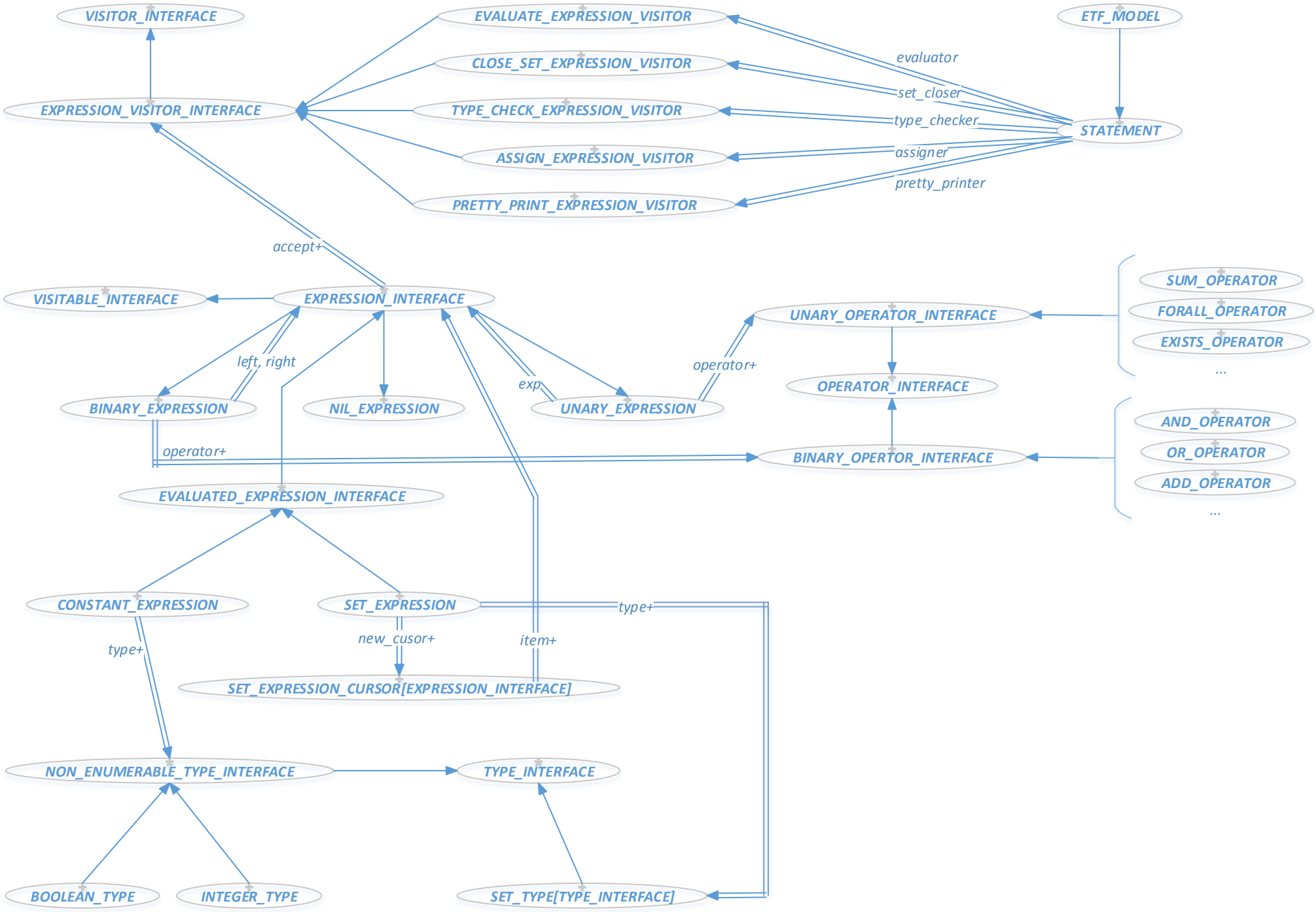
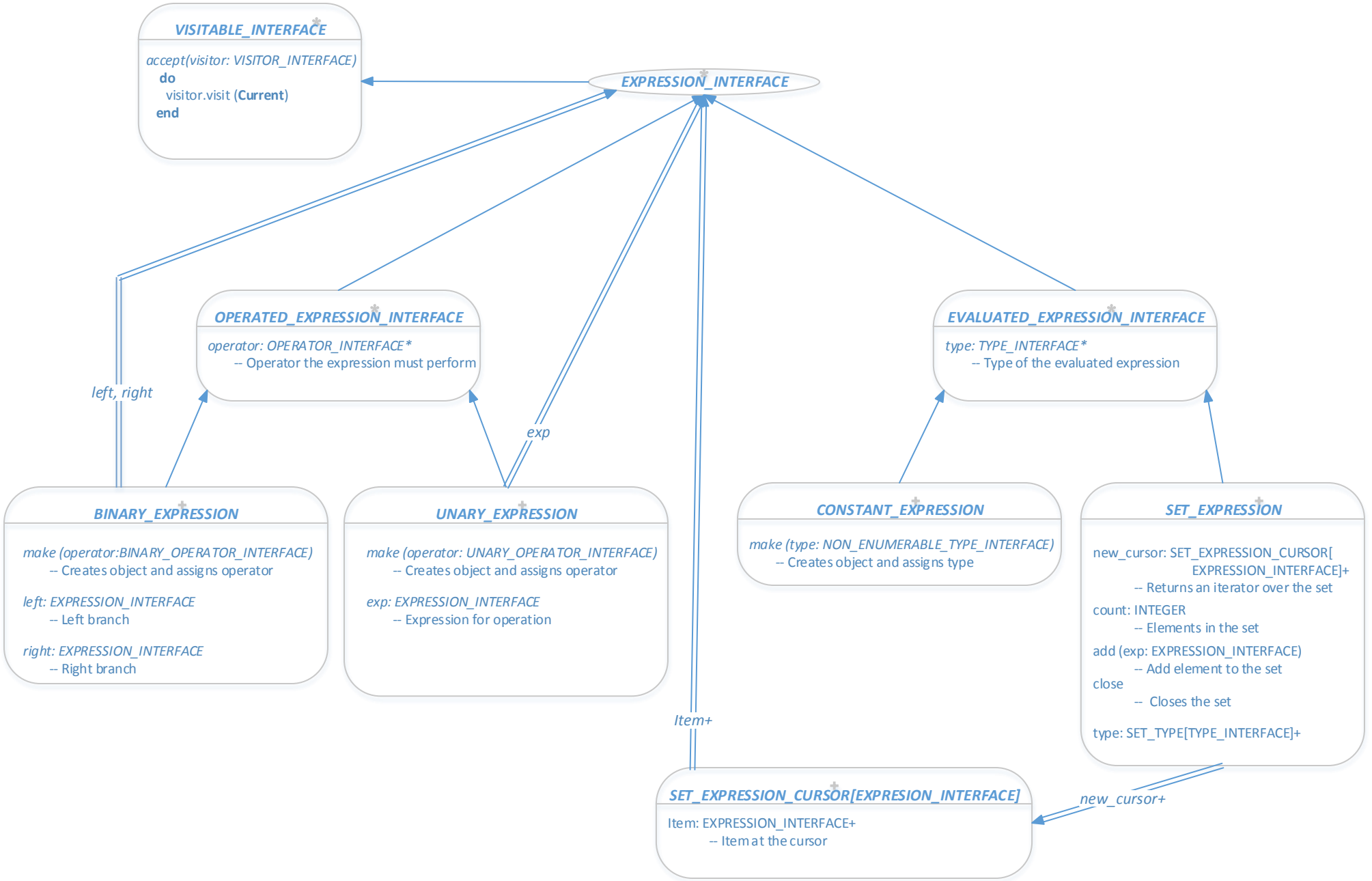
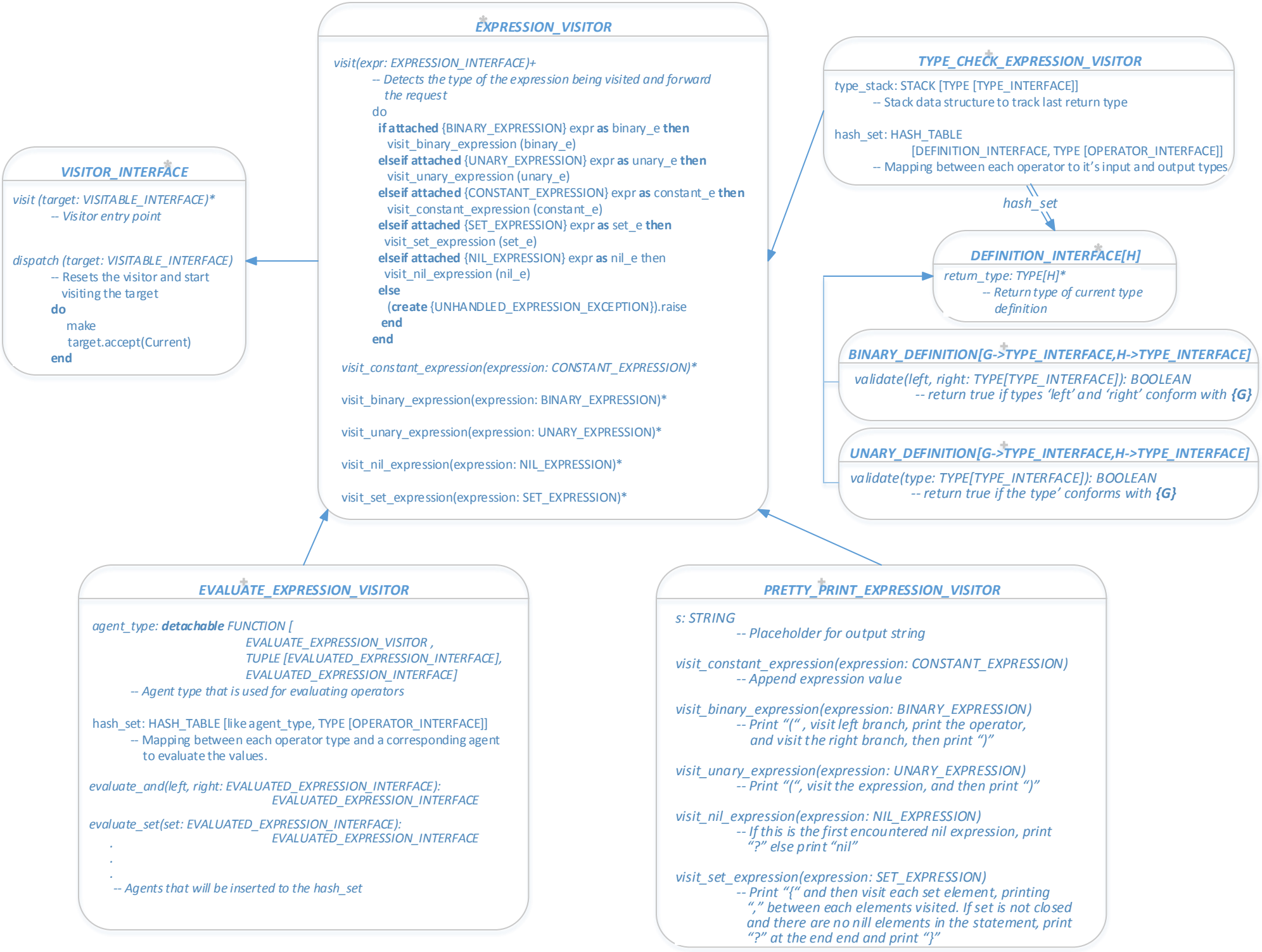
Static Analyzer Project Report

EECS 3311 – Summer 2015

Thalammeherage Raj Perera (cse11011)
Submitted by: cse11011


VISITOR_INTERFACE
EVALUATE_EXPRESSION_VISITOR
ETF_MODEL
CLOSE_SET_EXPRESSION_VISITOR
evaluator
set_closer
EXPRESSION_VISITOR_INTERFACE
TYPE_CHECK_EXPRESSION_VISITOR
type_checker
STATEMENT
ASSIGN_EXPRESSION_VISITOR
assigner
pretty_printer
PRETTY_PRINT_EXPRESSION_VISITOR
accept+
VISITABLE_INTERFACE
EXPRESSION_INTERFACE
UNARY_OPERATOR_INTERFACE
SUM_OPERATOR
FORALL_OPERATOR
EXISTS_OPERATOR
...
left, right
operator+
exp
OPERATOR_INTERFACE
BINARY_EXPRESSION
NIL_EXPRESSION
UNARY_EXPRESSION
AND_OPERATOR
OR_OPERATOR
ADD_OPERATOR
...
