INTERNATIONAL STANDARD

ISO 13584-20

First edition
1998-07-01

Industrial automation systems and integration — Parts library —

Part 20:
Logical resource: Logical model of expressions

Systèmes d'automatisation industrielle et intégration — Bibliothèque de composants —

Partie 20: Ressource logique: Modèle logique d'expressions

Reference number
ISO 13584-20:1998(E)

Contents

Page

International Organization for Standardization
Case postale 56 • CH-1211 Genève 20 • Switzerland
Internet iso@iso.ch

Printed in Switzerland

Figures

Table

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organisations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardisation.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

International Standard ISO 13584-20 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Subcommittee SC 4, *Industrial data.*

ISO 13584 consists of the following parts, under the general title *Industrial automation systems and integration — Parts library*:

— *Part 1: Overview and fundamental principles*

— *Part 10: Conceptual description: Conceptual model of parts library*

— *Part 20: Logical resource: Logical model of expressions*

— *Part 24: Logical resource: Logical model of supplier library*

— *Part 26: Logical resource: Supplier identification*

— *Part 31: Implementation resource: Geometric programming interface*

— *Part 42: Description methodology: Methodology for structuring part families*

— *Part 101: View exchange protocol: Geometric view exchange protocol by parametric program*

— *Part 102: View exchange protocol: View exchange protocol by ISO 10303 conforming specification*

The structure of ISO 13584 is described in ISO 13584-1. The numbering of the parts of ISO 13584 reflects its structure:

— Parts 10 to 19 specify the conceptual descriptions,

— Parts 20 to 29 specify the logical resources,

— Parts 30 to 39 specify the implementation resources,

— Parts 40 to 49 specify the description methodology,

— Parts 50 to 59 specify the conformance testing,

— Parts 100 to 199 specify the view exchange protocol,

— Parts 500 to 599 specify the standardised content.

Should further parts of ISO 13584 be published, they will follow the same numbering pattern.

Annexes A and B form an integral part of this part of ISO 13584. Annexes C, D, E and F are for information only.

Introduction

ISO 13584 is an International Standard for the computer-interpretable representation and exchange of part library data. The objective is to provide a neutral mechanism capable of transferring parts library data, independent of any application that is using a parts library data system. The nature of this description makes it suitable not only for the exchange of files containing parts, but also as a basis for implementing and sharing databases of parts library data.

ISO 13584 is organised as a series of parts, each published separately. The parts of ISO 13854 fall into one of the following series: conceptual descriptions, logical resources, implementation resources, description methodology, conformance testing, view exchange protocol, and standardised content. The series are described in ISO 13584-1. This part of ISO 13584 is a member of the logical resources series.

This part of ISO 13584 provides the general purpose EXPRESS resource constructs needed for expression modelling. These EXPRESS resource constructs are intended to be detailed in other parts of ISO 13584. They are also intended to be used outside ISO 13584 wherever EXPRESS information models of expressions prove to be useful.

Industrial automation systems and integration — Parts library — Part 20: Logical resource: Logical model of expressions

1 Scope

This part of ISO 13584 specifies:

— an EXPRESS schema for generic expressions;

— an EXPRESS schema for expressions, that models the subset of the allowed expressions in the EXPRESS language defined in ISO 10303-11 that corresponds to integer, real, Boolean and string data types. This schema uses the resources defined in the generic expression schema.

The following are within the scope of this part of ISO 13584:

— the exchange of expressions that involve both constants and variables;

— the function that checks whether or not a numeric expression should evaluate to an integer value;

— the constraints which ensure that an expression is semantically correct;

— the computation of the variables or functions used in an expression;

— the function that checks if an expression may be mapped on to the SQL query language.

The following are outside the scope of this part of ISO 13584:

— the assignment of values to variables within some context;

— the triggering mechanism that computes the value of an expression in a given context.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 13584. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 13584 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 9075: 1992, *Information technology — Database languages — SQL.*

ISO 10303-11: 1994, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

ISO 10303-44: 1994, *Industrial automation systems and integration — Product data representation and exchange — Part 44: Integrated generic resources: Product structure configuration.*

3 Terms and definitions

For the purposes of this part of ISO 13584, the following terms and definitions apply.

3.1 Terms and definitions from ISO 10303-11

For the purposes of this part of ISO 13584, the following terms and definitions given in ISO 10303-11:1994 (which are repeated below for convenience) apply.

3.1.1

data type
a domain of values.

[ISO 10303-11:1994]

3.1.2

entity
a class of information defined by common properties.

[ISO 10303-11:1994]

3.1.3

entity data type
a representation of an entity. An entity data type establishes a domain of values defined by common attributes and constraints.

[ISO 10303-11:1994]

3.1.4

entity (data type) instance
a named unit of data which represents a unit of information within the class defined by an entity. It is a member of the domain established by an entity data type.

[ISO 10303-11:1994]

3.2 Terms and definitions from ISO 10303-44

For the purpose of this part of ISO 13584, the following terms and definitions given in ISO 10303-44:1994 (which are repeated below for convenience) apply.

3.2.1

ancestor node
any node that can be reached from a given node, by successive traversals of links in the reverse direction. For a given node, its ancestor nodes include all parent nodes, all parent nodes of these parent nodes, etc.

[ISO 10303-44:1994]

3.2.2

child node
node to which a link is pointing.

[ISO 10303-44:1994]

3.2.3

descendent node
any node that can be reached from a given node, by successive traversals of links. For a given node, its descendent nodes include all children nodes, all children nodes of these children nodes, etc.

[ISO 10303-44:1994]

3.2.4

directed acyclic graph
collection of nodes and directed links such that no node is an ancestor (or descendant) of itself.

[ISO 10303-44:1994]

3.2.5

link
uni-directional relationship from one node to another node within a directed acyclic graph.

[ISO 10303-44:1994]

3.2.6

node
element of a directed acyclic graph, connected to other such elements by links.

[ISO 10303-44:1994]

3.2.7

parent node
node from which a link is initiated.

[ISO 10303-44:1994]

3.3 Other terms and definitions

For the purpose of this part of ISO 13584, the following apply.

3.3.1

arity of an operator
the maximum number of operands that shall be associated with an operator.

3.3.2

binary operator
an operator whose arity is equal to two.

3.3.3

environment
association between syntax and semantics.

NOTE In the context of this part of ISO 13584, the **environment** entity associates to a **generic_variable** (syntax) its corresponding meaning (semantics) represented by the **variable_semantics** entity.

3.3.4

evaluation
the computation of the value represented by an expression.

3.3.5

expression
set of variables or constants or both that are combined by operators.

NOTE 1 An expression specifies a function whose arguments are the variables occurring in the expression.

NOTE 2 The underlying structure is a directed acyclic graph, where nodes are operators, constants or variables, and where links represent the uni-directional relationship from each operators to its arguments, that are themselves expressions.

3.3.6

expression data type
domain where the result of an expression shall take its values.

NOTE In this part of ISO 13584, type control and type synthesis are only performed for the **ISO13584_expressions_schema.** Type control is ensured by the structure and the rules of this EXPRESS schema. Type synthesis is done by the EXPRESS TYPEOF function that computes whether an **expression** is a **numeric_expression**, a **Boolean_expression** or a **string_expression**, and by the **is_int** function that computes whether or not a **numeric_expression** evaluates to an INTERGER value.

3.3.7

interpretation
function that returns the semantics associated to each variable.

NOTE This function uses an environment that associates a variable to its corresponding semantics (and may be to its possible value).

3.3.8

multiple arity operator
an operator whose arity is greater than two.

3.3.9

operator
function that combines one or several values, named its operands, to produce a value, named its result.

NOTE The definition of an operator includes the data type definition of its operands and of its result.

3.3.10

semantics
meaning of a given concept.

4

EXAMPLE The semantics of a variable is the meaning carried by this variable.

NOTE In the context of this part of ISO 13584, the semantics is represented by the **variable_semantics** entity. This entity is an ABSTRACT SUPERTYPE, that can be specialised to carry specific meanings and values.

3.3.11

syntactic representation
sequence of characters that represents a given concept.

NOTE 1 In usual programming languages, the sequence of characters used to represent the different concepts shall obey to a set of rules known as the syntax of the language.

NOTE 2 In the context of this part of ISO 13584, a syntactic representation is an entity (data type) instance name.

3.3.12

type control
operation that allows the determination whether or not a given expression is correctly typed.

NOTE 1 An expression is correctly typed if the data type of each operand of each operator of this expression complies with the required data type.

NOTE 2 In this part of ISO 13584, type control is ensured by the constraints of the EXPRESS schema.

3.3.13

type synthesis
determination of an expression data type.

NOTE In the **ISO13584_expressions_schema** schema, the data type of each constant, variable or operator result is carried by the entity that represents it; therefore, the data type of an **expression** may be deduced from the result of the EXPRESS TYPEOF function applied to the node that represents this **expression**.

3.3.14

unary operator
an operator whose arity is equal to one.

3.3.15

variable
representation of a value that shall belong to a specified data type.

4 Abbreviated terms

For the purpose of this part of ISO 13584 the following abbreviated term applies.

— SQL : The structured Query Language defined by ISO/IEC 9075:1992.

5 Fundamental concepts and assumptions

5.1 Static and dynamic data

When exchanging parts library information, there is a need to exchange not only static properties, that may be modelled as data, but also dynamic behaviour that expresses e.g. how the value of a property may be deduced from the values of other properties. Expressions are one of the structures that enable the modelling of dynamic behaviours.

This part of ISO 13584 specifies a form for the unambiguous representation and exchange of computer interpretable expressions.

5.2 Syntax of expressions

The syntax of an expression consists of a set of symbols that represent the constants, the variables and the operators of this expression. In textual languages these symbols obey a set of rules usually defined in a grammar.

In this part of ISO 13584, the constants, the variables and the operators are represented as entity data types. The rules these entities shall obey are modelled in the EXPRESS schema that defines these entity data types.

5.3 Semantics of expressions

5.3.1 Semantic of expressions

An expression consists of operators and operands. The semantics of an expression is defined by the:

— range of the function performed by each operator;

— interpretation function that associates the corresponding value to each operand;

— evaluation function that computes the result of each operator when applied to its operands.

In this part of ISO 13584, the interpretation function shall be modelled by subtyping the **variable_semantics** entity. The evaluation function is not addressed in this part of ISO 13584.

5.3.2 Exchange time and evaluation time

The processing of expressions can be distinguished between two types of procedures.

— At exchange time an expression is represented by its structure. The corresponding directed acyclic graph is modelled and exchanged, and it is not required that the variables are bound to any value, but their semantics are known. Static analyses (acyclicity of a graph, type checking, mappability to the SQL language) can be performed at this level.

— At evaluation time, the expression can be assigned a value. At this stage, all the variables occurring in the expression must be bound to a value. Dynamic evaluation of an expression, testing and debugging, can be performed at this level.

Only the static analysis that correspond to exchange time are addressed in this part of ISO 13584. The informative annex F provides an overview of the different analyses of expressions.

5.4 Levels of abstraction in expression modelling

The operators used in the **ISO13584_generic_expressions_schema** allow the specification of abstract generic expressions that can be specialised for different purposes and on different data types.

5.4.1 Specialisation of the ISO13584_generic_expressions_schema

The **ISO13584_expressions_schema** is one specialisation of the **ISO13584_generic_expressions_schema**. The operators used in the **ISO13584_expressions_schema** belong to a subset of the operators defined in ISO 10303-11. Their range and evaluation function shall conform to the specification given in ISO 10303-11.

The operands are either constants, represented by literal values, variables or other expressions. Variables are strongly typed and it is assumed that, when the expression is evaluated in some context, an interpretation function provides each variable with a value conforming to its data type. When this condition is false, the expression evaluation results in an error.

The informative annex E discusses the details of such a specialisation process and outlines a methodology for other possible specialisation's of the **ISO13584_generic_expressions_schema**.

