EECS 3311 Project Report

Course: EECS3311

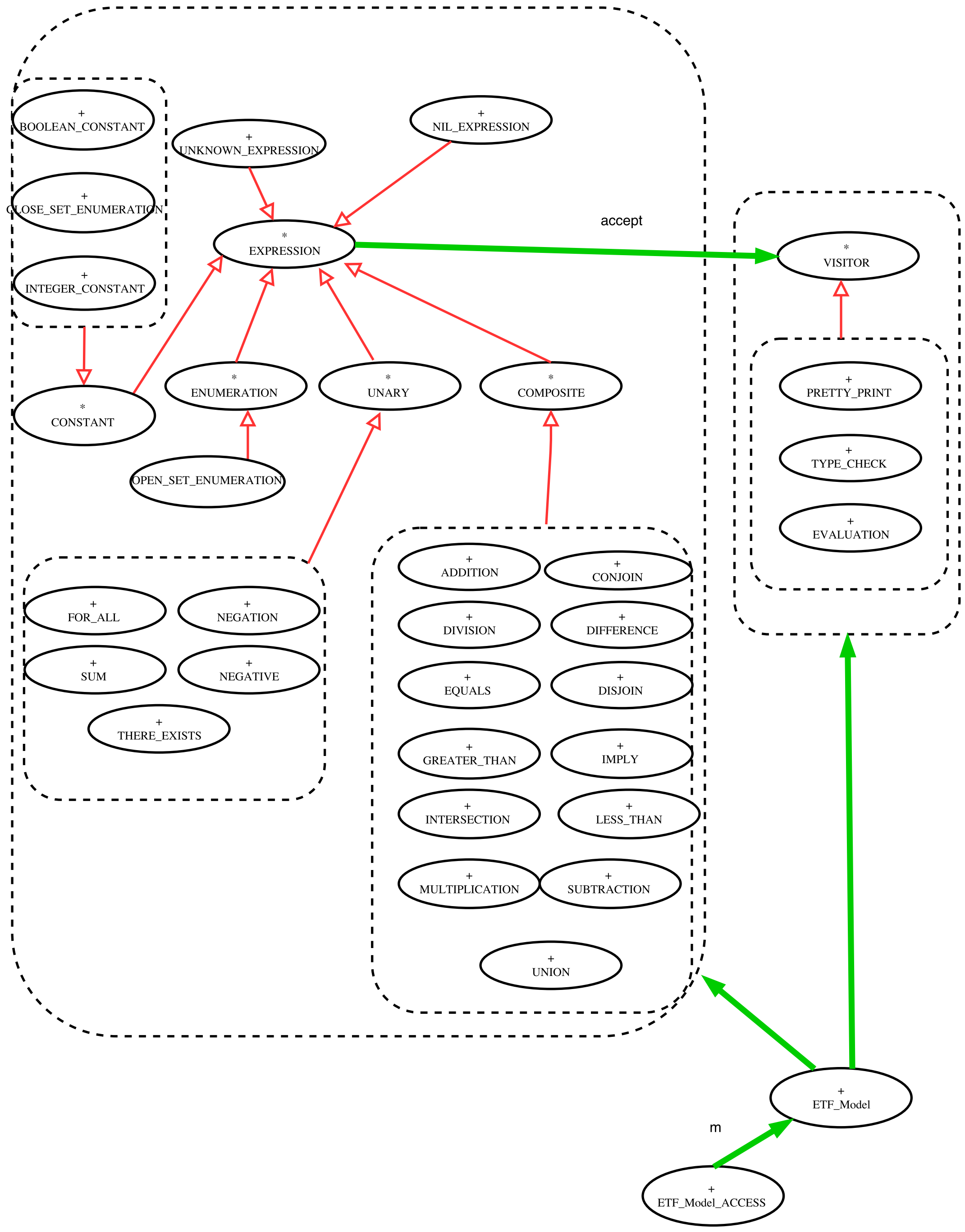
Semester: Fall 2017

Student Name: Yichun Dai (kris0302)
Shaohua Liu (sliu15)