operator+
BINARY_OPERTOR_INTERFACE
EVALUATED_EXPRESSION_INTERFACE
CONSTANT_EXPRESSION
SET_EXPRESSION
type+
type+
new_cusor+
item+
SET_EXPRESSION_CURSOR[EXPRESSION_INTERFACE]
NON_ENUMERABLE_TYPE_INTERFACE
TYPE_INTERFACE
BOOLEAN_TYPE
INTEGER_TYPE
SET_TYPE[TYPE_INTERFACE]


VISITABLE_INTERFACE
accept(visitor: VISITOR_INTERFACE)
do
visitor.visit (Current)
end
EXPRESSION_INTERFACE
OPERATED_EXPRESSION_INTERFACE
operator: OPERATOR_INTERFACE*
-- Operator the expression must perform
EVALUATED_EXPRESSION_INTERFACE
type: TYPE_INTERFACE*
-- Type of the evaluated expression
left, right
exp
BINARY_EXPRESSION
make (operator:BINARY_OPERATOR_INTERFACE)
-- Creates object and assigns operator
left: EXPRESSION_INTERFACE
-- Left branch
right: EXPRESSION_INTERFACE
-- Right branch
UNARY_EXPRESSION
make (operator: UNARY_OPERATOR_INTERFACE)
-- Creates object and assigns operator
exp: EXPRESSION_INTERFACE
-- Expression for operation
CONSTANT_EXPRESSION
make (type: NON_ENUMERABLE_TYPE_INTERFACE)
-- Creates object and assigns type
SET_EXPRESSION
new_cursor: SET_EXPRESSION_CURSOR[
EXPRESSION_INTERFACE]+
-- Returns an iterator over the set
count: INTEGER
-- Elements in the set
add (exp: EXPRESSION_INTERFACE)
-- Add element to the set
close
-- Closes the set
type: SET_TYPE[TYPE_INTERFACE]+
Item+
SET_EXPRESSION_CURSOR[EXPRESION_INTERFACE]
Item: EXPRESSION_INTERFACE+
-- Item at the cursor
new_cursor+


EXPRESSION_VISITOR
visit(expr: EXPRESSION_INTERFACE)+
-- Detects the type of the expression being visited and forward the request
do
if attached {BINARY_EXPRESSION} expr as binary_e then
visit_binary_expression (binary_e)
elseif attached {UNARY_EXPRESSION} expr as unary_e then
visit_unary_expression (unary_e)
elseif attached {CONSTANT_EXPRESSION} expr as constant_e then
visit_constant_expression (constant_e)
elseif attached {SET_EXPRESSION} expr as set_e then
visit_set_expression (set_e)
elseif attached {NIL_EXPRESSION} expr as nil_e then
visit_nil_expression (nil_e)
else
(create {UNHANDLED_EXPRESSION_EXCEPTION}).raise
end
end
visit_constant_expression(expression: CONSTANT_EXPRESSION)*
visit_binary_expression(expression: BINARY_EXPRESSION)*
visit_unary_expression(expression: UNARY_EXPRESSION)*
visit_nil_expression(expression: NIL_EXPRESSION)*
visit_set_expression(expression: SET_EXPRESSION)*
VISITOR_INTERFACE
visit (target: VISITABLE_INTERFACE)*
-- Visitor entry point
dispatch (target: VISITABLE_INTERFACE)
-- Resets the visitor and start visiting the target
do
make
target.accept(Current)
end
TYPE_CHECK_EXPRESSION_VISITOR
type_stack: STACK [TYPE [TYPE_INTERFACE]]
-- Stack data structure to track last return type
hash_set: HASH_TABLE
[DEFINITION_INTERFACE, TYPE [OPERATOR_INTERFACE]]
-- Mapping between each operator to it's input and output types
hash_set
DEFINITION_INTERFACE[H]
return_type: TYPE[H]*
-- Return type of current type definition
BINARY_DEFINITION[G->TYPE_INTERFACE,H->TYPE_INTERFACE]
validate(left, right: TYPE[TYPE_INTERFACE]): BOOLEAN
-- return true if types 'left' and 'right' conform with {G}
UNARY_DEFINITION[G->TYPE_INTERFACE,H->TYPE_INTERFACE]
validate(type: TYPE[TYPE_INTERFACE]): BOOLEAN
-- return true if the type' conforms with {G}
EVALUATE_EXPRESSION_VISITOR
agent_type: detachable FUNCTION [
EVALUATE_EXPRESSION_VISITOR ,
TUPLE [EVALUATED_EXPRESSION_INTERFACE],
EVALUATED_EXPRESSION_INTERFACE]
-- Agent type that is used for evaluating operators
hash_set: HASH_TABLE [like agent_type, TYPE [OPERATOR_INTERFACE]]
-- Mapping between each operator type and a corresponding agent to evaluate the values.
evaluate_and(left, right: EVALUATED_EXPRESSION_INTERFACE):
EVALUATED_EXPRESSION_INTERFACE
evaluate_set(set: EVALUATED_EXPRESSION_INTERFACE):
EVALUATED_EXPRESSION_INTERFACE
.
.
.
-- Agents that will be inserted to the hash_set
PRETTY_PRINT_EXPRESSION_VISITOR
s: STRING
-- Placeholder for output string
visit_constant_expression(expression: CONSTANT_EXPRESSION)
-- Append expression value
visit_binary_expression(expression: BINARY_EXPRESSION)
-- Print "(" , visit left branch, print the operator, and visit the right branch, then print ")"
visit_unary_expression(expression: UNARY_EXPRESSION)
-- Print "(", visit the expression, and then print ")"
visit_nil_expression(expression: NIL_EXPRESSION)
-- If this is the first encountered nil expression, print "?" else print "nil"
visit_set_expression(expression: SET_EXPRESSION)
-- Print "{" and then visit each set element, printing "," between each elements visited. If set is not closed and there are no nill elements in the statement, print "?" at the end end and print "}"