5.4.2 Specialisation of the ISO13584_expressions_schema

The **ISO13584_expression_schema** may itself be specialised to meet the requirements that are not addressed by the numeric-valued, string-valued or Boolean-valued operators defined in the EXPRESS language reference manual.

The specialisation of the **ISO13584_expressions_schema** consists either (1) in enlarging the schema by adding the definitions of new entities which carry the semantics of functions returning an integer, a real, a Boolean or a string value, or (2) in defining new sub-types of the **variable_semantics**.

The entities that carry the semantics of functions shall be defined as subtypes of the corresponding **defined_function** entity. Indeed, a function returning an integer, a real, a Boolean or a string value respectively shall be defined as a subtype of the **integer_defined_function**, **real_defined_function**, **boolean_defined_function** and **string_defined_function**.

Such a specialisation is compatible with the strong data type checking that results from the rules of the **ISO13584_expression_schema.**

The informative annex E discusses the details of such a specialisation process and outlines a methodology for specialisation of the **ISO13584_expressions_schema**.

5.5 Modelling a variable

A variable has three aspects:

— it is a (syntactic) symbol that may be used to build an expression;

— it is associated with a data type that defines the domain of its value;

— it is associated with a semantics that defines its meaning, and therefore its value at evaluation time.

5.5.1 Syntactic representation

In this part of ISO 13584, a variable is an instance of the **generic_variable** entity. An instance of such a data type is associated with an identifier, known as the instance identity, that constitutes the symbol of the corresponding variable when it is used in an expression.

5.5.2 Domain of values for a variable

The domain of values for a variable is modelled by subtyping the **generic_variable** entity. This approach, known as strong typing, ensures that the data type of any expression may be computed (synthesised) at exchange time.

5.5.3 Semantics of a variable

In the **ISO13584_generic_expression_schema**, shall be variables associated with a **variable_semantics.** This entity is defined as an ABSTRACT SUPERTYPE and shall be subtyped wherever a particular semantics is intended to be used. The description of a subtype of a **variable_semantics** shall contain the description of the context within which the variable shall be used. This description shall be accompanied with the description of the interpretation function that associates a value with this variable.

EXAMPLE The EXPRESS notation SELF.a is an example of **variable semantics**. Such a variable can only be used in the context of some instance of some class where its syntactic representation appears. Within this context, the interpretation function is the function that associates to this variable the value of the "a" attribute of this instance.

The informative annex D discusses the different approaches that may be used to define **variable_semantics** subtypes that meet some specific requirements. It gives the details of the definition of the interpretation function through the **variable_semantics** entity.

5.6 Mappability to the SQL language

In programming languages, expressions are commonly used to access databases. The SQL language only allows a small number of operators and restricts the use of expressions to a given category of expressions. In order to make a clear distinction between those expressions that are SQL mappable and the other expressions, a specific derived Boolean attribute states whether or not an expression is mappable onto the SQL language.

6 ISO13584_generic_expressions_schema

This clause defines the requirements for the **ISO13584_generic_expression_schema**. The following EXPRESS declaration introduces the **ISO13584_generic_expression_schema** block.

EXPRESS specification:

```
*)
SCHEMA ISO13584_generic_expressions_schema;
(*
```

6.1 Introduction

The **ISO13584_generic_expressions_schema** is an abstract resource that provides a common framework for the set of all the possible expressions.

EXAMPLE In a parametric shape model, a real value may be represented as an expression of which operands are geometric representation items, e.g., *distance* (point_1, point_2). A geometric item may be specified by a geometric operator, e.g., *centre_of* (circle_1).

The underlying structure is an acyclic graph, where nodes are operators, literal values or variables, and where links represent the relation of operators to their arguments, that are expressions themselves.

The following capabilities are provided in the **ISO13584_generic_expressions_schema**:

— generic constants corresponding to the values of the manipulated data type;

— generic variables that stand for values interpreted from the context;

— unary expressions for unary operators;

— binary expressions for binary operators;

— multiple arity expressions for multiple-arity operators.

Each specific kind of expressions is obtained by specialising (actually subtyping) the constants, the variables and the operators into those of the considered data type(s).

6.2 ISO13584_generic_expressions_schema entity definitions

6.2.1 Generic_expression

A **generic_expression** entity is the information model of a **generic_expression** as defined in clause 3.1.6. It is the ABSTRACT SUPERTYPE of all the possible expressions. In order to be able to assert the acyclicity of expressions, it is subtyped according to its arity. When a subtype of a **generic_expression** is not a subtype of either a **simple_generic_expression**, or an **unary_generic_expression**, or a **binary_generic_expression**, or a **multiple_arity_generic_expression** it shall not contain any variable.

EXPRESS specification:

```
*)
ENTITY generic_expression
ABSTRACT SUPERTYPE OF(ONEOF(simple_generic_expression,
                            unary_generic_expression,
                            binary_generic_expression,
                            multiple_arity_generic_expression));
WHERE
     WR1: is_acyclic(SELF);
END_ENTITY;
(*
```

Formal propositions:

WR1 : the graph associated with the described expression shall be acyclic.

Informal propositions:

IP1 : either a **generic_expression** shall not contain any **generic_variable** or this **generic_expression** shall be a subtype of either a **simple_generic_expression**, or an **unary_generic_expression**, or a **binary_generic_expression**, or a **multiple_arity_generic_expression**.

NOTE This informal proposition ensures that the set of variables computed by the **used_variable** function (see: 5.3.2.) includes all the variables involved in the **generic_expression**.

6.2.2 Simple_generic_expression

A **simple_generic_expression** is a generic expression that represents either a generic variable or a generic literal.

EXPRESS specification:

```
*)
ENTITY simple_generic_expression
ABSTRACT SUPERTYPE OF (ONEOF(generic_literal, generic_variable))
SUBTYPE OF (generic_expression);
END_ENTITY;
(*
```

6.2.3 Generic_literal

A **generic_literal** is an abstract constant that can be involved in a generic expression.

EXPRESS specification:

```
*)
ENTITY generic_literal
ABSTRACT SUPERTYPE
SUBTYPE OF (simple_generic_expression);
END_ENTITY;
(*
```

6.2.4 Generic_variable

A **generic_variable** is an abstract variable that can be involved in a generic expression. A **generic_variable** shall be subtyped to specify the data type of its permitted values. It shall be referenced by an environment that represents the interpretation function that associates a value with this variable.

NOTE **real_numeric_variable**, **integer_numeric_variable**, **boolean_variable** and **string_variable** are examples of subtypes of **generic_variable**. These subtypes are defined in clause 6.

EXPRESS specification:

```
*)
ENTITY generic_variable
ABSTRACT SUPERTYPE
SUBTYPE OF (simple_generic_expression);
INVERSE
     interpretation :
            environment FOR syntactic_representation;
END_ENTITY;
(*
```

Attribute definitions:

interpretation: the **environment** that enables to associate a value with the variable.

6.2.5 Variable_semantics

A **variable_semantics** entity is used to represent the meaning of a **generic_variable**. It is an ABSTRACT SUPERTYPE that shall be subtyped wherever a **variable_semantics** is used. A **variable_semantics** shall specify the context within which the variable shall be used together with the interpretation function that associates a value with this variable

EXAMPLE When modelling a class of components to which properties "a" and "b", with integer values apply. The constraint, stating that for every instance of this class, its value for "a" shall be greater than its value for "b", may be modelled through:

— two instances of **real_variable**;

— two instances of **self_property_semantics**, subtype of **variable_semantics** that refers respectively to property "a" and property "b";

— two instances of **environment** that associate each variable with its corresponding **variable_semantics**, and

— a **boolean_expression** between the two **real_variable**s that specifies that the first one is greater that the second one.

EXPRESS specification:

```
*)
ENTITY variable_semantics
ABSTRACT SUPERTYPE;
END_ENTITY;
(*
```

6.2.6 Environment

An **environment** entity is an association between the syntax and the semantics of a variable. It represents the interpretation function that associates a value to a **generic_variable.**

EXPRESS specification:

```
*)
ENTITY environment;
    syntactic_representation: generic_variable;
    semantics: variable_semantics;
END_ENTITY;
(*
```

Attribute definitions:

syntactic_representation: the **generic_variable** that stands for the value of the variable.

semantics: the meaning of the variable that includes the mechanism to access its value.

6.2.7 Unary_generic_expression

An **unary_generic_expression** is the ABSTRACT SUPERTYPE of all the unary operators.

EXPRESS specification:

```
*)
ENTITY unary_generic_expression
ABSTRACT SUPERTYPE
SUBTYPE OF(generic_expression);
     operand: generic_expression;
END_ENTITY;
(*
```

Attribute definitions:

operand: a generic expression that represents the operand of the unary operator.

6.2.8 Binary_generic_expression

A **binary_generic_expression** is the ABSTRACT SUPERTYPE of all the binary operators.

EXPRESS specification:

```
*)
ENTITY binary_generic_expression
ABSTRACT SUPERTYPE
SUBTYPE OF(generic_expression);
     operands: LIST [2:2] OF generic_expression;
END_ENTITY;
(*
```

Attribute definitions:

operands: a list of two generic expressions that represent the two operands of the binary operator.

6.2.9 Multiple_arity_generic_expression

A **multiple_arity_generic_expression** is the ABSTRACT SUPERTYPE of all the multiple-arity operators.

EXPRESS specification:

```
*)
ENTITY multiple_arity_generic_expression
ABSTRACT SUPERTYPE
SUBTYPE OF(generic_expression);
       operands: LIST [2:?] OF generic_expression;
END_ENTITY;
(*
```

Attribute definitions:

operands: is a list of generic expressions that represent the operands of the multiple-arity operator.

6.3 ISO13584_generic_expressions_schema function definitions

This clause introduces the functions of the **ISO13584_generic_expressions_schema**.

6.3.1 Is_acyclic function

The **is_acyclic** function checks that there is no cycle in the reference graph of the expression.

Two functions are defined. The first one (named **is_acyclic**) has as argument the **generic_expression** that shall be checked. The function then calls the second one (**acyclic**) with two arguments: the **generic_expression** and an empty set. This set will be updated such that it contains the set of all the nodes already visited in the chain of recursive calls down to this node.

Is_acyclic returns TRUE if the underlying structure of **arg** is a directed acyclic graph, if not, it returns FALSE.

NOTE The informative annex F discusses the role of such a structure for the static analysis (e.g. type control, variable collection and so on) and dynamic analysis (e.g. evaluation) of an expression.

EXPRESS specification:

```
*)
FUNCTION is_acyclic (arg: generic_expression): BOOLEAN;
RETURN (acyclic (arg, []));
END_FUNCTION ; -- is_acyclic

FUNCTION acyclic (arg1: generic_expression;
           arg2: SET OF generic_expression): BOOLEAN;

LOCAL
     result: BOOLEAN;
END_LOCAL;

IF ('ISO13584_GENERIC_EXPRESSIONS_SCHEMA.SIMPLE_GENERIC_EXPRESSION'
     IN TYPEOF (arg1))
THEN
     RETURN (TRUE);
END_IF;

IF arg1 IN arg2
THEN
     RETURN (FALSE);
END_IF;

IF 'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.UNARY_GENERIC_EXPRESSION'
     IN TYPEOF (arg1)
THEN
     RETURN
     (acyclic(arg1\unary_generic_expression.operand,arg2+[arg1]));
END_IF;

IF 'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.BINARY_GENERIC_EXPRESSION'
     IN TYPEOF (arg1)
THEN
     RETURN
     (acyclic(arg1\binary_generic_expression.operands[1],arg2+[arg1])
     AND
     acyclic(arg1\binary_generic_expression.operands[2],arg2+[arg1]));
```

```
END_IF;

IF
'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.MULTIPLE_ARITY_GENERIC_EXPRESSION'
     IN TYPEOF (arg1)
THEN
     result := TRUE;
     REPEAT i := 1 TO
            SIZEOF (arg1\multiple_arity_generic_expression.operands);
       result := result AND
       acyclic(arg1\multiple_arity_generic_expression.operands[i],
arg2+[arg1]);
     END_REPEAT;

     RETURN (result);
END_IF;

END_FUNCTION; -- acyclic
(*
```

6.3.2 Used_variables function

The **used_variables** function walks through the whole generic expression graph collecting all the variables used and finally returning them. It traverses the directed acyclic graph representing a **generic_expression**.

