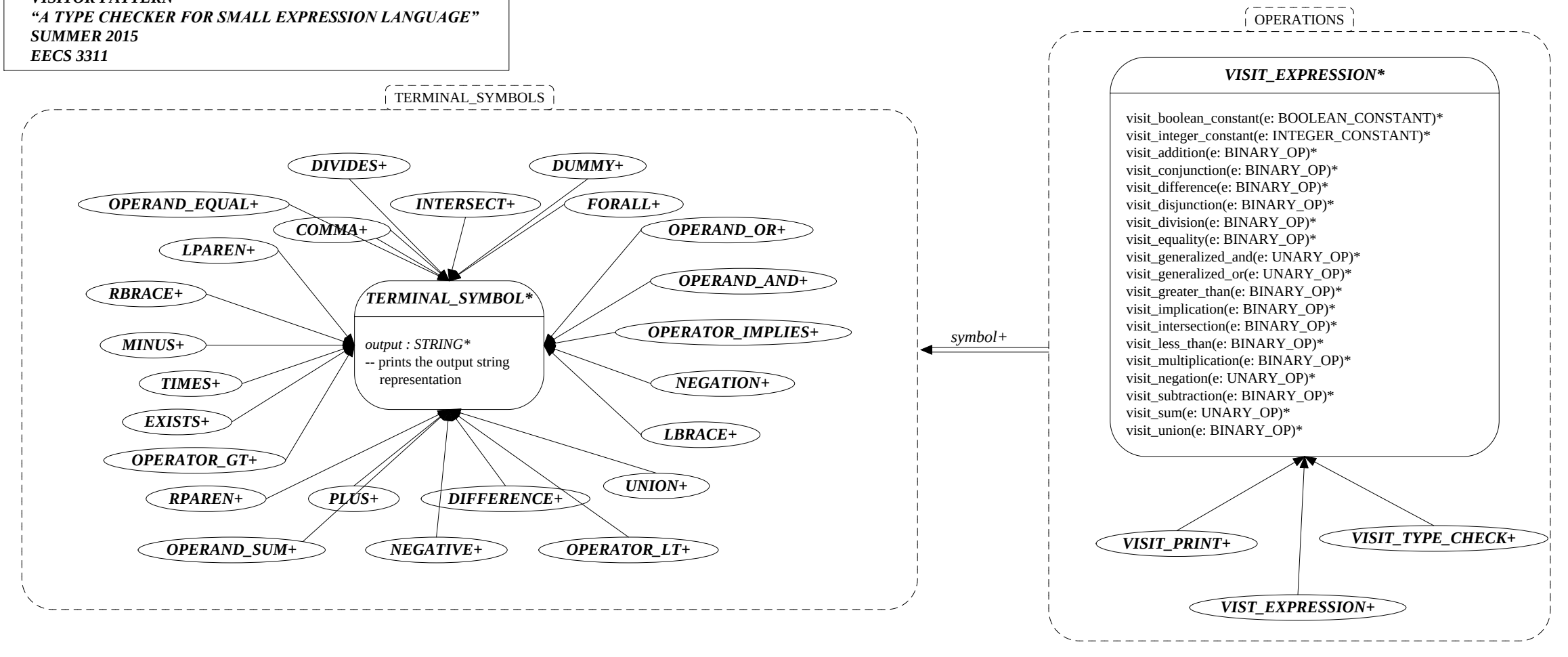
UNARY_OP+

make
-- creates an empty unary operation
child : EXPRESSION
-- returns the child expression
accept (visitor : VISITOR_EXPRESSION)
--performs the given operation

TERMINAL_SYMBOLS

TERMINAL_SYMBOL*

*output : STRING**
-- prints the output representation

DIVIDES+ ***DUMMY+*** ***OPERAND_EQUAL+*** ***INTERSECT+*** ***FORALL+*** ***COMMA+*** ***OPERAND_OR+*** ***LPAREN+*** ***OPERAND_AND+*** ***RBRACE+*** ***OPERATOR_IMPLIES+*** ***MINUS+*** ***NEGATION+*** ***TIMES+*** ***LBRACE+*** ***EXISTS+*** ***OPERATOR_GT+*** ***UNION+*** ***RPAREN+*** ***PLUS+*** ***DIFFERENCE+*** ***OPERAND_SUM+*** ***NEGATIVE+*** ***OPERATOR_LT+***

VISITOR PATTERN
"A TYPE CHECKER FOR SMALL EXPRESSION LANGUAGE"
SUMMER 2015
EECS 3311