Appendix F:
Example Run

This appendix presents a small, complete example of running an EXPRESS [34] model through the constraint-understanding system. Section F.1 contains the input model, and Section F.2 contains the FIL files which were output from the constraint-understanding system to show the frames in memory.

F.1 The Course World Model

An EXPRESS model has been devised by the Author to test the range of higher-level constraints (HLCs) that the constraint-understanding system is capable of extracting. The results of running this model through the system, in terms of the number of HLCs of each kind that were extracted, are presented in Section 7.2.

The model describes courses in a university, the students who take those courses (and the marks they get), and the staff who teach those courses. The model is in no way intended to exemplify the principles of 'good' information modelling - its purpose is merely to exercise the range of EXPRESS constructs that the system can handle, the range of HLCs that it can extract (see Appendix B), and a good range (though not all) of the cases in Appendix C.

The model itself follows.

```
SCHEMA course_world;

   (*******************************************************)
   (* ENTITY person                                       *)
   (*******************************************************)
   (* Represents a living person, who must be either a man, *)
   (* or a woman, but not both.                           *)
   (*******************************************************)

   ENTITY person ABSTRACT SUPERTYPE OF (ONEOF (man, woman));
      surname : STRING;
      first_name : STRING;
      middle_names : LIST [0 : ?] OF STRING;
      age : INTEGER;
      role : OPTIONAL university_role;
```

```
WHERE
  age_must_be_sensible:
    ((age >= 0) AND (age < 130));
END_ENTITY;

(****************************************)
(* ENTITY man                           *)
(****************************************)
(* Represents a man (a kind of person). *)
(****************************************)

ENTITY man SUBTYPE OF (person);
END_ENTITY;

(******************************************)
(* ENTITY woman                           *)
(******************************************)
(* Represents a woman (a kind of person). *)
(******************************************)

ENTITY woman SUBTYPE OF (person);
END_ENTITY;

(****************************************************************)
(* TYPE university_role                                         *)
(****************************************************************)
(* Represents the role that a person can play in a university: *)
(* either a student, or a member of staff, but not both.       *)
(****************************************************************)

TYPE university_role = SELECT (student, staff);
END_TYPE;

(*************************************************************)
(* ENTITY student                                            *)
(*************************************************************)
(* Represents a person’s role as a student in a university. *)
(*************************************************************)

ENTITY student;
  courses_taken : SET [0 : ?] OF student_course_record;
DERIVE
  average_mark : REAL
    := sum_marks(courses_taken) / SIZEOF(courses_taken);
INVERSE
  the_person : person FOR role;
END_ENTITY;
```

```
(*****************************************************************)
(* ENTITY student_course_record                                  *)
(*****************************************************************)
(* Represents the association between a particular student and a *)
(* particular course that the student takes.                     *)
(*****************************************************************)

ENTITY student_course_record;
  course : course;
  exam_mark : REAL;
  lab_mark : REAL;
DERIVE
  total_mark : REAL
    := exam_mark + lab_mark;
INVERSE
  the_student : student FOR courses_taken;
END_ENTITY;

(**********************************************************)
(* FUNCTION sum_marks                                     *)
(**********************************************************)
(* Function to compute the sum of marks of a given set of *)
(* student course records.                                *)
(**********************************************************)

FUNCTION sum_marks (records : SET OF student_course_record) : REAL;
  LOCAL
    i : INTEGER;
    sum : REAL := 0.0;
  END_LOCAL;

  REPEAT i:= 1 TO SIZEOF (records);
    sum := sum + records[i].total_mark;
  END_REPEAT;

  RETURN(sum);
END_FUNCTION;

(********************************************************************)
(* ENTITY staff                                                     *)
(********************************************************************)
(* Represents a person's role as a member of staff in a university. *)
(********************************************************************)

ENTITY staff;
  courses_taught : OPTIONAL SET [1 : ?] OF course;
INVERSE
  the_person : person FOR role;
END_ENTITY;
```

```
(**********************************************************)
(* ENTITY course                                          *)
(**********************************************************)
(* Represents a course module in a university.            *)
(* Course codes are of the form `DDlpn' or `DDlpns', where:    *)
(* - `DD' is the department;                              *)
(* - `l' is the level (year);                             *)
(* - `p' is the programme (stream);                       *)
(* - `n' is used to give each module a different `lpn' number; *)
(* - `s' is the semester that the module runs in.         *)
(**********************************************************)

  ENTITY course;
    department : STRING(2) FIXED;
    level : INTEGER(1);
    programme : INTEGER(1);
    id_number : INTEGER(1);
    semester : OPTIONAL INTEGER(1);
    title : STRING;
  INVERSE
    the_lecturer : staff FOR courses_taught;
    the_students : SET [0 : ?] OF student_course_record FOR course;
  UNIQUE
    department, level, programme, id_number;
  END_ENTITY;
END_SCHEMA;
```