NOTE This function is not used in the **ISO13584_generic_expression_schema**. It is a resource intended to be used in the schemata that use the **ISO13584_generic_expressions_schema** or its possible specialisation's. For example, it is used in several parts of ISO13584 to write constraints on the variables occurring in expressions.

EXPRESS specification:

```
*)
FUNCTION used_variables (arg : generic_expression) :
            SET OF generic_variable;

LOCAL
     result : SET OF generic_variable := [];
END_LOCAL;

IF 'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.GENERIC_VARIABLE'
     IN TYPEOF (arg)
THEN
     RETURN ([arg]);
END_IF;

IF 'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.UNARY_GENERIC_EXPRESSION'
     IN TYPEOF (arg)
THEN
     RETURN (used_variables (arg\unary_generic_expression.operand));
END_IF;

IF 'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.BINARY_GENERIC_EXPRESSION'
```

```
     IN TYPEOF (arg)
THEN
     RETURN(used_variables(arg\binary_generic_expression.operands[1])
       + used_variables (arg\binary_generic_expression.operands[2]));
END_IF;

IF
'ISO13584_GENERIC_EXPRESSIONS_SCHEMA.MULTIPLE_ARITY_GENERIC_EXPRESSION'
     IN TYPEOF (arg)
THEN
     REPEAT i := 1 TO
       SIZEOF(arg\multiple_arity_generic_expression.operands);
       result := result + used_variables(
            arg\multiple_arity_generic_expression.operands[i]);
     END_REPEAT;

     RETURN (result);
END_IF;
RETURN ([ ]);      -- in this case the subtype shall not contain
                        -- any variable (see IP1 in generic_expression)
END_FUNCTION; -- used_variables

END_SCHEMA; -- ISO13584_generic_expressions_schema
(*
```

7 ISO13584_expressions_schema

This clause defines the requirements for the **ISO13584_expressions_schema.** The following EXPRESS declaration introduces the **ISO13584_expressions_schema** blockand identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA ISO13584_expressions_schema;

REFERENCE FROM ISO13584_generic_expressions_schema(
           generic_expression,
           simple_generic_expression,
           generic_variable,
           generic_literal,
           unary_generic_expression,
           binary_generic_expression,
           multiple_arity_generic_expression);
(*
```

7.1 Introduction

The **ISO13584_expressions_schema** provides for modelling numeric, Boolean, and string expressions built with the EXPRESS operators and functions. Names for operators and functions are

taken as far as possible as defined in ISO 10303-11. They have the same semantics as the corresponding elements in ISO 10303-11.

The underlying structure is inherited from the **ISO13584_generic_expression_schema**. It is a directed acyclic graph, where nodes are operators, literal values or variables, and where links represent the relation of operators to its arguments, that are expressions themselves.

The **ISO13584_expressions_schema** ensures strong typing of an expression. Any **expression** stands either for a number, or a string or a Boolean value. This data type may be computed from the structure of the **expression**. To enable to involve in an **expression** a **generic_expression** or an application defined function of which the data type of result is either a number, or a string or a Boolean value, the **ISO13584_expressions_schema** contains the following abstract entities: **numeric_defined_function**, **integer_defined_function**, **real_defined_function**, **string_defined_function**, and **boolean_defined_function**, The value provided by any of these entities shall conform to the data type of value associated with that entity. A **generic_expression** that is not a subtype of one of these entities is assumed to have a value whose data type is not one of the types modelled by the **ISO13584_expressions_schema** .

The **ISO13584_expressions_schema** include the information that an expression may, or may not , be mapped onto the Structured Query Language (SQL) defined by ISO/IEC 9075. To enable the involvement in an **expression** a **generic_expression** or an application defined function that may be mapped onto SQL, the **ISO13584_expressions_schema** contains the following abstract entity: **SQL_mappable_defined_function**. A **generic_expression** that is a subtype of this entity shall be mappable onto SQL. A **generic_expression** that is not a subtype of this entity is assumed to have a structure that is not mappable on the SQL language.

Expressions are defined by subtyping **generic_expression** from the **ISO13584_generic_expressions_schema**. Each of the constants, variables, and operators defined in the **ISO13584_expressions_schema** are subtypes of a corresponding **ISO13584_generic_expressions_schema**. This correspondence is defined by the arity of each operator.

7.2 ISO13584_expressions_schema overall entity definitions

7.2.1 Expression

An **expression** entity is a **generic_expression** restricted to numeric, Boolean or string domain of values.

EXPRESS specification:

```
*)
ENTITY expression
ABSTRACT SUPERTYPE OF (ONEOF (numeric_expression,
                              boolean_expression,
                              string_expression))
SUBTYPE OF (generic_expression);
END_ENTITY;
(*
```

7.2.1.1 Variable

A **variable** is an entity that stands for a value interpreted from some context.

EXPRESS specification:

```
*)
ENTITY variable
ABSTRACT SUPERTYPE OF (ONEOF (numeric_variable,
       boolean_variable,
       string_variable))
SUBTYPE OF(generic_variable);
END_ENTITY;
(*
```

7.2.1.2 Defined_function

A **defined_function** is a (strongly typed) function intended to be subtyped in the information models that use the **ISO13584_expressions_schema**.

EXPRESS specification:

```
*)
ENTITY defined_function
ABSTRACT SUPERTYPE OF ((ONEOF (numeric_defined_function,
                               string_defined_function,
                               boolean_defined_function)
                               )
                        ANDOR SQL_mappable_defined_function);
END_ENTITY;
(*
```

7.2.1.3 SQL_mappable_defined_function

A **SQL_mappable_defined_function** is a function intended to be subtyped in the information models that use the **ISO13584_expressions_schema** to specify that some **generic_expression** is mappable onto SQL.

EXPRESS specification:

```
*)
ENTITY SQL_mappable_defined_function
ABSTRACT SUPERTYPE
SUBTYPE OF (defined_function);
END_ENTITY;
(*
```

7.3 ISO13584_expressions_schema : entity definitions for numeric expressions

7.3.1 Numeric_expression

A **numeric_expression** is an expression of which the range is the NUMBER data type defined in 8.1.1 of ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY numeric_expression
ABSTRACT SUPERTYPE OF (ONEOF (simple_numeric_expression,
```

```
        unary_numeric_expression,
        binary_numeric_expression,
        multiple_arity_numeric_expression,
         length_function,
         value_function,
         numeric_defined_function))
SUBTYPE OF (expression);
DERIVE
     is_int: BOOLEAN := is_int_expr (SELF);
     sql_mappable: BOOLEAN := is_SQL_mappable (SELF);
END_ENTITY;
(*
```

Attribute definitions:

is_int: a Boolean attribute that indicates if an expression evaluates to integer or not.

sql_mappable: a Boolean attribute that indicates if an expression is mappable to the SQL query language.

7.3.2 Simple_numeric_expression

A **simple_numeric_expression** is either a numeric literal or a numeric variable.

EXPRESS specification:

```
*)
ENTITY simple_numeric_expression
ABSTRACT SUPERTYPE OF (ONEOF (literal_number, numeric_variable))
SUBTYPE OF (numeric_expression, simple_generic_expression);
END_ENTITY;
(*
```

7.3.3 Literal_number

A **literal_number** is an EXPRESS NUMBER literal . A **literal_number** may be either an integer literal or a real literal.

NOTE The EXPRESS NUMBER data type has the numeric values as domain.

EXPRESS specification:

```
*)
ENTITY literal_number
ABSTRACT SUPERTYPE OF (ONEOF (int_literal, real_literal))
SUBTYPE OF (simple_numeric_expression, generic_literal);
     the_value: NUMBER;
END_ENTITY;
(*
```

Attribute definitions:

the_value: a NUMBER literal value.

7.3.4 Int_literal

An **int_literal** is an EXPRESS INTEGER literal.

NOTE An EXPRESS INTEGER literal represents an integer VALUE that is composed of one or more digits.

EXPRESS specification:

```
*)
ENTITY int_literal
SUBTYPE OF (literal_number);
     SELF\literal_number.the_value: INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

the_value: an INTEGER literal value.

7.3.5 Real_literal

A **real_literal** is an EXPRESS REAL literal.

NOTE An EXPRESS REAL literal represents a real VALUE that is composed of a mantissa and an optional exponent; the mantissa shall include a decimal point.

EXPRESS specification:

```
*)
ENTITY real_literal
SUBTYPE OF (literal_number);
     SELF\literal_number.the_value: REAL;
END_ENTITY;
(*
```

Attribute definitions:

the_value: a REAL literal value.

7.3.6 Numeric_variable

A **numeric_variable** is a variable that stands for a number value.

EXPRESS specification:

```
*)
ENTITY numeric_variable
SUPERTYPE OF (ONEOF (int_numeric_variable,
                     real_numeric_variable))
SUBTYPE OF (simple_numeric_expression, variable);
WHERE
     WR1:   ('ISO13584_EXPRESSIONS_SCHEMA.INT_NUMERIC_VARIABLE'
            IN TYPEOF(SELF) ) OR
```

```
            ('ISO13584_EXPRESSIONS_SCHEMA.REAL_NUMERIC_VARIABLE'
            IN TYPEOF(SELF) );
END_ENTITY;
(*
```

Formal propositions:

WR1: **numeric_variable** can only be a real variable or an integer variable.

7.3.7 Int_numeric_variable

An **int_numeric_variable** is a variable that stands for an integer value.

EXPRESS specification:

```
*)
ENTITY int_numeric_variable
SUBTYPE OF (numeric_variable);
END_ENTITY;
(*
```

7.3.8 Real_numeric_variable

A **real_numeric_variable** is a variable that stands for a real value.

EXPRESS specification:

```
*)
ENTITY real_numeric_variable
SUBTYPE OF (numeric_variable);
END_ENTITY;
(*
```

7.3.9 Unary_numeric_expression

An **unary_numeric_expression** is an unary operator whose range is the NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY unary_numeric_expression
ABSTRACT SUPERTYPE OF (ONEOF (unary_function_call))
SUBTYPE OF (numeric_expression, unary_generic_expression);
     SELF\unary_generic_expression.operand  : numeric_expression;
END_ENTITY;
(*
```

Attribute definitions

operand: a numeric expression that represents the parameter of the unary operator.

7.3.10 Binary_numeric_expression

A **binary_numeric_expression** is a binary operator whose range is the NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY binary_numeric_expression
ABSTRACT SUPERTYPE OF (ONEOF (minus_expression,
                              div_expression,
                              mod_expression,
                              slash_expression,
                              power_expression,
                              binary_function_call))
SUBTYPE OF (numeric_expression, binary_generic_expression);
     SELF\binary_generic_expression.operands : LIST [2:2] OF
                                               numeric_expression;
END_ENTITY;
(*
```

Attribute definitions

operands: a list containing the two parameters of the binary operator.

7.3.11 Multiple_arity_numeric_expression

A **multiple_arity_numeric_expression** is a multiple arity operator whose range is the NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY multiple_arity_numeric_expression
ABSTRACT SUPERTYPE OF (ONEOF (plus_expression,
                              mult_expression,
                              multiple_arity_function_call))
SUBTYPE OF (numeric_expression, multiple_arity_generic_expression);
     SELF\multiple_arity_generic_expression.operands : LIST [2:?] OF
                                               numeric_expression;
END_ENTITY;
(*
```

Attribute definitions

operands: a list containing the parameters of the multiple arity operator. The length of this list is equal to the arity of the operator.