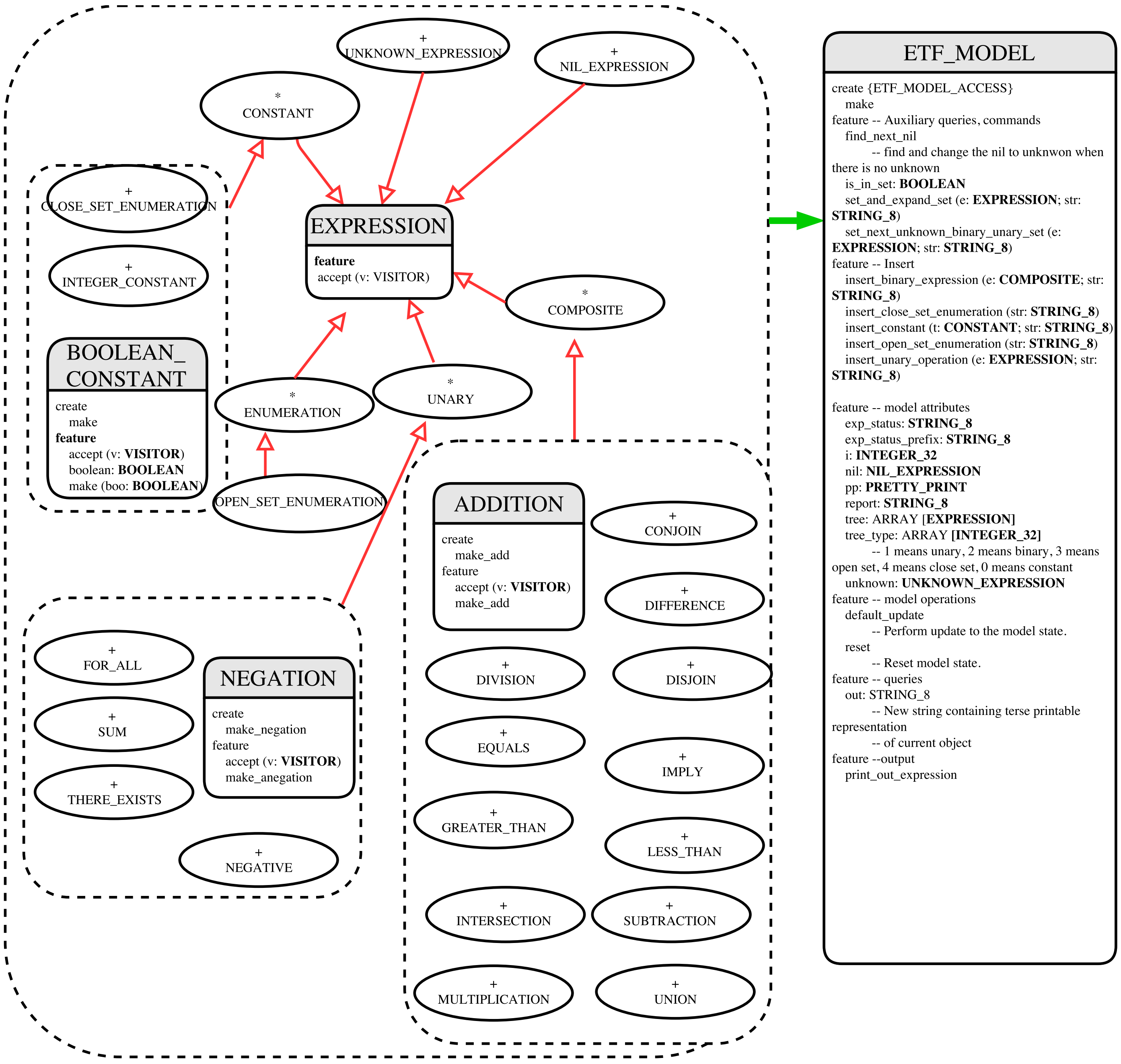
CSE login of the submitting account:

kris0302

Relations Between All Relevant Classes (Diagram1)