F.2 The Output

Running the Course World model in Section F.1 through the constraint-understanding system results in a collection of FIL files, holding the frames that represent the augmented model (i.e. the frame representation of the actual model, and the HLCs extracted from it).

The frames are stored in separate FIL files for each frame class. HTML versions of these files (c.f. Appendix E, Section E.5) are available for browsing on the Author's web site (Broad [8]).

Since *all* frames in the system are written out in the FIL files, only those frames which are part of the augmented model are shown here. The frames that are not shown here represent the frame class models of Appendix A, and the cases in Appendix C.

F.2.1 `adder` Frames

```
INSTANCE adder#3;
        extracted_from = {equation#7};
        addends = {explicit_attribute#131, explicit_attribute#132};
        sum = derived_attribute#3;
END_INSTANCE;
```

F.2.2 `aggregation_type` Frames

```
INSTANCE aggregation_type#28;
        kind = set_type;
        element_type = entity_type#49;
        lower_bound = literal_expression#44;
        upper_bound = built_in_constant#26;
        is_unique = true;
        is_optional = false;
END_INSTANCE;

INSTANCE aggregation_type#29;
        kind = list_type;
        element_type = simple_type(string_type);
        lower_bound = literal_expression#45;
        upper_bound = built_in_constant#27;
        is_unique = false;
        is_optional = false;
END_INSTANCE;

INSTANCE aggregation_type#30;
        kind = set_type;
        element_type = entity_type#50;
        lower_bound = literal_expression#46;
        upper_bound = built_in_constant#28;
        is_unique = true;
        is_optional = false;
END_INSTANCE;

INSTANCE aggregation_type#31;
        kind = set_type;
        element_type = entity_type#49;
        lower_bound = literal_expression#49;
        upper_bound = built_in_constant#29;
        is_unique = true;
        is_optional = false;
END_INSTANCE;
```

```
INSTANCE aggregation_type#32;
        kind = set_type;
        element_type = entity_type#50;
        lower_bound = literal_expression#50;
        upper_bound = built_in_constant#30;
        is_unique = true;
        is_optional = false;
END_INSTANCE;

INSTANCE aggregation_type#33;
        kind = set_type;
        element_type = entity_type#50;
        lower_bound = literal_expression#51;
        upper_bound = built_in_constant#31;
        is_unique = true;
        is_optional = false;
END_INSTANCE;
```

F.2.3 at_most_one Frames

```
INSTANCE at_most_one#4;
        extracted_from = {defined_type#27};
        conjunct_of = conjunction#4;
        choice_set = {entity_type#46, entity_type#48};
        constrainee = explicit_attribute#122;
END_INSTANCE;
```

F.2.4 binary_operation Frames

```
INSTANCE binary_operation#23;
        operator = divide;
        first_expression = user_function_call#1;
        second_expression = built_in_function_call#2;
END_INSTANCE;

INSTANCE binary_operation#24;
        operator = greater_than_or_equals;
        first_expression = explicit_attribute#119;
        second_expression = literal_expression#47;
END_INSTANCE;

INSTANCE binary_operation#25;
        operator = less_than;
        first_expression = explicit_attribute#119;
        second_expression = literal_expression#48;
END_INSTANCE;
```

```
INSTANCE binary_operation#26;
        operator = and_operator;
        first_expression = binary_operation#24;
        second_expression = binary_operation#25;
END_INSTANCE;

INSTANCE binary_operation#27;
        operator = add;
        first_expression = explicit_attribute#131;
        second_expression = explicit_attribute#132;
END_INSTANCE;
```