7.3.12 Length_function

The **length_function** entity carries the semantics of the LENGTH function defined in ISO 10303-11.

NOTE The length function returns the number of characters in a string.

— Parameters: a **string_expression** carrying a value of the string data type.

— Result: the returned value is the number of characters in the string and shall be greater than or equal to zero.

EXPRESS specification:

```
*)
ENTITY length_function
SUBTYPE OF (numeric_expression, unary_generic_expression);
     SELF\unary_generic_expression.operand: string_expression;
END_ENTITY;
(*
```

Attribute definitions

SELF\unary_generic_expression.operand: the string whose length is represented by the **length_function**.

7.3.13 Value_function

The **value_function** entity carries the semantics of the VALUE function defined in ISO 10303-11. When the value represented in the **string_expression** that corresponds to the **operand** inherited attribute is an integer, the **int_value_function** subtype shall be used.

NOTE The value function returns the numeric representation of a string.

— Parameters: a **string_expression** carrying a value of the string data type

— Result: a number corresponding to the string representation.

EXPRESS specification:

```
*)
ENTITY value_function
SUPERTYPE OF (int_value_function)
SUBTYPE OF (numeric_expression, unary_generic_expression);
     SELF\unary_generic_expression.operand: string_expression;
END_ENTITY;
(*
```

Attribute definitions

SELF\unary_generic_expression.operand: the string expression of which the value is represented by the **value_function**.

7.3.14 Int_value_function

The **int_value_function** entity carries the semantics of the VALUE function defined in ISO 10303-11, and asserts that the **string_expression** that corresponds to the **operand** inherited attribute represents an integer number.

NOTE The integer value function returns the integer number representation of a string.

— Parameters: a **string_expression** carrying a value of the string data type.

— Result: an integer number corresponding to the string representation.

EXPRESS specification:

```
*)
ENTITY int_value_function
SUBTYPE OF (value_function);
END_ENTITY;
(*
```

Informal proposition

IP1: the **string_expression** that corresponds to the **operand** attribute shall evaluate to a string that represents an integer number.

7.3.15 Numeric_defined_function

A **numeric_defined_function** entity is a function whose range is the NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY numeric_defined_function
ABSTRACT SUPERTYPE OF (ONEOF (integer_defined_function,
                                  real_defined_function))
SUBTYPE OF (numeric_expression, defined_function);
END_ENTITY;
(*
```

7.3.16 Plus_expression

The **plus_expression** entity carries the semantics of the '+' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **plus_expression** is an expression which evaluates to the sum of all its operands. In the **ISO13584_expression_schema**, all these operands shall evaluate to a number in the number data type domain.

EXPRESS specification:

```
*)
ENTITY plus_expression
SUBTYPE OF (multiple_arity_numeric_expression);
END_ENTITY;
(*
```

7.3.17 Minus_expression

The **minus_expression** entity carries the semantics of the '-' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **minus_expression** is an expression which evaluates to the difference of its operands. In the **ISO13584_expression_schema**, all these operands shall evaluate to a number in the number data type domain.

EXPRESS specification:

```
*)
ENTITY minus_expression
SUBTYPE OF (binary_numeric_expression);
END_ENTITY;
(*
```

7.3.18 Mult_expression

The **mult_expression** entity carries the semantics of the EXPRESS '*' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **mult_expression** is an expression which evaluates to the product of all its operands. In the **ISO13584_expression_schema**, all these operands shall evaluate to a number in the number data type domain.

EXPRESS specification:

```
*)
ENTITY mult_expression
SUBTYPE OF (multiple_arity_numeric_expression);
END_ENTITY;
(*
```

7.3.19 Div_expression

The **div_expression** entity carries the semantics of the EXPRESS 'DIV' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **div_expression** is an expression which evaluates to the integer division of its operands. In the **ISO13584_expression_schema**, all these operands shall evaluate to a number in the number data type domain. The result is an integer number.

EXPRESS specification:

```
*)
ENTITY div_expression
SUBTYPE OF (binary_numeric_expression);
END_ENTITY;
(*
```

Informal Proposition

IP1: the **SELF\binary_generic_expression.operands[2]** shall not evaluate to 0.

7.3.20 Mod_expression

The **mod_expression** entity carries the semantics of the EXPRESS 'MOD' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **mod_expression** is an expression which evaluates to the first argument value modulo operation of the second argument value. Its operands shall evaluate to a number in the number data type domain. The result is an integer number.

EXPRESS specification:

```
*)
ENTITY mod_expression
SUBTYPE OF (binary_numeric_expression);
END_ENTITY;
(*
```

7.3.21 Slash_expression

The **slash_expression** entity carries the semantics of the EXPRESS '/' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **slash_expression** is an expression which evaluates to the real division of its first operand by the second one. Its operands shall evaluate to a number in the number data type domain. The result is a number.

EXPRESS specification:

```
*)
ENTITY slash_expression
SUBTYPE OF (binary_numeric_expression);
END_ENTITY;
(*
```

Informal Proposition :

IP1: the **SELF\binary_generic_expression.operands[2]** shall not evaluate to 0.

7.3.22 Power_expression

The **power_expression** entity carries the semantics of the EXPRESS '**' operator, defined in ISO 10303-11, on NUMBER data type.

NOTE The **power_expression** is an expression which evaluates to the exponentiation of the first operand value to the second operand value. Its operands shall evaluate to a number in the number data type domain. The result is a number.

EXPRESS specification:

```
*)
ENTITY power_expression
SUBTYPE OF (binary_numeric_expression);
END_ENTITY;
(*
```

7.3.23 Unary_function_call

An **unary_function_call** is an unary operator on NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY unary_function_call
ABSTRACT SUPERTYPE OF (ONEOF (abs_function,
                              minus_function,
                              sin_function,
                              cos_function,
                              tan_function,
                              asin_function,
                              acos_function,
                              exp_function,
                              log_function,
                              log2_function,
                              log10_function,
                              square_root_function))
SUBTYPE OF (unary_numeric_expression);
END_ENTITY;
(*
```

7.3.24 binary_function_call

A **binary_function_call** is a binary operator on NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY binary_function_call
ABSTRACT SUPERTYPE OF (ONEOF (atan_function))
SUBTYPE OF (binary_numeric_expression);
END_ENTITY;
(*
```

7.3.25 Multiple_arity_function_call

A **multiple_arity_function_call** is a multiple-arity operator on NUMBER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY multiple_arity_function_call
ABSTRACT SUPERTYPE OF (ONEOF (maximum_function,
                              minimum_function))
SUBTYPE OF (multiple_arity_numeric_expression);
END_ENTITY;
(*
```

7.3.26 Abs_function

The **abs_function** entity carries the semantics of the 'ABS' function, defined in ISO 10303-11, on NUMBER data type, defined in ISO 10303-11.

NOTE The ABS function returns the absolute value of a number.

— Parameters: a **numeric_expression** carrying a value of the number data type.

— Result: the absolute value of the parameter. The returned data type is identical to the one of the parameter.

EXPRESS specification:

```
*)
ENTITY abs_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.27 Minus_function

The **minus_function** entity carries the semantics of the EXPRESS unary '-' operator, defined in ISO 10303-11, on NUMBER data type, defined in ISO 10303-11.

NOTE The minus function returns the opposite value of a number.

— Parameters: a **numeric_expression** carrying a value of the number data type.

— Result: the opposite value of the parameter. The returned data type is identical to the one of the parameter.

EXPRESS specification:

```
*)
ENTITY minus_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.28 Sin_function

The **sin_function** entity carries the semantics of the EXPRESS 'SIN' function defined in ISO 10303-11.

NOTE The SIN function returns the sine of an angle.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing an angle expressed in radians.

— Result: the sine of the parameter ($-1.0 \leq \text{result} \leq 1.0$).

EXPRESS specification:

```
*)
ENTITY sin_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.29 Cos_function

The **cos_function** entity carries the semantics of the EXPRESS 'COS' function defined in ISO 10303-11.

NOTE The COS function returns the cosine of an angle.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing an angle expressed in radians.

— Result: the cosine of the parameter (-1.0 ≤ result ≤ 1.0)

EXPRESS specification:

```
*)
ENTITY cos_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.30 Tan_function

The **tan_function** entity carries the semantics of the EXPRESS 'TAN' function defined in ISO 10303-11.

NOTE The TAN function returns the tangent of an angle.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing an angle expressed in radians.

— Result: the tangent of the parameter.

EXPRESS specification:

```
*)
ENTITY tan_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.31 Asin_function

The **asin_function** entity carries the semantics of the EXPRESS 'ASIN' function defined in ISO 10303-11.

NOTE The ASIN function returns the angle given a sine value.

— Parameters: a numeric_expression carrying a value of the number data type, representing the sine of an angle expressed in radians.

— Result: the angle expressed in radians (-Π/2 ≤result ≤ Π/2) whose sine is carried by the parameter.

EXPRESS specification:

```
*)
ENTITY asin_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.32 Acos_function

The **acos_function** entity carries the semantics of the EXPRESS 'ACOS' function defined in ISO 10303-11.

NOTE The ACOS function returns the angle given a cosine value.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing the cosine of an angle expressed in radians.

— Result: the angle expressed in radians ($-\Pi/2 \leq$result $\leq \Pi/2$) whose cosine is carried by the parameter.

EXPRESS specification:

```
*)
ENTITY acos_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.33 Exp_function

The **exp_function** entity carries the semantics of the EXPRESS 'EXP' function defined in ISO 10303-11.

NOTE The EXP function returns *e* (the base of the natural logarithm system) raised to the power of the value of the parameter.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing the value of the parameter.

— Result: the value *e* raised to the power of the value of the parameter.

EXPRESS specification:

```
*)
ENTITY exp_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.34 Log_function

The **log_function** entity carries the semantics of the EXPRESS 'LOG' function defined in ISO 10303-11.

NOTE The LOG function returns the natural logarithm of a number.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing the value of the parameter. It shall be greater than 0.

— Result: the real number which is the natural logarithm of the value of the parameter.

EXPRESS specification:

```
*)
ENTITY log_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.35 Log2_function

The **log2_function** entity carries the semantics of the EXPRESS 'LOG2' function defined in ISO 10303-11.

NOTE The LOG2 function returns the base 2 logarithm of a number.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing the value of the parameter. It shall be greater than 0.

— Result: the real number which is the base 2 logarithm of the value of the parameter.

EXPRESS specification:

```
*)
ENTITY log2_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.36 Log10_function

The **log10_function** entity carries the semantics of the EXPRESS 'LOG10' function defined in ISO 10303-11.

NOTE The LOG10 function returns the base 10 logarithm of a number.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing the value of the parameter. It shall be greater than 0.

— Result: the real number which is the base 10 logarithm of the value of the parameter.