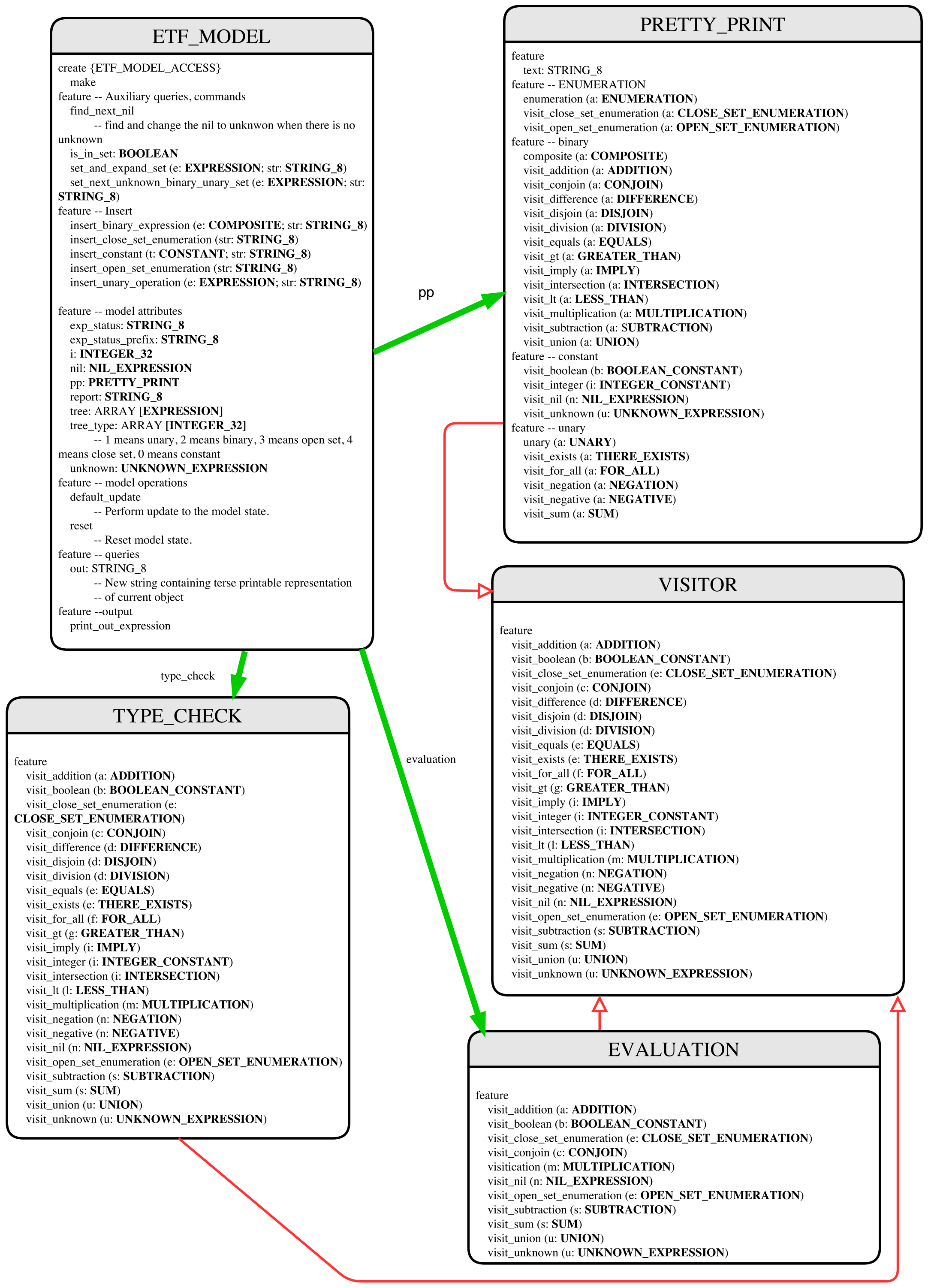
Expression Language Structure (Diagram2)



```
create {ETF_MODEL_ACCESS}
   make
feature -- Auxiliary queries, commands
   find_next_nil
         -- find and change the nil to unknwon when
there is no unknown
   is_in_set: BOOLEAN
   set_and_expand_set (e: EXPRESSION; str:
STRING_8)
   set_next_unknown_binary_unary_set (e:
EXPRESSION; str: STRING_8)
feature -- Insert
   insert_binary_expression (e: COMPOSITE; str:
STRING_8)
   insert_close_set_enumeration (str: STRING_8)
   insert_constant (t: CONSTANT; str: STRING_8)
   insert_open_set_enumeration (str: STRING_8)
   insert_unary_operation (e: EXPRESSION; str:
STRING_8)

feature -- model attributes
   exp_status: STRING_8
   exp_status_prefix: STRING_8
   i: INTEGER_32
   nil: NIL_EXPRESSION
   pp: PRETTY_PRINT
   report: STRING_8
   tree: ARRAY [EXPRESSION]
   tree_type: ARRAY [INTEGER_32]
         -- 1 means unary, 2 means binary, 3 means
open set, 4 means close set, 0 means constant
   unknown: UNKNOWN_EXPRESSION
feature -- model operations
   default_update
         -- Perform update to the model state.
   reset
         -- Reset model state.
feature -- queries
   out: STRING_8
         -- New string containing terse printable
representation
         -- of current object
feature --output
   print_out_expression
```