F.2.5 bound Frames

```
INSTANCE bound#19;
        extracted_from = {binary_operation#24};
        conjunct_of = conjunction#5;
        kind = lower;
        inclusive = true;
        discrete = true;
        variable = explicit_attribute#119;
        limit = 0;
END_INSTANCE;

INSTANCE bound#20;
        extracted_from = {binary_operation#25};
        conjunct_of = conjunction#5;
        kind = upper;
        inclusive = false;
        discrete = true;
        variable = explicit_attribute#119;
        limit = 130;
END_INSTANCE;
```

F.2.6 built_in_constant Frames

```
INSTANCE built_in_constant#26;
        built_in_value = built_in(infinity);
END_INSTANCE;

INSTANCE built_in_constant#27;
        built_in_value = built_in(infinity);
END_INSTANCE;

INSTANCE built_in_constant#28;
        built_in_value = built_in(infinity);
END_INSTANCE;
```

```
INSTANCE built_in_constant#29;
        built_in_value = built_in(infinity);
END_INSTANCE;

INSTANCE built_in_constant#30;
        built_in_value = built_in(infinity);
END_INSTANCE;

INSTANCE built_in_constant#31;
        built_in_value = built_in(infinity);
END_INSTANCE;
```

F.2.7 `built_in_function_call` Frames

```
INSTANCE built_in_function_call#2;
        actual_parameters = [explicit_attribute#120];
        function_called = sizeof_function;
END_INSTANCE;
```

F.2.8 `conjunction` Frames

```
INSTANCE conjunction#3;
        extracted_from = {entity_type#50};
        conjuncts = {non_null#3, non_null#4, non_null#5, equation#7};
                    (* set indirectly *)
END_INSTANCE;

INSTANCE conjunction#4;
        extracted_from = {entity_type#47};
        conjuncts = {exactly_one#4, non_null#6, non_null#7, non_null#8,
                     zero_or_more#3, non_null#9, at_most_one#4,
                     conjunction#5}; (* set indirectly *)
END_INSTANCE;

INSTANCE conjunction#5;
        extracted_from = {binary_operation#26};
        conjunct_of = conjunction#4;
        conjuncts = {bound#19, bound#20, range#2}; (* set indirectly *)
END_INSTANCE;

INSTANCE conjunction#6;
        extracted_from = {entity_type#49};
        conjuncts = {non_null#10, non_null#11, non_null#12, non_null#13,
                     non_null#14}; (* set indirectly *)
END_INSTANCE;
```

```
INSTANCE conjunction#7;
        extracted_from = {entity_type#45};
END_INSTANCE;

INSTANCE conjunction#8;
        extracted_from = {entity_type#44};
END_INSTANCE;

INSTANCE conjunction#9;
        extracted_from = {entity_type#46};
        conjuncts = {non_null#15, zero_or_more#4, equation#8};
                    (* set indirectly *)
END_INSTANCE;

INSTANCE conjunction#10;
        extracted_from = {entity_type#48};
        conjuncts = {zero_or_more#5}; (* set indirectly *)
END_INSTANCE;
```

F.2.9 **defined_type** Frames

```
INSTANCE defined_type#27;
        name = "university_role";
        tipe = select_type({entity_type#46, entity_type#48});
END_INSTANCE;
```

F.2.10 **derived_attribute** Frames

```
INSTANCE derived_attribute#2;
        name = "average_mark";
        tipe = simple_type(real_type);
        formula = binary_operation#23;
        the_entity = entity_type#46; (* set indirectly *)
END_INSTANCE;

INSTANCE derived_attribute#3;
        name = "total_mark";
        tipe = simple_type(real_type);
        formula = binary_operation#27;
        the_entity = entity_type#50; (* set indirectly *)
END_INSTANCE;
```

F.2.11 `domain_rule` Frames

```
INSTANCE domain_rule#1;
        label = "age_must_be_sensible";
        expression = binary_operation#26;
        the_type = entity_type#47; (* set indirectly *)
END_INSTANCE;
```