EXPRESS specification:

```
*)
ENTITY log10_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.37 Square_root_function

The **square_root_function** entity carries the semantics of the EXPRESS 'SQRT' function defined in ISO 10303-11.

NOTE The SQRT function (square root) returns the non-negative square root of a number.

— Parameters: a **numeric_expression** carrying a value of the number data type, representing the value of the parameter. It shall be greater than or equal to 0.

— Result: the non negative real number which is the square root of the value of the parameter.

EXPRESS specification:

```
*)
ENTITY square_root_function
SUBTYPE OF (unary_function_call);
END_ENTITY;
(*
```

7.3.38 Atan_function

The **atan_function** entity carries the semantics of the EXPRESS 'ATAN' function defined in ISO 10303-11.

NOTE The ATAN function returns the angle given a tangent value represented by the parameters.

— Parameters: two **numeric_expressions** representing the value **SELF\binary_generic_expression[1]** / **SELF\binary_generic_expression[2].**

— Result: the angle in radians ($-\Pi/2 \leq$result $\leq \Pi/2$) whose tangent is represented by the value of the parameter. If the second expression corresponding to **SELF\binary_generic_expression[2]** evaluates to 0 then the result is $-\Pi/2$ or $\Pi/2$ depending on the sign of **SELF\binary_generic_expression[1].**

EXPRESS specification:

```
*)
ENTITY atan_function
SUBTYPE OF (binary_function_call);
END_ENTITY;
(*
```

7.3.39 Maximum_function

The **maximum_function** entity returns the maximum value of the list of values of its operands that shall evaluate to a number in the NUMBER data type domain.

EXPRESS specification:

```
*)
ENTITY maximum_function
SUBTYPE OF (multiple_arity_function_call);
END_ENTITY;
(*
```

7.3.40 Minimum_function

The **minimum_function** entity returns the minimum value of the list of values of its operands that shall evaluate to a number in the NUMBER data type domain.

EXPRESS specification:

```
*)
ENTITY minimum_function
SUBTYPE OF (multiple_arity_function_call);
END_ENTITY;
(*
```

NOTE The maximum function returns the minimum value of the list of values.

7.3.41 Integer_defined_function

An **integer_defined_function** is any function of which the range is the INTEGER data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY integer_defined_function
ABSTRACT SUPERTYPE
SUBTYPE OF (numeric_defined_function);
END_ENTITY ;
(*
```

Attribute definitions:

IP1: A subtype of an **integer_defined_function** may either be a **generic_expression**, or it shall not contain any **generic_variable**.

NOTE The **integer_defined_function** entity is a resource allowing to specialise this schema with any application defined function that returns a value in the integer data type domain.

7.3.42 Real_defined_function

A **real_defined_function** is any function of which the range is the REAL data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY real_defined_function
ABSTRACT SUPERTYPE
SUBTYPE OF (numeric_defined_function);
END_ENTITY ;
(*
```

Attribute definitions:

IP1: a subtype of a **real_defined_function** may either be a **generic_expression,** or it shall not contain any **generic_variable**.

NOTE The **real_defined_function** entity is a resource allowing to specialise this schema with any application defined function that returns a value in the real data type domain.

7.4 Boolean_expression

A **boolean_expression** is an **expression** for which the range is the BOOLEAN data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY boolean_expression
ABSTRACT SUPERTYPE OF (ONEOF (simple_boolean_expression,
                              unary_boolean_expression,
                              binary_boolean_expression,
                              multiple_arity_Boolean_expression,
                              comparison_expression,
                              interval_expression,
                              boolean_defined_function))
SUBTYPE OF (expression);
END_ENTITY;
(*
```

7.4.1 Simple_boolean_expression

A **simple_boolean_expression** is an unary operator for which the range is BOOLEAN, such a data type being defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY simple_boolean_expression
ABSTRACT SUPERTYPE OF (ONEOF (boolean_literal,
                              boolean_variable))
SUBTYPE OF (boolean_expression, simple_generic_expression);
END_ENTITY;
(*
```

7.4.2 Boolean_literal

A **boolean_literal** is an EXPRESS BOOLEAN literal.

NOTE The EXPRESS Boolean data type has as its domain the set containing the two literals TRUE and FALSE.

EXPRESS specification:

```
*)
ENTITY boolean_literal
SUBTYPE OF (simple_Boolean_expression, generic_literal);
     the_value: BOOLEAN;
END_ENTITY;
(*
```

Attribute definitions:

the_value: a BOOLEAN literal value**.**

7.4.3 Boolean_variable

A **boolean_variable** is a variable that stands for a Boolean value.

EXPRESS specification:

```
*)
ENTITY boolean_variable
SUBTYPE OF (simple_Boolean_expression, variable);
END_ENTITY;
(*
```

7.4.4 Unary_boolean_expression

An **unary_boolean_expression** is an unary operator for which the range is the BOOLEAN data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY unary_boolean_expression
ABSTRACT SUPERTYPE OF (ONEOF (not_expression, odd_function))
SUBTYPE OF (boolean_expression, unary_generic_expression);
END_ENTITY;
(*
```

7.4.5 Not_expression

The **not_expression** entity carries the semantics of the 'NOT' on Boolean data type operator defined in ISO 10303-11.

NOTE The NOT operator requires one Boolean operand and evaluates to a Boolean value.

EXPRESS specification:

```
*)
ENTITY not_expression
SUBTYPE OF (unary_Boolean_expression);
SELF\unary_generic_expression.operand: boolean_expression;
END_ENTITY;
(*
```

Attribute definitions:

SELF\unary_generic_expression.operand: the **boolean_expression** that represents the operand of the 'NOT' operator.

7.4.6 Odd_function

The **odd_function** entity carries the semantics of the EXPRESS 'ODD' operator defined in ISO 10303-11.

NOTE The ODD function returns TRUE or FALSE depending on whether a number is odd or even.

— Parameters: a numeric expression carrying a value of the integer data type.

— Result: the function returns TRUE when the value of the parameters modulo 2 equals to 1, and FALSE otherwise.

EXPRESS specification:

```
*)
ENTITY odd_function
SUBTYPE OF (unary_Boolean_expression);
     SELF\unary_generic_expression.operand: numeric_expression;
WHERE
     WR1: is_int_expr(SELF\numeric_expression);
END_ENTITY;
(*
```

Attribute definitions

SELF\unary_generic_expression: the **numeric_expression** that represents the operand of the 'ODD' function.

Formal proposition:

WR1: the operand shall be a **numeric_expression** for which the range is the same as the EXPRESS INTEGER data type.

7.4.7 Binary_boolean_expression

A **binary_boolean_expression** is a binary operator of which the range is the BOOLEAN data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY binary_boolean_expression
ABSTRACT SUPERTYPE OF (ONEOF (xor_expression, equals_expression))
SUBTYPE OF (boolean_expression, binary_generic_expression);
END_ENTITY;
(*
```

7.4.8 Multiple_arity_boolean_expression

A **multiple_arity_boolean_expression** is a multiple-arity operator for which the range is the same as the EXPRESS BOOLEAN data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY multiple_arity_boolean_expression
ABSTRACT SUPERTYPE OF (ONEOF (and_expression, or_expression))
SUBTYPE OF (boolean_expression, multiple_arity_generic_expression);
     SELF\multiple_arity_generic_expression.operands :
                                LIST [2:?] OF boolean_expression;
END_ENTITY;
(*
```

Attribute definitions:

operand: the occurrences of **boolean_expression** that represents the parameters of the **m-arity** operator.

7.4.9 Xor_expression

The **xor_expression** entity carries the semantics of the 'XOR' operator defined in ISO 10303-11.

NOTE The XOR operator requires two Boolean operands and evaluates to a Boolean value that is the exclusive disjunction of the two operands.

EXPRESS specification:

```
*)
ENTITY xor_expression
SUBTYPE OF (binary_boolean_expression);
     SELF\binary_generic_expression.operands:
                                  LIST [2:2] OF boolean_expression;
END_ENTITY;
(*
```

Attribute definitions:

operands: the two **boolean_expression** operands of the XOR operator that shall be **boolean_expression**s.

7.4.10 Equals_expression

The **equals_expression** entity carries the semantics of the ':=:' instance equal operator defined in ISO 10303-11, the domain of which is entities of data type **generic_expression**.

NOTE The entity instance equality operator accepts two compatible instance values that are **generic_expression** and evaluates to a Boolean value. In the context of the **ISO13584_expression_schema** the two **generic_expression**s instance values are compatible either if their data types are the same, or if one data type is a subtype of the other one.

EXPRESS specification:

```
*)
ENTITY equals_expression
SUBTYPE OF (binary_boolean_expression);
END_ENTITY;
(*
```

7.4.11 And_expression

The **and_expression** entity carries the semantics of the 'AND' operator defined in ISO 10303-11 and uses its associativity property to handle multiple arity.

NOTE The AND operator requires at least two Boolean expressions and evaluates to a Boolean value that is the conjunction of the two operands.

EXPRESS specification:

```
*)
ENTITY and_expression
SUBTYPE OF (multiple_arity_boolean_expression);
END_ENTITY;
(*
```

7.4.12 Or_expression

The **or_expression** entity carries the semantics of the 'OR' operator defined in ISO 10303-11 and uses its associativity property to handle multiple arity.

NOTE The OR operator requires at least two Boolean expressions and evaluates to a Boolean value that is the inclusive disjunction of the two operands.

EXPRESS specification:

```
*)
ENTITY or_expression
SUBTYPE OF (multiple_arity_boolean_expression);
END_ENTITY;
(*
```

7.4.13 Comparison_expression

The **comparison_expression** expressions carry the semantics of the different value comparison operators defined in ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

EXPRESS specification:

```
*)
ENTITY comparison_expression
ABSTRACT SUPERTYPE OF (ONEOF (comparison_equal,
                              comparison_greater,
                              comparison_greater_equal,
                              comparison_less,
                              comparison_less_equal,
                              comparison_not_equal,
                              like_expression))
SUBTYPE OF (boolean_expression, binary_generic_expression);
    SELF\binary_generic_expression.operands    :    LIST    [2:2]    OF
expression;
WHERE
    WR1: (('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
             IN TYPEOF(SELF\binary_generic_expression.operands[1]))
       AND
           'ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
             IN TYPEOF(SELF\binary_generic_expression.operands[2]))
    OR
         (('ISO13584_EXPRESSIONS_SCHEMA.BOOLEAN_EXPRESSION'
             IN TYPEOF(SELF\binary_generic_expression.operands[1]))
       AND
           ('ISO13584_EXPRESSIONS_SCHEMA.BOOLEAN_EXPRESSION'
           IN TYPEOF(SELF\binary_generic_expression.operands[2])))
    OR
          (('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
             IN TYPEOF(SELF\binary_generic_expression.operands[1]))
       AND
           ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
             IN TYPEOF(SELF\binary_generic_expression.operands[2]))) ;
END_ENTITY;
(*
```

Attribute definitions

operands: a list of **generic_expressions** that contains the expressions to be compared.

Formal proposition

WR1 : the operands of the **comparison_expression** shall be of compatible in type.

7.4.14 Comparison_equal

A **comparison_equal** carries the semantics of the value equal ('=') operator defined in ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

EXPRESS specification:

```
*)
ENTITY comparison_equal
SUBTYPE OF (comparison_expression);
END_ENTITY;
(*
```

NOTE The **comparison_equal** evaluates to TRUE if the two operands evaluate to the same value.

