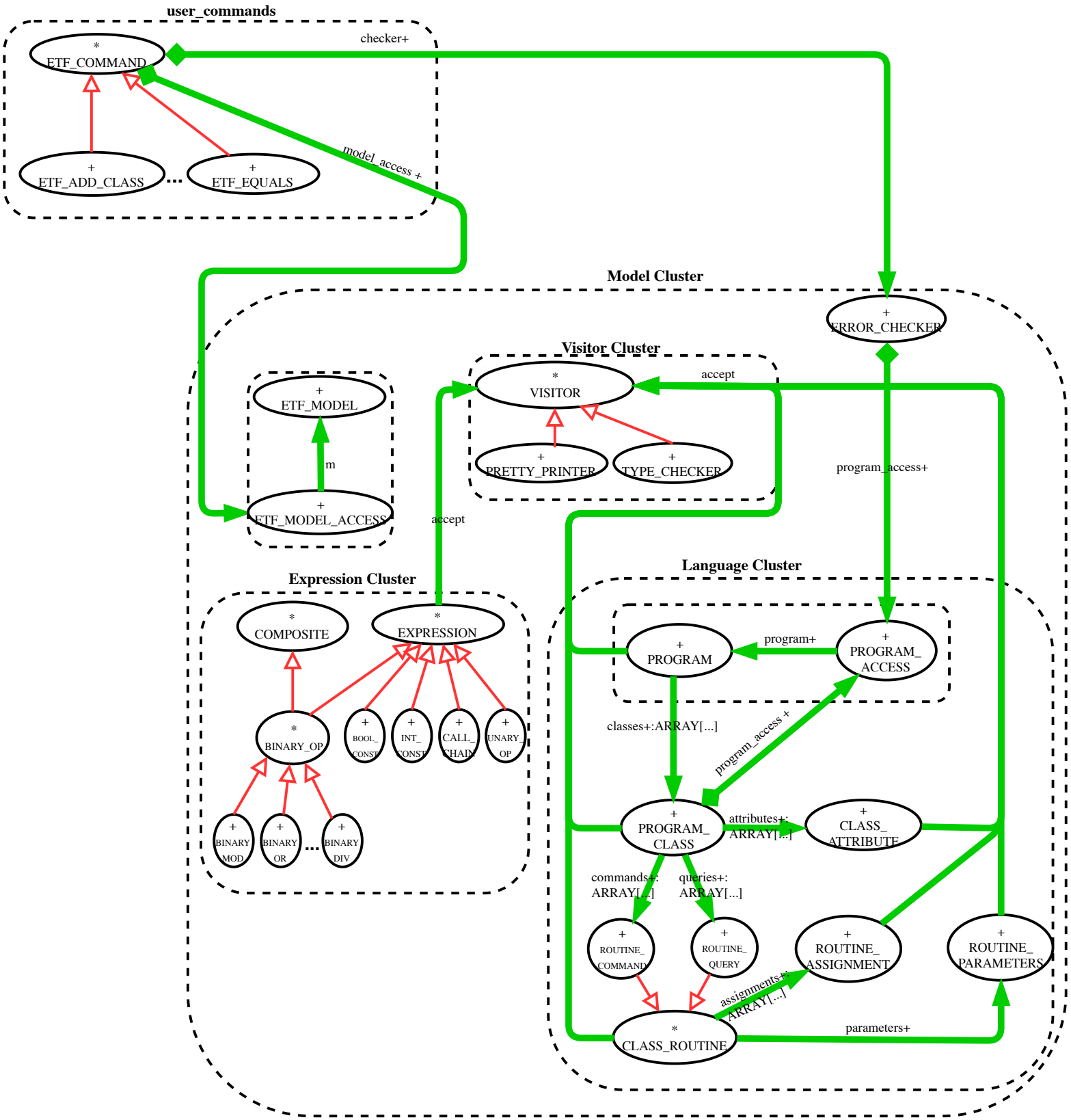
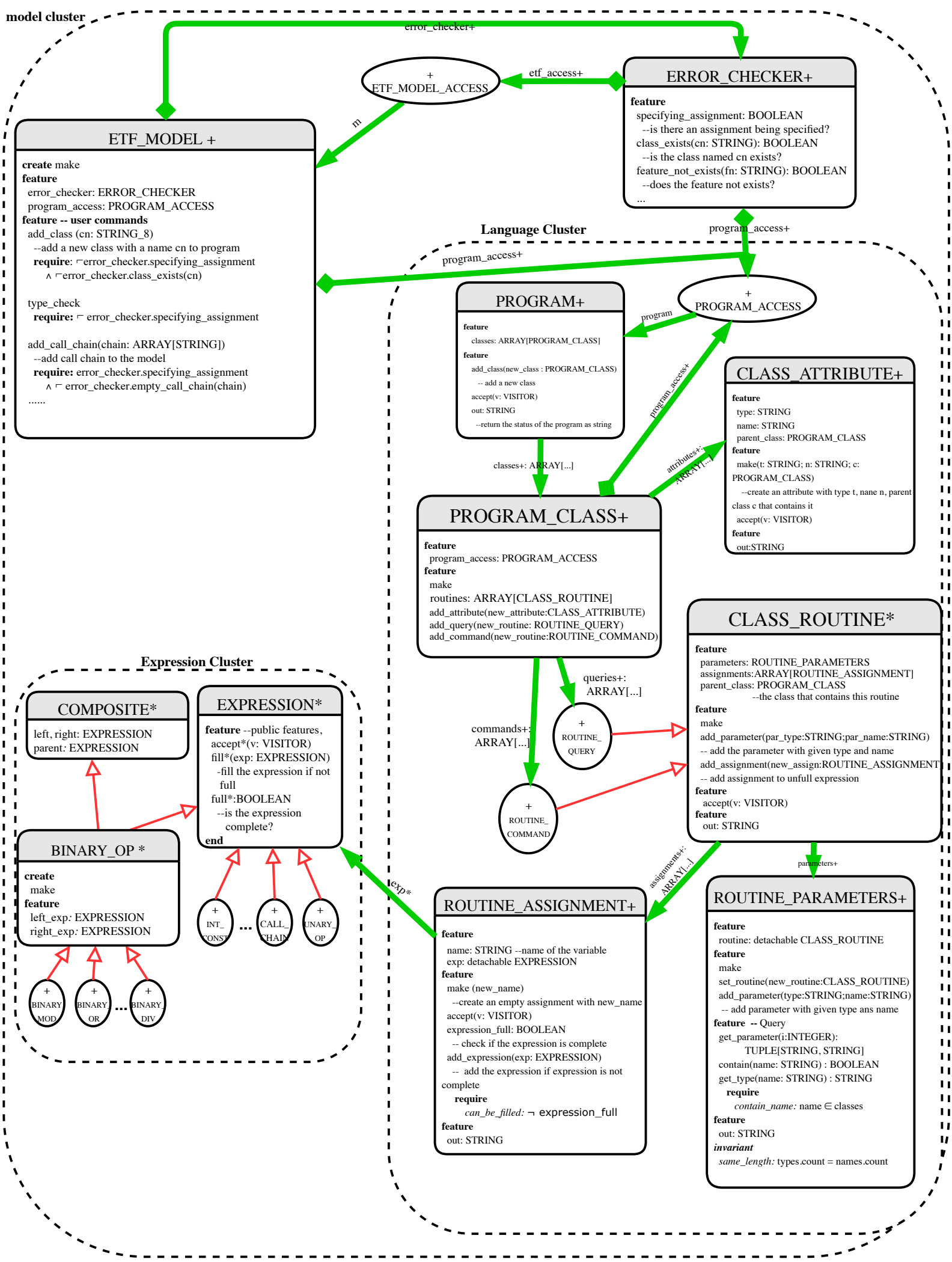
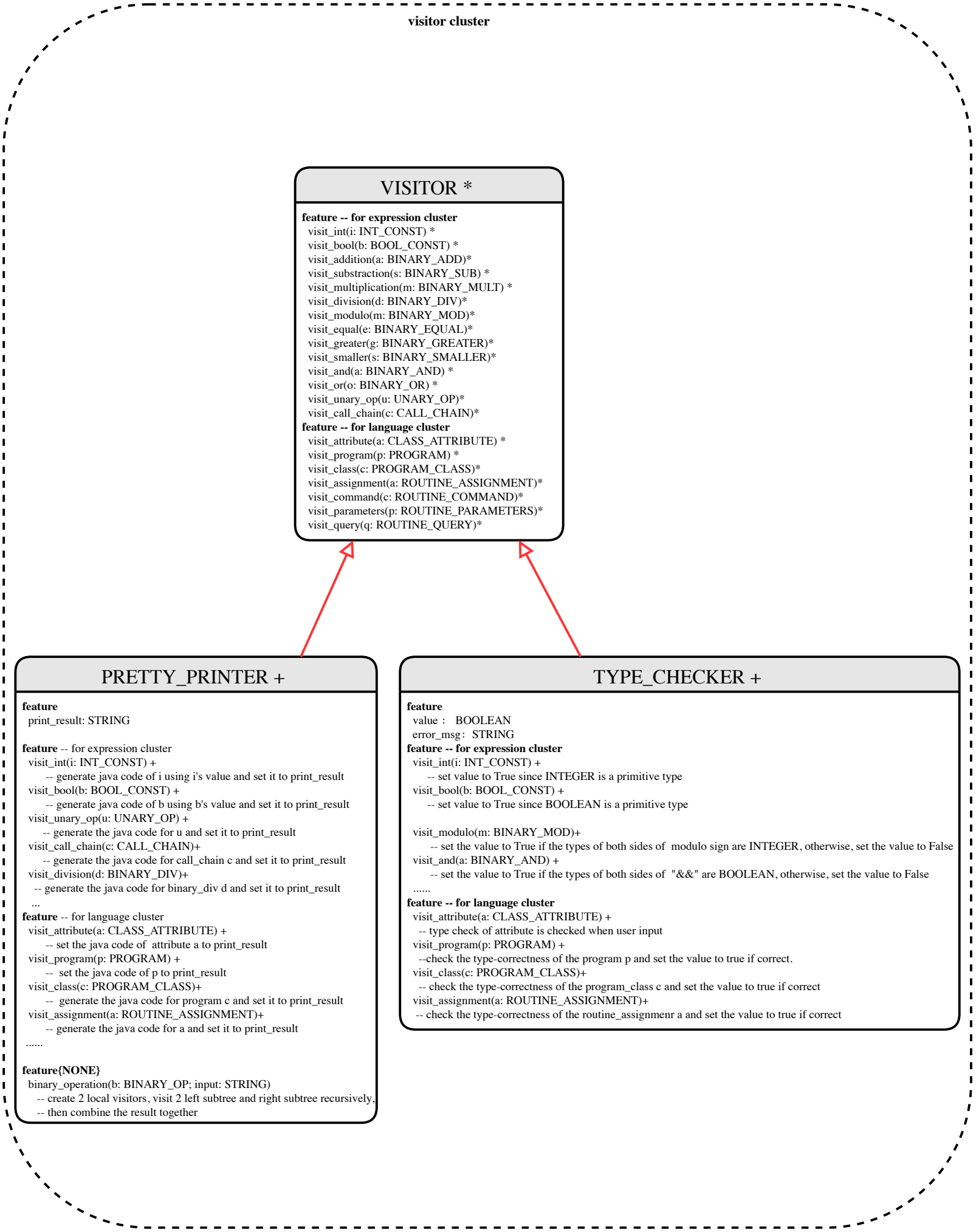

user_commands
checker+
* ETF_COMMAND
+ ETF_ADD_CLASS
+ ETF_EQUALS
model_access +
Model Cluster
+ ERROR_CHECKER
Visitor Cluster
* VISITOR
accept
+ PRETTY_PRINTER
+ TYPE_CHECKER
program_access+
+ ETF_MODEL
m
+ ETF_MODEL_ACCESS
accept
Expression Cluster
* COMPOSITE
* EXPRESSION
* BINARY_OP
+ BOOL_CONST
+ INT_CONST
+ CALL_CHAIN
+ UNARY_OP
+ BINARY_MOD
+ BINARY_OR
+ BINARY_DIV
Language Cluster
+ PROGRAM
program+
+ PROGRAM_ACCESS
classes+:ARRAY[...]
program_access +
+ PROGRAM_CLASS
attributes+: ARRAY[...]