F.2.12 `entity_type` Frames

```
INSTANCE entity_type#44;
        name = "man";
        is_abstract = false;
        subtype_of = {entity_type#47};
        explicit_attributes = {};
        higher_level_constraints = conjunction#8;
END_INSTANCE;

INSTANCE entity_type#45;
        name = "woman";
        is_abstract = false;
        subtype_of = {entity_type#47};
        explicit_attributes = {};
        higher_level_constraints = conjunction#7;
END_INSTANCE;

INSTANCE entity_type#46;
        name = "student";
        is_abstract = false;
        explicit_attributes = {explicit_attribute#120};
        derived_attributes = {derived_attribute#2};
        inverse_attributes = {inverse_attribute#9};
        higher_level_constraints = conjunction#9;
END_INSTANCE;

INSTANCE entity_type#47;
        name = "person";
        domain_rules = {domain_rule#1};
        is_abstract = true;
        supertype_of = oneof_expression#8;
        explicit_attributes = {explicit_attribute#116,
                               explicit_attribute#117,
                               explicit_attribute#118,
                               explicit_attribute#119,
                               explicit_attribute#122};
        higher_level_constraints = conjunction#4;
END_INSTANCE;
```

```
INSTANCE entity_type#48;
        name = "staff";
        is_abstract = false;
        explicit_attributes = {explicit_attribute#115};
        inverse_attributes = {inverse_attribute#10};
        higher_level_constraints = conjunction#10;
END_INSTANCE;

INSTANCE entity_type#49;
        name = "course";
        is_abstract = false;
        explicit_attributes = {explicit_attribute#109,
                               explicit_attribute#110,
                               explicit_attribute#111,
                               explicit_attribute#112,
                               explicit_attribute#113,
                               explicit_attribute#114};
       inverse_attributes = {inverse_attribute#11, inverse_attribute#12};
        unique_rules = {unique_rule#1};
        higher_level_constraints = conjunction#6;
END_INSTANCE;

INSTANCE entity_type#50;
        name = "student_course_record";
        is_abstract = false;
        explicit_attributes = {explicit_attribute#130,
                               explicit_attribute#131,
                               explicit_attribute#132};
        derived_attributes = {derived_attribute#3};
        inverse_attributes = {inverse_attribute#13};
        higher_level_constraints = conjunction#3;
END_INSTANCE;
```

F.2.13 equation Frames

```
INSTANCE equation#7;
        extracted_from = {derived_attribute#3};
        conjunct_of = conjunction#3;
        equals = {derived_attribute#3, binary_operation#27};
END_INSTANCE;

INSTANCE equation#8;
        extracted_from = {derived_attribute#2};
        conjunct_of = conjunction#9;
        equals = {derived_attribute#2, binary_operation#23};
END_INSTANCE;
```

F.2.14 `exactly_one` Frames

```
INSTANCE exactly_one#4;
        extracted_from = {oneof_expression#8};
        conjunct_of = conjunction#4;
        choice_set = {entity_type#44, entity_type#45};
        constrainee = entity_type#47;
END_INSTANCE;
```

F.2.15 `explicit_attribute` Frames

```
INSTANCE explicit_attribute#109;
        name = "department";
        tipe = simple_type(string_type);
        is_optional = false;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#110;
        name = "level";
        tipe = simple_type(integer_type);
        is_optional = false;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#111;
        name = "programme";
        tipe = simple_type(integer_type);
        is_optional = false;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#112;
        name = "id_number";
        tipe = simple_type(integer_type);
        is_optional = false;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#113;
        name = "semester";
        tipe = simple_type(integer_type);
        is_optional = true;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;
```

```
INSTANCE explicit_attribute#114;
        name = "title";
        tipe = simple_type(string_type);
        is_optional = false;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#115;
        name = "courses_taught";
        tipe = aggregation_type#28;
        is_optional = true;
        the_entity = entity_type#48; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#116;
        name = "surname";
        tipe = simple_type(string_type);
        is_optional = false;
        the_entity = entity_type#47; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#117;
        name = "first_name";
        tipe = simple_type(string_type);
        is_optional = false;
        the_entity = entity_type#47; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#118;
        name = "middle_names";
        tipe = aggregation_type#29;
        is_optional = false;
        the_entity = entity_type#47; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#119;
        name = "age";
        tipe = simple_type(integer_type);
        is_optional = false;
        the_entity = entity_type#47; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#120;
        name = "courses_taken";
        tipe = aggregation_type#30;
        is_optional = false;
        the_entity = entity_type#46; (* set indirectly *)
END_INSTANCE;
```

```
INSTANCE explicit_attribute#121;
        name = "role";
        tipe = defined_type#27;
        is_optional = true;
END_INSTANCE;

INSTANCE explicit_attribute#122;
        name = "role";
        tipe = defined_type#27;
        is_optional = true;
        the_entity = entity_type#47; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#123;
        name = "role";
        tipe = defined_type#27;
        is_optional = true;
END_INSTANCE;

INSTANCE explicit_attribute#124;
        name = "courses_taught";
        tipe = aggregation_type#31;
        is_optional = true;
END_INSTANCE;

INSTANCE explicit_attribute#125;
        name = "course";
        tipe = entity_type#49;
        is_optional = false;
END_INSTANCE;

INSTANCE explicit_attribute#126;
        name = "department";
        tipe = simple_type(string_type);
        is_optional = false;
END_INSTANCE;

INSTANCE explicit_attribute#127;
        name = "level";
        tipe = simple_type(integer_type);
        is_optional = false;
END_INSTANCE;

INSTANCE explicit_attribute#128;
        name = "programme";
        tipe = simple_type(integer_type);
        is_optional = false;
END_INSTANCE;
```

```
INSTANCE explicit_attribute#129;
        name = "id_number";
        tipe = simple_type(integer_type);
        is_optional = false;
END_INSTANCE;

INSTANCE explicit_attribute#130;
        name = "course";
        tipe = entity_type#49;
        is_optional = false;
        the_entity = entity_type#50; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#131;
        name = "exam_mark";
        tipe = simple_type(real_type);
        is_optional = false;
        the_entity = entity_type#50; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#132;
        name = "lab_mark";
        tipe = simple_type(real_type);
        is_optional = false;
        the_entity = entity_type#50; (* set indirectly *)
END_INSTANCE;

INSTANCE explicit_attribute#133;
        name = "courses_taken";
        tipe = aggregation_type#33;
        is_optional = false;
END_INSTANCE;
```