7.4.15 Comparison_greater

A **comparison_greater** carries the semantics of the greater than ('>') operator defined in of ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

NOTE The **comparison_greater** evaluates to TRUE if the first operand evaluates to a value greater than the second operand. The relevant orders are:

— mathematical ordering of the real number

— FALSE < TRUE

— lexicographic order on strings

EXPRESS specification:

```
*)
ENTITY comparison_greater
SUBTYPE OF (comparison_expression);
END_ENTITY;
(*
```

7.4.16 Comparison_greater_equal

A **comparison_greater_equal** carries the semantics of the greater than or equal ('>=') operator defined in of ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

NOTE The **comparison_greater_equal** evaluates to TRUE if the first operand evaluates to a value greater than or equal to the second operand. The relevant orders are:

— mathematical ordering of the real number

— FALSE < TRUE

— lexicographic order on strings

EXPRESS specification:

```
*)
```

```
ENTITY comparison_greater_equal
SUBTYPE OF (comparison_expression);
END_ENTITY;
(*
```

7.4.17 Comparison_less

A **comparison_less** carries the semantics of the less than ('<') operator defined in ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

NOTE The **comparison_less** evaluates to TRUE if the first operand evaluates to a value less than the second operand. The relevant orders are:

— mathematical ordering of the real number

— FALSE < TRUE

— lexicographic order on strings

EXPRESS specification:

```
*)
ENTITY comparison_less
SUBTYPE OF (comparison_expression);
END_ENTITY;
(*
```

7.4.18 Comparison_less_equal

A **comparison_less_or_equal** carries the semantics of the EXPRESS less than or equal ('<=') operator defined in ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

NOTE The **comparison_less_equal** evaluates to TRUE if the first operand evaluates to a value less than or equal to the second operand. The relevant orders are:

— mathematical ordering of the real number

— FALSE < TRUE

— lexicographic order on strings

EXPRESS specification:

```
*)
ENTITY comparison_less_equal
SUBTYPE OF (comparison_expression);
END_ENTITY;
(*
```

7.4.19 Comparison_not_equal

A **comparison_not_equal** carries the semantics of the value not equal ('<>') operator defined in ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

NOTE The **comparison_not_equal** evaluates to TRUE if the first operand evaluates to a value different from the second operand. The relevant orders are:

— mathematical ordering of the real number

— FALSE < TRUE

— lexicographic order on strings

EXPRESS specification:

```
*)
ENTITY comparison_not_equal
     SUBTYPE OF (comparison_expression);
END_ENTITY;
(*
```

7.4.20 Like_expression

A **like_expression** carries the semantics of the LIKE string matching operator defined in ISO 10303-11. The first operand is the target string. The second operand is the pattern string. The special characters (often called "wild cards") in the pattern string are defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY like_expression
SUBTYPE OF (comparison_expression);
WHERE
     WR1: ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
        IN TYPEOF(SELF\comparison_expression.operands[1])) AND
        ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
        IN TYPEOF(SELF\comparison_expression.operands[2]));
END_ENTITY;
(*
```

Formal propositions:

WR1: the operand list shall contain two **generic_expression**s that are **string_expression**s.

7.4.21 Interval_expression

An **interval_expression** carries the semantics of the interval expression defined in ISO 10303-11 restricted to the NUMBER, BOOLEAN and STRING EXPRESS data types. Both operands shall have the same data type.

NOTE An **interval_expression** expression tests whether or not a value falls within a given interval. It evaluates to TRUE if **interval_low** < = **interval_item** < = **interval_high**. The relevant orders are:

— mathematical ordering of the real number

— lexicographic order on strings

EXPRESS specification:

```
*)
ENTITY interval_expression
SUBTYPE OF (boolean_expression, multiple_arity_generic_expression) ;
DERIVE
     interval_low: generic_expression
            := SELF\multiple_arity_generic_expression.operands[1];
     interval_item:generic_expression
            := SELF\multiple_arity_generic_expression.operands[2];
     interval_high:generic_expression
            := SELF\multiple_arity_generic_expression.operands[3];
WHERE
     WR1:('ISO13584_EXPRESSIONS_SCHEMA.EXPRESSION'
                                IN TYPEOF(interval_low))
        AND ('ISO13584_EXPRESSIONS_SCHEMA.EXPRESSION'
                                IN TYPEOF(interval_item) )
        AND ('ISO13584_EXPRESSIONS_SCHEMA.EXPRESSION'
                                IN TYPEOF(interval_high));
     WR2:(('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
                                IN TYPEOF (SELF.interval_low))
            AND ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
                                IN TYPEOF (SELF.interval_high))
            AND ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
                                IN TYPEOF (SELF.interval_item)))
        OR
        (('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
                                IN TYPEOF(SELF.interval_low))
            AND ('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
                                IN TYPEOF(SELF.interval_item))
            AND ('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
                                IN TYPEOF(SELF.interval_high)));
END_ENTITY;
(*
```

Attribute definitions:

interval_low: the bound_1 operand of the interval expression (see 12.2.4 of ISO 10303-11).

interval_high: the bound_2 operand of the interval expression (see 12.2.4 of ISO 10303-11).

interval_item: the interval_item operand of the interval expression (see 12.2.4 of ISO 10303-11).

Formal propositions:

WR1: the data type of the operands shall be **expression**s.

WR2: the types of the **expression**s to be compared in the **interval_expression**s shall evaluates to comparable **expression**s.

7.4.22 Boolean_defined_function

A **boolean_defined_function** is any application-defined operator of which the range is the BOOLEAN data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY boolean_defined_function
ABSTRACT SUPERTYPE
SUBTYPE OF (defined_function, boolean_expression);
END_ENTITY ;
(*
```

Informal propositions:

IP1: a subtype of a **boolean_defined_function** may either be a **generic_expression,** or it shall not contain any **generic_variable**.

7.5 String_expression

A **string_expression** is an **expression** whose range is the EXPRESS STRING data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY string_expression
ABSTRACT SUPERTYPE OF (ONEOF (simple_string_expression,
                              index_expression,
                              substring_expression,
                              concat_expression,
                              format_function,
                              string_defined_function))
SUBTYPE OF (expression);
END_ENTITY;
(*
```

7.5.1 Simple_string_expression

A **simple_string_expression** is an unary operator of whose range is EXPRESS STRING data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY simple_string_expression
ABSTRACT SUPERTYPE OF (ONEOF (string_literal,
                              string_variable))
SUBTYPE OF (string_expression, simple_generic_expression);
END_ENTITY;
(*
```

7.5.2 String_literal

A **string_literal** is an EXPRESS STRING literal.

NOTE The EXPRESS STRING data type has as its domain sequences of characters.

EXPRESS specification:

```
*)
ENTITY string_literal
SUBTYPE OF (simple_string_expression, generic_literal);
     the_value: STRING;
END_ENTITY;
(*
```

Attribute definitions:

the value: a STRING literal value.

7.5.3 String_variable

A **string_variable** is a variable that stands for a STRING value.

EXPRESS specification:

```
*)
ENTITY string_variable
SUBTYPE OF (simple_string_expression, variable);
END_ENTITY;
(*
```

7.5.4 Index_expression

The **index_expression** carries the semantics of the string indexing operator ('[]') defined in ISO 10303-11.

NOTE The **index_expression** string operator takes two operands, the string value (represented by the **operand** attribute) being indexed by the index specification (represented by the **index** attribute). The resulting string value is the character at position the value of the **index** operator.

EXPRESS specification:

```
*)
ENTITY index_expression
SUBTYPE OF (string_expression, binary_generic_expression);
DERIVE
     operand:generic_expression:=
SELF\binary_generic_expression.operands[1];
     index:generic_expression:=
SELF\binary_generic_expression.operands[2];
WHERE
     WR1: ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
            IN TYPEOF(operand))
        AND ('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
            IN TYPEOF(index));
     WR2: is_int_expr (index);
END_ENTITY;
(*
```

Attribute definitions:

operand: the **string_expression** that represents the STRING.

index: the integer **numeric_expression** that indicates the index value.

Formal propositions:

WR1: the first operand shall be a **string_expression** and the second operand shall be a **numeric_expression**.

WR2: the position described by the index operand shall be an integer value.

Informal propositions:

IP1: the **index** shall evaluate to an INTEGER value greater than zero and less or equal to the length of the **operand** STRING.

7.5.5 Substring_expression

The **substring_expression** carries the semantics of the EXPRESS substring indexing operator ('[:]') defined in ISO 10303-11.

NOTE The **substring_expression** string operator takes three operands, the string value (represented by the **operand** attribute) being indexed by the index specification (represented by the **index1** and **index2** attributes). The **substring_expression** evaluates to a string value of length (**index2 - index1 + 1**). The resulting string value is equivalent to the sequence of characters at position **index1** through **index2**.

EXPRESS specification:

```
*)
ENTITY substring_expression
SUBTYPE OF (string_expression, multiple_arity_generic_expression);
DERIVE
     operand:generic_expression:=
            SELF\multiple_arity_generic_expression.operands[1];
     index1:generic_expression:=
            SELF\multiple_arity_generic_expression.operands[2];
     index2:generic_expression:=
            SELF\multiple_arity_generic_expression.operands[3];
WHERE
     WR1: ('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION'
                                     IN TYPEOF(operand))
            AND ('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
                                     IN TYPEOF(index1))
            AND ('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION'
                                     IN TYPEOF(index2));
     WR2: SIZEOF(SELF\multiple_arity_generic_expression.operands)=3;
     WR3: is_int_expr (index1);
     WR4: is_int_expr (index2);
END_ENTITY;
(*
```

Attribute definitions:

operand: the **string_expression** from which a substring is extracted**.**

index1: the **numeric_expression** that indicates the low bound of the substring in **operand**.

index2: the **numeric_expression** that indicates the high bound of the substring in **operand**.

Formal propositions:

WR1: operand shall be a **string_expression**, index1 and index2 shall be **numeric_expression** .

WR2: the **substring_expression operands** LIST shall contain three elements.

WR3: **index1** shall evaluate to an INTEGER value.

WR4: **index2** shall evaluate to an INTEGER value.

Informal propositions:

IP1: **index1** and **index2** shall evaluate to an INTEGER value greater than zero and less or equal to the length of the **operand** STRING.

IP2: **index1** value shall be less or equal to **index2** value.

7.5.6 Concat_expression

The **concat_expression** carries the semantics of the concatenation ('+') operator defined in ISO 10303-11, on the STRING data type defined in ISO 10303-11.

NOTE The **concat_expression** operator accepts at least two **string_expression**s as operands and evaluates to a string value resulting from the concatenation of all the elements of the **SELF\multiple_arity_generic_expression.operands** list. These elements are concatenated in the list order.

EXPRESS specification:

```
*)
ENTITY concat_expression
SUBTYPE OF (string_expression, multiple_arity_generic_expression);
     SELF\multiple_arity_generic_expression.operands:
                                   LIST [2 : ?] OF string_expression;
END_ENTITY;
(*
```

Attribute definitions:

SELF\multiple_arity_generic_expression.operands: the **string_expression**s to be concatenated.

7.5.7 Format_function

The **format_function** carries the semantics of the FORMAT function defined in ISO 10303-11.

NOTE The format function returns a formatted string representation of a number.

— Parameters: a **numeric_expression** corresponding to the **value_to_format** operand and which evaluates to a real or integer number and a **string_expression** corresponding to the **format_string** operand which evaluates to a string containing the formatting commands.

— Result: a string representation of the value of the **value_to_format** operand according to the value of the **format_string** operand. The formatting string contains special characters to indicate the appearance of the result. The different ways to describe the formatting string are defined in ISO 10303-11 PART 11.

EXPRESS specification:

```
*)
ENTITY format_function
SUBTYPE OF (string_expression, binary_generic_expression);
DERIVE
     value_to_format: generic_expression:=
            SELF\binary_generic_expression.operands[1];
     format_string:generic_expression:=
            SELF\binary_generic_expression.operands[2];
WHERE
     WR1: (('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_EXPRESSION')
                                    IN TYPEOF(value_to_format))
            AND (('ISO13584_EXPRESSIONS_SCHEMA.STRING_EXPRESSION')
                                    IN TYPEOF(format_string));
END_ENTITY;
(*
```

Attribute definitions:

value_to_format: the **numeric_expression** that is to be formatted.

format_string: the formatting commands that defines the appearance of the result.

Formal proposition:

WR1: the **value_to_format** shall be a **numeric_expression** and the **format_string** shall be a **string_expression**.

7.5.8 String_defined_function

A **string_defined_function** is any application-defined operator of which the range is the STRING data type defined in ISO 10303-11.

EXPRESS specification:

```
*)
ENTITY string_defined_function
ABSTRACT SUPERTYPE
SUBTYPE OF (defined_function, string_expression);
END_ENTITY ;
(*
```

Informal proposition:

IP1: a subtype of a **string_defined_function** may either be a **generic_expression,** or it shall not contain any **generic_variable**.

7.6 Functions to determine properties of the expression

The following EXPRESS declarations specifies different functions that enable the properties of **expression**s to be determined.

7.6.1 Is_int_expr

The **is_int_expr** function checks if a syntactically correct expression (see clause 7) evaluates to an integer value or not.