Language Operations (Diagram3)



```
ETF_MODEL

create {ETF_MODEL_ACCESS}
   make
feature -- Auxiliary queries, commands
   find_next_nil
         -- find and change the nil to unknwon when there is no
unknown
   is_in_set: BOOLEAN
   set_and_expand_set (e: EXPRESSION; str: STRING_8)
   set_next_unknown_binary_unary_set (e: EXPRESSION; str:
STRING_8)
feature -- Insert
   insert_binary_expression (e: COMPOSITE; str: STRING_8)
   insert_close_set_enumeration (str: STRING_8)
   insert_constant (t: CONSTANT; str: STRING_8)
   insert_open_set_enumeration (str: STRING_8)
   insert_unary_operation (e: EXPRESSION; str: STRING_8)

feature -- model attributes
   exp_status: STRING_8
   exp_status_prefix: STRING_8
   i: INTEGER_32
   nil: NIL_EXPRESSION
   pp: PRETTY_PRINT
   report: STRING_8
   tree: ARRAY [EXPRESSION]
   tree_type: ARRAY [INTEGER_32]
         -- 1 means unary, 2 means binary, 3 means open set, 4
means close set, 0 means constant
   unknown: UNKNOWN_EXPRESSION
feature -- model operations
   default_update
         -- Perform update to the model state.
   reset
         -- Reset model state.
feature -- queries
   out: STRING_8
         -- New string containing terse printable representation
         -- of current object
feature --output
   print_out_expression
```

```
PRETTY_PRINT

feature
   text: STRING_8
feature -- ENUMERATION
   enumeration (a: ENUMERATION)
   visit_close_set_enumeration (a: CLOSE_SET_ENUMERATION)
   visit_open_set_enumeration (a: OPEN_SET_ENUMERATION)
feature -- binary
   composite (a: COMPOSITE)
   visit_addition (a: ADDITION)
   visit_conjoin (a: CONJOIN)
   visit_difference (a: DIFFERENCE)
   visit_disjoin (a: DISJOIN)
   visit_division (a: DIVISION)
   visit_equals (a: EQUALS)
   visit_gt (a: GREATER_THAN)
   visit_imply (a: IMPLY)
   visit_intersection (a: INTERSECTION)
   visit_lt (a: LESS_THAN)
   visit_multiplication (a: MULTIPLICATION)
   visit_subtraction (a: SUBTRACTION)
   visit_union (a: UNION)
feature -- constant
   visit_boolean (b: BOOLEAN_CONSTANT)
   visit_integer (i: INTEGER_CONSTANT)
   visit_nil (n: NIL_EXPRESSION)
   visit_unknown (u: UNKNOWN_EXPRESSION)
feature -- unary
   unary (a: UNARY)
   visit_exists (a: THERE_EXISTS)
   visit_for_all (a: FOR_ALL)
   visit_negation (a: NEGATION)
   visit_negative (a: NEGATIVE)
   visit_sum (a: SUM)
```

```
VISITOR

feature
   visit_addition (a: ADDITION)
   visit_boolean (b: BOOLEAN_CONSTANT)
   visit_close_set_enumeration (e: CLOSE_SET_ENUMERATION)
   visit_conjoin (c: CONJOIN)
   visit_difference (d: DIFFERENCE)
   visit_disjoin (d: DISJOIN)
   visit_division (d: DIVISION)
   visit_equals (e: EQUALS)
   visit_exists (e: THERE_EXISTS)
   visit_for_all (f: FOR_ALL)
   visit_gt (g: GREATER_THAN)
   visit_imply (i: IMPLY)
   visit_integer (i: INTEGER_CONSTANT)
   visit_intersection (i: INTERSECTION)
   visit_lt (l: LESS_THAN)
   visit_multiplication (m: MULTIPLICATION)
   visit_negation (n: NEGATION)
   visit_negative (n: NEGATIVE)
   visit_nil (n: NIL_EXPRESSION)
   visit_open_set_enumeration (e: OPEN_SET_ENUMERATION)
   visit_subtraction (s: SUBTRACTION)
   visit_sum (s: SUM)
   visit_union (u: UNION)
   visit_unknown (u: UNKNOWN_EXPRESSION)
```

```
TYPE_CHECK

feature
   visit_addition (a: ADDITION)
   visit_boolean (b: BOOLEAN_CONSTANT)
   visit_close_set_enumeration (e:
CLOSE_SET_ENUMERATION)
   visit_conjoin (c: CONJOIN)
   visit_difference (d: DIFFERENCE)
   visit_disjoin (d: DISJOIN)
   visit_division (d: DIVISION)
   visit_equals (e: EQUALS)
   visit_exists (e: THERE_EXISTS)
   visit_for_all (f: FOR_ALL)
   visit_gt (g: GREATER_THAN)
   visit_imply (i: IMPLY)
   visit_integer (i: INTEGER_CONSTANT)
   visit_intersection (i: INTERSECTION)
   visit_lt (l: LESS_THAN)
   visit_multiplication (m: MULTIPLICATION)
   visit_negation (n: NEGATION)
   visit_negative (n: NEGATIVE)
   visit_nil (n: NIL_EXPRESSION)
   visit_open_set_enumeration (e: OPEN_SET_ENUMERATION)
   visit_subtraction (s: SUBTRACTION)
   visit_sum (s: SUM)
   visit_union (u: UNION)
   visit_unknown (u: UNKNOWN_EXPRESSION)
```

```
EVALUATION

feature
   visit_addition (a: ADDITION)
   visit_boolean (b: BOOLEAN_CONSTANT)
   visit_close_set_enumeration (e: CLOSE_SET_ENUMERATION)
   visit_conjoin (c: CONJOIN)
   visitication (m: MULTIPLICATION)
   visit_nil (n: NIL_EXPRESSION)
   visit_open_set_enumeration (e: OPEN_SET_ENUMERATION)
   visit_subtraction (s: SUBTRACTION)
   visit_sum (s: SUM)
   visit_union (u: UNION)
   visit_unknown (u: UNKNOWN_EXPRESSION)
```