F.2.16 `inverse_attribute` Frames

```
INSTANCE inverse_attribute#9;
        name = "the_person";
        tipe = entity_type#47;
        for_attribute = explicit_attribute#121;
        the_entity = entity_type#46; (* set indirectly *)
END_INSTANCE;

INSTANCE inverse_attribute#10;
        name = "the_person";
        tipe = entity_type#47;
        for_attribute = explicit_attribute#123;
        the_entity = entity_type#48; (* set indirectly *)
END_INSTANCE;
```

```
INSTANCE inverse_attribute#11;
        name = "the_lecturer";
        tipe = entity_type#48;
        for_attribute = explicit_attribute#124;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE inverse_attribute#12;
        name = "the_students";
        tipe = aggregation_type#32;
        for_attribute = explicit_attribute#125;
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;

INSTANCE inverse_attribute#13;
        name = "the_student";
        tipe = entity_type#46;
        for_attribute = explicit_attribute#133;
        the_entity = entity_type#50; (* set indirectly *)
END_INSTANCE;
```

F.2.17 `literal_expression` Frames

```
INSTANCE literal_expression#44;
        literal_value = integer_literal(1);
END_INSTANCE;

INSTANCE literal_expression#45;
        literal_value = integer_literal(0);
END_INSTANCE;

INSTANCE literal_expression#46;
        literal_value = integer_literal(0);
END_INSTANCE;

INSTANCE literal_expression#47;
        literal_value = integer_literal(0);
END_INSTANCE;

INSTANCE literal_expression#48;
        literal_value = integer_literal(130);
END_INSTANCE;

INSTANCE literal_expression#49;
        literal_value = integer_literal(1);
END_INSTANCE;
```

```
INSTANCE literal_expression#50;
        literal_value = integer_literal(0);
END_INSTANCE;

INSTANCE literal_expression#51;
        literal_value = integer_literal(0);
END_INSTANCE;
```

F.2.18 multiplier Frames

```
INSTANCE multiplier#3;
        extracted_from = {equation#8};
        multiplicands = {derived_attribute#2, built_in_function_call#2};
        product = user_function_call#1;
END_INSTANCE;
```