EXPRESS specification:

```
*)
FUNCTION is_int_expr (arg: numeric_expression) : BOOLEAN;

LOCAL
     i: INTEGER;
END_LOCAL;

IF 'ISO13584_EXPRESSIONS_SCHEMA.INT_LITERAL' IN TYPEOF(arg)
THEN
     RETURN (TRUE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.REAL_LITERAL' IN TYPEOF(arg)
THEN
     RETURN (FALSE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.INT_NUMERIC_VARIABLE' IN TYPEOF(arg)
THEN
     RETURN (TRUE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.REAL_NUMERIC_VARIABLE' IN TYPEOF(arg)
THEN
     RETURN (FALSE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.ABS_FUNCTION' IN TYPEOF(arg)
THEN
     RETURN (is_int_expr(arg\unary_numeric_expression.operand));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.MINUS_FUNCTION' IN TYPEOF(arg)
THEN
     RETURN (is_int_expr(arg\unary_numeric_expression.operand));
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.SIN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.COS_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.TAN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.ASIN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.ACOS_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.ATAN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.EXP_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LOG_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LOG2_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LOG10_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.SQUARE_ROOT_FUNCTION'
                                                  IN TYPEOF(arg))
THEN
     RETURN (FALSE);
```

```
END_IF;
IF  ('ISO13584_EXPRESSIONS_SCHEMA.PLUS_EXPRESSION' IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.MULT_EXPRESSION'
                                                    IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.MAXIMUM_FUNCTION'
                                                    IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.MINIMUM_FUNCTION'
                                                    IN TYPEOF(arg))
THEN
    REPEAT i :=1 TO SIZEOF (
           arg\multiple_arity_numeric_expression.operands);
    IF NOT
      is_int_expr(arg\multiple_arity_numeric_expression.operands[i])
    THEN
      RETURN (FALSE);
    END_IF;
    END_REPEAT;
    RETURN (TRUE);
END_IF;
IF  ('ISO13584_EXPRESSIONS_SCHEMA.MINUS_EXPRESSION' IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.POWER_EXPRESSION'
                                                    IN TYPEOF(arg))
THEN
    RETURN (is_int_expr(arg\binary_numeric_expression.operands[1])
      AND is_int_expr(arg\binary_numeric_expression.operands[2]));
END_IF;
IF  ('ISO13584_EXPRESSIONS_SCHEMA.DIV_EXPRESSION' IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.MOD_EXPRESSION' IN TYPEOF(arg))
THEN
    RETURN(TRUE);               (*always deliver an INTEGER result *)
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.SLASH_EXPRESSION' IN TYPEOF(arg)
THEN
    RETURN (FALSE);             (* always delivers a REAL result *)
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.LENGTH_FUNCTION' IN TYPEOF(arg)
THEN
    RETURN (TRUE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.VALUE_FUNCTION' IN TYPEOF(arg)
THEN
    IF 'ISO13584_EXPRESSIONS_SCHEMA.INT_VALUE_FUNCTION'
                                                    IN TYPEOF(arg)
    THEN
      RETURN (TRUE);
    ELSE
      RETURN (FALSE);
    END_IF;
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.INTEGER_DEFINED_FUNCTION'
                                                    IN TYPEOF(arg)
THEN
    RETURN(TRUE) ;
```

```
END_IF;
IF'ISO13584_EXPRESSIONS_SCHEMA.REAL_DEFINED_FUNCTION' IN TYPEOF(arg)
THEN
    RETURN(FALSE) ;
END_IF ;
IF 'ISO13584_EXPRESSIONS_SCHEMA.BOOLEAN_DEFINED_FUNCTION'
                                                    IN TYPEOF(arg)
THEN
    RETURN(FALSE) ;
END_IF ;
IF 'ISO13584_EXPRESSIONS_SCHEMA.STRING_DEFINED_FUNCTION'
                                                    IN TYPEOF(arg)
THEN
    RETURN (FALSE) ;
END_IF ;
(* If another generic_expression is involved that is not a subtype of
integer_defined_function then its result is not integer. *)
RETURN (FALSE);

END_FUNCTION; -- is_int_expr
(*
```

7.6.2 Is_SQL_mappable

The **is_SQL_mappable** function checks if the acyclic graph that represents an expression only contains elements that are mappable to SQL (Structured Query Language).

EXPRESS specification:

```
*)
FUNCTION is_SQL_mappable (arg: expression) : BOOLEAN;

LOCAL
    i: INTEGER;
END_LOCAL;

IF 'ISO13584_EXPRESSIONS_SCHEMA.SIMPLE_NUMERIC_EXPRESSION'
                                                    IN TYPEOF (arg)
THEN
    RETURN (TRUE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.SQL_MAPPABLE_DEFINED_FUNCTION'
                                                    IN TYPEOF (arg)
THEN
    RETURN (TRUE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.MINUS_FUNCTION' IN TYPEOF(arg)
THEN
    RETURN (is_SQL_mappable(arg\unary_numeric_expression.operand));
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.ABS_FUNCTION' IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.SIN_FUNCTION' IN TYPEOF(arg))
      OR ('ISO13584_EXPRESSIONS_SCHEMA.COS_FUNCTION' IN TYPEOF(arg))
```

```
       OR ('ISO13584_EXPRESSIONS_SCHEMA.TAN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.ASIN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.ACOS_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.ATAN_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.EXP_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LOG_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LOG2_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LOG10_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.SQUARE_ROOT_FUNCTION'
                                                    IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.VALUE_FUNCTION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LENGTH_FUNCTION'
                                                    IN TYPEOF(arg))
THEN
    RETURN (FALSE);
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.PLUS_EXPRESSION' IN TYPEOF(arg))
       OR('ISO13584_EXPRESSIONS_SCHEMA.MULT_EXPRESSION' IN TYPEOF(arg))
       OR('ISO13584_EXPRESSIONS_SCHEMA.MAXIMUM_FUNCTION'
                                                    IN TYPEOF(arg))
       OR('ISO13584_EXPRESSIONS_SCHEMA.MINIMUM_FUNCTION'
                                                    IN TYPEOF(arg))
THEN
    REPEAT i :=1 TO SIZEOF (
           arg\multiple_arity_numeric_expression.operands);
       IF NOT is_SQL_mappable(
           arg\multiple_arity_numeric_expression.operands[i])
       THEN
           RETURN (FALSE);
       END_IF;
    END_REPEAT;
RETURN (TRUE);
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.MINUS_EXPRESSION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.SLASH_EXPRESSION' IN
                                                    TYPEOF(arg))
THEN
       RETURN (is_SQL_mappable(
           arg\binary_numeric_expression.operands[1])
       AND is_SQL_mappable(arg\binary_numeric_expression.operands[2]));
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.DIV_EXPRESSION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.MOD_EXPRESSION' IN TYPEOF(arg))
       OR('ISO13584_EXPRESSIONS_SCHEMA.POWER_EXPRESSION'
                                                    IN TYPEOF(arg))
THEN
    RETURN (FALSE);             (* operators not supported by SQL *)
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.SIMPLE_BOOLEAN_EXPRESSION'
                                                    IN TYPEOF (arg)
THEN
    RETURN (TRUE);
END_IF;
```

```
IF 'ISO13584_EXPRESSIONS_SCHEMA.NOT_EXPRESSION' IN TYPEOF (arg)
THEN
     RETURN (is_SQL_mappable (arg\UNARY_GENERIC_EXPRESSION.OPERAND));
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.ODD_FUNCTION'IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.XOR_EXPRESSION'
                                                  IN TYPEOF (arg))
THEN
     RETURN (FALSE);
END_IF;
IF ('ISO13584_EXPRESSIONS_SCHEMA.AND_EXPRESSION' IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.OR_EXPRESSION' IN TYPEOF (arg))
THEN
     REPEAT i:=1 TO SIZEOF (
            arg\MULTIPLE_ARITY_BOOLEAN_EXPRESSION.OPERANDS);
       IF NOT is_SQL_mappable (
            arg\MULTIPLE_ARITY_BOOLEAN_EXPRESSION.OPERANDS[i])
       THEN
            RETURN (FALSE);
       END_IF;
     END_REPEAT;
     RETURN (TRUE);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.EQUALS_EXPRESSION' IN TYPEOF (arg)
THEN
     RETURN(is_SQL_mappable (
            arg\BINARY_GENERIC_EXPRESSION.OPERANDS [1])
       AND is_SQL_mappable(
            arg\BINARY_GENERIC_EXPRESSION.OPERANDS [2]));
END_IF;
IF  ('ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_EQUAL' IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_GREATER'
                                                  IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_GREATER_EQUAL'
                                                  IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_LESS'
                                                  IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_LESS_EQUAL'
                                                  IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_NOT_EQUAL'
                                                  IN TYPEOF (arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.LIKE_EXPRESSION'
                                                  IN TYPEOF (arg))
THEN
     RETURN (is_SQL_mappable (arg\COMPARISON_EXPRESSION.OPERANDS[1])
       AND is_SQL_mappable (arg\COMPARISON_EXPRESSION.OPERANDS[2]));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.INTERVAL_EXPRESSION' IN TYPEOF(arg)
THEN
     RETURN (is_SQL_mappable(arg\interval_expression.interval_low)
       AND is_SQL_mappable(arg\interval_expression.interval_high)
       AND is_SQL_mappable(arg\interval_expression.interval_item));
END_IF;
```

```
IF ('ISO13584_EXPRESSIONS_SCHEMA.NUMERIC_DEFINED_FUNCTION'
                                                     IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.BOOLEAN_DEFINED_FUNCTION'
                                                     IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.STRING_DEFINED_FUNCTION'
                                                     IN TYPEOF(arg))
THEN
       RETURN (FALSE) ;
END_IF;
(* It has been assumed that all the defined functions are not mappable
to the SQl language *)
IF 'ISO13584_EXPRESSIONS_SCHEMA.SIMPLE_STRING_EXPRESSION'
                                                     IN TYPEOF(ARG)
THEN
     RETURN (TRUE);
END_IF;
IF   ('ISO13584_EXPRESSIONS_SCHEMA.INDEX_EXPRESSION' IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.SUBSTRING_EXPRESSION'
                                                     IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.CONCAT_EXPRESSION'
                                                     IN TYPEOF(arg))
       OR ('ISO13584_EXPRESSIONS_SCHEMA.FORMAT_FUNCTION'
                                                     IN TYPEOF(arg))
THEN
     RETURN (FALSE);
END_IF;
(* If another generic_expression is involved that is not a subtype of
SQL_mappable_defined_function then it is not mappable on SQL. *)
     RETURN (FALSE);
END_FUNCTION; -- is_SQL_mappable
(*
```

7.6.3 Used_functions

The **used_function** function walks through the whole expression graph collecting all the **application_defined** functions and finally returning them. It traverses the directed acyclic graph representing an expression.

NOTE This function is a resource for the schemata that use the **ISO13584_expressions_schema** or its possible specialisation's. As an example, it is used in several parts of ISO 13584 to write constraints on the functions occurring in expressions.