Information Hiding

For information Hiding is to make certain attributes, commands or queries only available for the class, other classes cannot access those attributes and features. For inheritance, the parent classes without feature {NONE} for attributes ,commands or queries, the child class will inherit those features. For the class which defined as feature {NONE} in the attribute section or command, queries section, then only this class can access those features which are not shown in other classes. It is important to implement as information hiding princely in the design, since it will precent other classes change attribute, command and queries.

Single Choice Principle

Single choice principle: If there is change on the common part, we should not modify more then one place. In this project we use inheritance to solve this problem. We define the feature in the parent class and we should make the feature as public, so that the child class can inherit this feature. If there is a change, we only make changes on parent class, so the child class would not change the implementation, it automatically inherits the parent class' updated feature. When there is a feature call of the child class (the feature also defines in parent class) , if that feature is not redefine, we call the feature in the parent class, so the change will take place.

Open-Close Principle

Closed for syntactic construction of the language open for new operations of the language. So in the pretty print, type check and evaluation class inherit visitor class to make corresponding operation. We do not change the construction, we just inherit the feature in Visitor class and redefine it. For example, pretty print, we just redefine the operation feature like visit_addition to print on the screen. For type check we redefine the feature visit_addition to check the type of right and left child. For evaluation we also redefine the feature visit_addition to evaluate the result. Although they have the same feature name, but their functionality are totally different.

Uniform Access Principle

Uniform access principle is to allow the client access to the service is uniform, so the program change, it won't affect the clients' access. So, in the project, we use expand class to make client uniform access to the model section. We don't care too much about how the implementation of the model feature change. The client can use the feature defined in the model class to achieve corresponding operations.