F.2.19 non_null Frames

```
INSTANCE non_null#3;
        extracted_from = {explicit_attribute#130};
        conjunct_of = conjunction#3;
        attribute = explicit_attribute#130;
END_INSTANCE;

INSTANCE non_null#4;
        extracted_from = {explicit_attribute#131};
        conjunct_of = conjunction#3;
        attribute = explicit_attribute#131;
END_INSTANCE;

INSTANCE non_null#5;
        extracted_from = {explicit_attribute#132};
        conjunct_of = conjunction#3;
        attribute = explicit_attribute#132;
END_INSTANCE;

INSTANCE non_null#6;
        extracted_from = {explicit_attribute#116};
        conjunct_of = conjunction#4;
        attribute = explicit_attribute#116;
END_INSTANCE;

INSTANCE non_null#7;
        extracted_from = {explicit_attribute#117};
        conjunct_of = conjunction#4;
        attribute = explicit_attribute#117;
END_INSTANCE;
```

```
INSTANCE non_null#8;
        extracted_from = {explicit_attribute#118};
        conjunct_of = conjunction#4;
        attribute = explicit_attribute#118;
END_INSTANCE;

INSTANCE non_null#9;
        extracted_from = {explicit_attribute#119};
        conjunct_of = conjunction#4;
        attribute = explicit_attribute#119;
END_INSTANCE;

INSTANCE non_null#10;
        extracted_from = {explicit_attribute#109};
        conjunct_of = conjunction#6;
        attribute = explicit_attribute#109;
END_INSTANCE;

INSTANCE non_null#11;
        extracted_from = {explicit_attribute#110};
        conjunct_of = conjunction#6;
        attribute = explicit_attribute#110;
END_INSTANCE;

INSTANCE non_null#12;
        extracted_from = {explicit_attribute#111};
        conjunct_of = conjunction#6;
        attribute = explicit_attribute#111;
END_INSTANCE;

INSTANCE non_null#13;
        extracted_from = {explicit_attribute#112};
        conjunct_of = conjunction#6;
        attribute = explicit_attribute#112;
END_INSTANCE;

INSTANCE non_null#14;
        extracted_from = {explicit_attribute#114};
        conjunct_of = conjunction#6;
        attribute = explicit_attribute#114;
END_INSTANCE;

INSTANCE non_null#15;
        extracted_from = {explicit_attribute#120};
        conjunct_of = conjunction#9;
        attribute = explicit_attribute#120;
END_INSTANCE;
```

F.2.20 `oneof_expression` Frames

```
INSTANCE oneof_expression#8;
        expressions = {entity_type#44, entity_type#45};
END_INSTANCE;
```

F.2.21 `range` Frames

```
INSTANCE range#2;
        extracted_from = {bound#19, bound#20};
        conjunct_of = conjunction#5;
        lower_bound = bound#19;
        upper_bound = bound#20;
        variable = explicit_attribute#119;
END_INSTANCE;
```

F.2.22 `unique_rule` Frames

```
INSTANCE unique_rule#1;
        label = "";
        attributes = {explicit_attribute#126,
                      explicit_attribute#127,
                      explicit_attribute#128,
                      explicit_attribute#129};
        the_entity = entity_type#49; (* set indirectly *)
END_INSTANCE;
```

F.2.23 `user_function_call` Frames

```
INSTANCE user_function_call#1;
        actual_parameters = [explicit_attribute#120];
        function_called = "sum_marks";
END_INSTANCE;
```

F.2.24 `zero_or_more` Frames

```
INSTANCE zero_or_more#3;
        extracted_from = {explicit_attribute#118};
        conjunct_of = conjunction#4;
        attribute = explicit_attribute#118;
END_INSTANCE;
```

```
INSTANCE zero_or_more#4;
        extracted_from = {explicit_attribute#120};
        conjunct_of = conjunction#9;
        attribute = explicit_attribute#120;
END_INSTANCE;

INSTANCE zero_or_more#5;
        extracted_from = {explicit_attribute#115};
        conjunct_of = conjunction#10;
        attribute = explicit_attribute#115;
END_INSTANCE;
```

An observant reader may have noticed some imperfections in the frame representation of the EXPRESS model:

- There is no frame representation of the function definition `sum_marks`. The frame representing the call to this function (`user_function_call#1`) simply gives the name of the function as a string. ExJava (Appendix E, Section E.3) does not represent the bodies of functions either (although it goes a little further than the EXPRESS frame class model (Appendix A, Section A.1) and holds the parameters and return type of each user-defined function in a `Function` object).
- Some of the `explicit_attribute` frames have been redundantly duplicated, namely those referred to in the `INVERSE` and `UNIQUE` clauses. This duplication appears to originate from ExJava, rather than the frame system has introduced it.

These flaws are minor irritants rather than serious problems, as they do not affect the HLCs that the system extracts from this or any other model. However, they will have to be fixed before any cases which handle uniqueness rules, inverse attributes or function definitions can be introduced!