+ CLASS_ATTRIBUTE
commands+: ARRAY[...]
queries+: ARRAY[...]
+ ROUTINE_COMMAND
+ ROUTINE_QUERY
+ ROUTINE_ASSIGNMENT
+ ROUTINE_PARAMETERS
assignments+: ARRAY[...]
* CLASS_ROUTINE
parameters+


model cluster
error_checker+
+
ETF_MODEL_ACCESS
etf_access+
ERROR_CHECKER+
feature
specifying_assignment: BOOLEAN
--is there an assignment being specified?
class_exists(cn: STRING): BOOLEAN
--is the class named cn exists?
feature_not_exists(fn: STRING): BOOLEAN
--does the feature not exists?
...
m
ETF_MODEL +
create make
feature
error_checker: ERROR_CHECKER
program_access: PROGRAM_ACCESS
feature -- user commands
add_class (cn: STRING_8)
--add a new class with a name cn to program
require: ⌜error_checker.specifying_assignment
∧ ⌜error_checker.class_exists(cn)
type_check
require: ⌜ error_checker.specifying_assignment
add_call_chain(chain: ARRAY[STRING])
--add call chain to the model
require: error_checker.specifying_assignment
∧ ⌜ error_checker.empty_call_chain(chain)
.....
Language Cluster
program_access+
program_access+
PROGRAM+
feature
classes: ARRAY[PROGRAM_CLASS]
feature
add_class(new_class : PROGRAM_CLASS)
-- add a new class
accept(v: VISITOR)
out: STRING
--return the status of the program as string
program
+
PROGRAM_ACCESS
program_access+
CLASS_ATTRIBUTE+
feature
type: STRING
name: STRING
parent_class: PROGRAM_CLASS
feature
make(t: STRING; n: STRING; c: PROGRAM_CLASS)
--create an attribute with type t, nane n, parent class c that contains it
accept(v: VISITOR)
feature
out:STRING
classes+: ARRAY[...]
attributes+: ARRAY[...]
PROGRAM_CLASS+
feature
program_access: PROGRAM_ACCESS
feature
make
routines: ARRAY[CLASS_ROUTINE]
add_attribute(new_attribute:CLASS_ATTRIBUTE)
add_query(new_routine: ROUTINE_QUERY)
add_command(new_routine:ROUTINE_COMMAND)
queries+: ARRAY[...]
commands+: ARRAY[...]
+
ROUTINE_ QUERY
+
ROUTINE_ COMMAND
CLASS_ROUTINE*
feature
parameters: ROUTINE_PARAMETERS
assignments:ARRAY[ROUTINE_ASSIGNMENT]
parent_class: PROGRAM_CLASS
--the class that contains this routine
feature
make
add_parameter(par_type:STRING;par_name:STRING)
-- add the parameter with given type and name
add_assignment(new_assign:ROUTINE_ASSIGNMENT)
-- add assignment to unfull expression
feature
accept(v: VISITOR)
feature
out: STRING
parameters+
assignments+: ARRAY[...]
Expression Cluster
COMPOSITE*
left, right: EXPRESSION
parent: EXPRESSION
EXPRESSION*
feature --public features,
accept*(v: VISITOR)
fill*(exp: EXPRESSION)
-fill the expression if not full
full*:BOOLEAN
--is the expression complete?
end
BINARY_OP *
create
make
feature
left_exp: EXPRESSION
right_exp: EXPRESSION
+
INT_ CONST
+
CALL_ CHAIN
+
BINARY_ OP
+
BINARY_ MOD
+
BINARY_ OR
+
BINARY_ DIV
exp*
ROUTINE_ASSIGNMENT+
feature
name: STRING --name of the variable
exp: detachable EXPRESSION
feature
make (new_name)
--create an empty assignment with new_name
accept(v: VISITOR)
expression_full: BOOLEAN
-- check if the expression is complete
add_expression(exp: EXPRESSION)
-- add the expression if expression is not complete
require
can_be_filled: ¬ expression_full
feature
out: STRING
ROUTINE_PARAMETERS+
feature
routine: detachable CLASS_ROUTINE
feature
make
set_routine(new_routine:CLASS_ROUTINE)
add_parameter(type:STRING;name:STRING)
-- add parameter with given type ans name
feature -- Query
get_parameter(i:INTEGER): TUPLE[STRING, STRING]
contain(name: STRING) : BOOLEAN
get_type(name: STRING) : STRING
require
contain_name: name ∈ classes
feature
out: STRING
invariant
same_length: types.count = names.count


visitor cluster
VISITOR *
feature -- for expression cluster
visit_int(i: INT_CONST) *
visit_bool(b: BOOL_CONST) *
visit_addition(a: BINARY_ADD)*
visit_substraction(s: BINARY_SUB) *
visit_multiplication(m: BINARY_MULT) *
visit_division(d: BINARY_DIV)*
visit_modulo(m: BINARY_MOD)*
visit_equal(e: BINARY_EQUAL)*
visit_greater(g: BINARY_GREATER)*
visit_smaller(s: BINARY_SMALLER)*
visit_and(a: BINARY_AND) *
visit_or(o: BINARY_OR) *
visit_unary_op(u: UNARY_OP)*
visit_call_chain(c: CALL_CHAIN)*
feature -- for language cluster
visit_attribute(a: CLASS_ATTRIBUTE) *
visit_program(p: PROGRAM) *
visit_class(c: PROGRAM_CLASS)*
visit_assignment(a: ROUTINE_ASSIGNMENT)*
visit_command(c: ROUTINE_COMMAND)*
visit_parameters(p: ROUTINE_PARAMETERS)*
visit_query(q: ROUTINE_QUERY)*
PRETTY_PRINTER +
feature
print_result: STRING
feature -- for expression cluster
visit_int(i: INT_CONST) +
-- generate java code of i using i's value and set it to print_result
visit_bool(b: BOOL_CONST) +
-- generate java code of b using b's value and set it to print_result
visit_unary_op(u: UNARY_OP) +
-- generate the java code for u and set it to print_result
visit_call_chain(c: CALL_CHAIN)+
-- generate the java code for call_chain c and set it to print_result
visit_division(d: BINARY_DIV)+
-- generate the java code for binary_div d and set it to print_result
...
feature -- for language cluster
visit_attribute(a: CLASS_ATTRIBUTE) +
-- set the java code of attribute a to print_result
visit_program(p: PROGRAM) +
-- set the java code of p to print_result
visit_class(c: PROGRAM_CLASS)+
-- generate the java code for program c and set it to print_result
visit_assignment(a: ROUTINE_ASSIGNMENT)+
-- generate the java code for a and set it to print_result
.....
feature{NONE}
binary_operation(b: BINARY_OP; input: STRING)
-- create 2 local visitors, visit 2 left subtree and right subtree recursively,
-- then combine the result together
TYPE_CHECKER +
feature
value : BOOLEAN
error_msg : STRING
feature -- for expression cluster
visit_int(i: INT_CONST) +
-- set value to True since INTEGER is a primitive type
visit_bool(b: BOOL_CONST) +
-- set value to True since BOOLEAN is a primitive type
visit_modulo(m: BINARY_MOD)+
-- set the value to True if the types of both sides of modulo sign are INTEGER, otherwise, set the value to False
visit_and(a: BINARY_AND) +
-- set the value to True if the types of both sides of "&&" are BOOLEAN, otherwise, set the value to False
.....
feature -- for language cluster
visit_attribute(a: CLASS_ATTRIBUTE) +
-- type check of attribute is checked when user input
visit_program(p: PROGRAM) +
--check the type-correctness of the program p and set the value to true if correct.
visit_class(c: PROGRAM_CLASS)+
-- check the type-correctness of the program_class c and set the value to true if correct
visit_assignment(a: ROUTINE_ASSIGNMENT)+
-- check the type-correctness of the routine_assignmenr a and set the value to true if correct