EXPRESS specification:

```
*)
FUNCTION used_functions (arg : expression) : SET OF defined_function;

LOCAL
     result : SET OF defined_function := [];
END_LOCAL;

IF ('ISO13584_EXPRESSIONS_SCHEMA.DEFINED_FUNCTION' IN TYPEOF(arg))
THEN
     RETURN ( [arg] ) ;
```

```
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.UNARY_NUMERIC_EXPRESSION' IN
     TYPEOF (arg)
THEN
     RETURN (used_functions (arg\unary_numeric_expression.operand));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.BINARY_NUMERIC_EXPRESSION' IN
       TYPEOF (arg)
THEN
     RETURN (used_functions (arg\binary_numeric_expression.operands[1])
       + used_functions (arg\binary_numeric_expression.operands[2]));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.MULTIPLE_ARITY_NUMERIC_EXPRESSION' IN
       TYPEOF (arg)
THEN
     REPEAT i := 1 TO SIZEOF (
            arg\multiple_arity_numeric_expression.operands);
       result := result + used_functions (
            arg\multiple_arity_numeric_expression.operands[i]);
     END_REPEAT;
     RETURN (result);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.UNARY_GENERIC_EXPRESSION'
       IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\unary_generic_expression.operand));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.BINARY_BOOLEAN_EXPRESSION'
                                                    IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\binary_generic_expression.operands[1])
       + used_functions (
            arg\binary_generic_expression.operands[2]));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.MULTIPLE_ARITY_BOOLEAN_EXPRESSION'
                                                    IN TYPEOF (arg)
THEN
     REPEAT i := 1 TO
       SIZEOF (arg\multiple_arity_Boolean_expression.operands);
       result := result + used_functions(
            arg\multiple_arity_Boolean_expression.operands[i]);
     END_REPEAT;
     RETURN (result);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.COMPARISON_EXPRESSION'
                                                    IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\comparison_expression.operands[1])
            + used_functions (arg\comparison_expression.operands[2]));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.INTERVAL_EXPRESSION' IN TYPEOF(arg)
THEN
     RETURN (used_functions(arg\interval_expression.interval_low)
```

```
       + used_functions(arg\interval_expression.interval_high)
       + used_functions(arg\interval_expression.interval_item));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.INDEX_EXPRESSION' IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\index_expression.operand)
       + used_functions (arg\index_expression.index));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.SUBSTRING_EXPRESSION' IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\substring_expression.operand)
       + used_functions (arg\substring_expression.index1)
       + used_functions (arg\substring_expression.index2));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.CONCAT_EXPRESSION' IN TYPEOF (arg)
THEN
     REPEAT i := 1 TO SIZEOF (arg\concat_expression.operands);
       result := result + used_functions (
            arg\concat_expression.operands[i]);
     END_REPEAT;
     RETURN (result);
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.FORMAT_FUNCTION' IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\format_function.value_to_format)
       + used_functions (arg\format_function.format_string));
END_IF;
IF 'ISO13584_EXPRESSIONS_SCHEMA.LIKE_EXPRESSION' IN TYPEOF (arg)
THEN
     RETURN (used_functions (arg\like_expression.operands[1])
       + used_functions (arg\like_expression.operands[2]));
END_IF;
RETURN ([ ]);

END_FUNCTION; -- used_functions

END_SCHEMA; -- ISO13584_expressions_schema
(*
```

Annex A

(normative)

Short names of entities

Table A.1 provides the short names of entities specified in this part of ISO 13584. Requirements on the use of short names are found in the implementation methods included in ISO 10303.

Table A.1 — Short names of entities

Long Name	Short Name
ABS_FUNCTION	ABSFNC
ACOS_FUNCTION	ACSFNC
AND_EXPRESSION	ANDEXP
ASIN_FUNCTION	ASNFNC
ATAN_FUNCTION	ATNFNC
BINARY_BOOLEAN_EXPRESSION	BNBLEX
BINARY_FUNCTION_CALL	BNFNCL
BINARY_GENERIC_EXPRESSION	BNGNEX
BINARY_NUMERIC_EXPRESSION	BNNMEX
BOOLEAN_DEFINED_FUNCTION	BLDFFN
BOOLEAN_EXPRESSION	BLNEXP
BOOLEAN_LITERAL	BLNLTR
BOOLEAN_VARIABLE	BLNVRB
COMPARISON_EQUAL	CMPEQL
COMPARISON_EXPRESSION	CMPEXP
COMPARISON_GREATER	CMPGRT
COMPARISON_GREATER_EQUAL	CMGREQ
COMPARISON_LESS	CMPLSS
COMPARISON_LESS_EQUAL	CMLSEQ
COMPARISON_NOT_EQUAL	CMNTEQ
CONCAT_EXPRESSION	CNCEXP
COS_FUNCTION	CSFNC
DEFINED_FUNCTION	DFNFNC
DIV_EXPRESSION	DVEXP
ENVIRONMENT	ENVRNM
EQUALS_EXPRESSION	EQLEXP
EXPRESSION	EXPRSS
EXP_FUNCTION	EXPFNC

Table A.1 (continued)

Long Name	Short Name
FORMAT_FUNCTION	FRMFNC
GENERIC_EXPRESSION	GNREXP
GENERIC_LITERAL	GNRLTR
GENERIC_VARIABLE	GNRVRB
SQUARE_ROOT_FUNCTION	SQRTFN
STRING_DEFINED_FUNCTION	STDFFN
STRING_EXPRESSION	STREXP
STRING_LITERAL	STRLTR
STRING_VARIABLE	STRVRB
SUBSTRING_EXPRESSION	SBSEXP
TAN_FUNCTION	TNFNC
UNARY_BOOLEAN_EXPRESSION	UNBLEX
UNARY_FUNCTION_CALL	UNFNCL
UNARY_GENERIC_EXPRESSION	UNGNEX
UNARY_NUMERIC_EXPRESSION	UNNMEX
VALUE_FUNCTION	VLFNC
VARIABLE	VRBL
VARIABLE_SEMANTICS	VRBSMN
XOR_EXPRESSION	XREXP

Annex B

(normative)

Information object registration

B.1 Document identification

In order to provide for unambiguous identification of an information object in an open system, the object identifier

{ ISO standard 13584 part (20) version(1) }

is assigned to this part of ISO 13584. The meaning of this value is defined in ISO 8824-1, and is described in ISO 13584-1.

B.2 Schema identification

B.2.1 ISO13584_generic_expressions_schema

The **ISO13584_generic_expressions_schema** (see clause 5) is assigned the object identifier

{ ISO standard 13584 part (20) version(1) object(1) ISO13584-generic-expressions-schema(1) }

B.2.2 ISO13584_expressions_schema

The **ISO13584_expressions_schema** (see clause 6) is assigned the object identifier

{ ISO standard 13584 part (20) version(1) object(1) ISO13584-expressions-schema(2) }

Annex C

(informative)

EXPRESS-G diagrams

Figure C.1 through C.13 correspond to the EXPRESS given in clauses 5 and 6. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex A of ISO 10303-11.

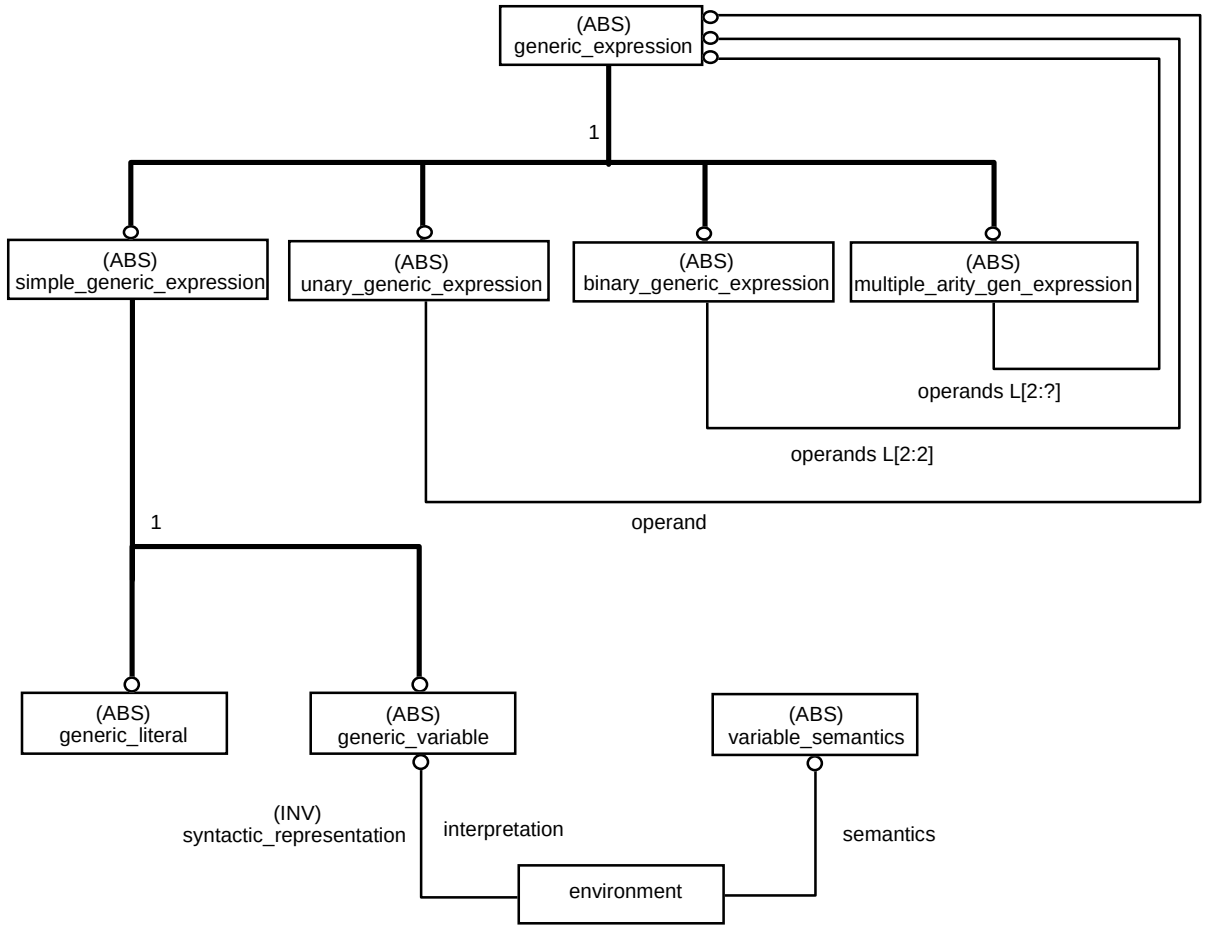

Figure C.1 — ISO13584_generic_expressions_schema - EXPRESS-G diagram 1 of 1

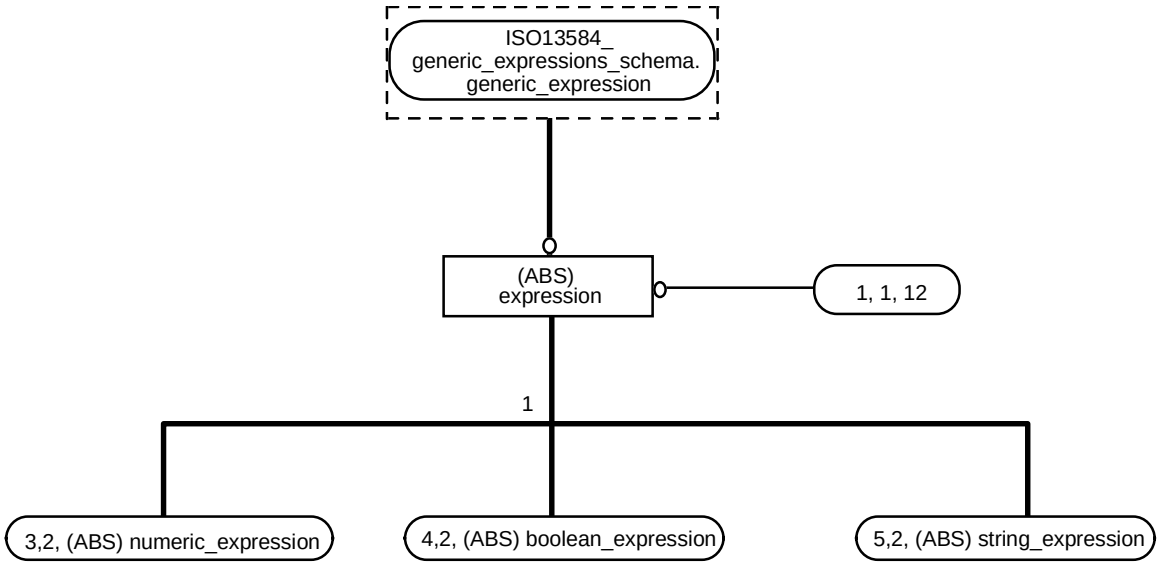

Figure C.2 — ISO13584_expressions_schema - EXPRESS-G diagram 1 of 12- String typing of expressions

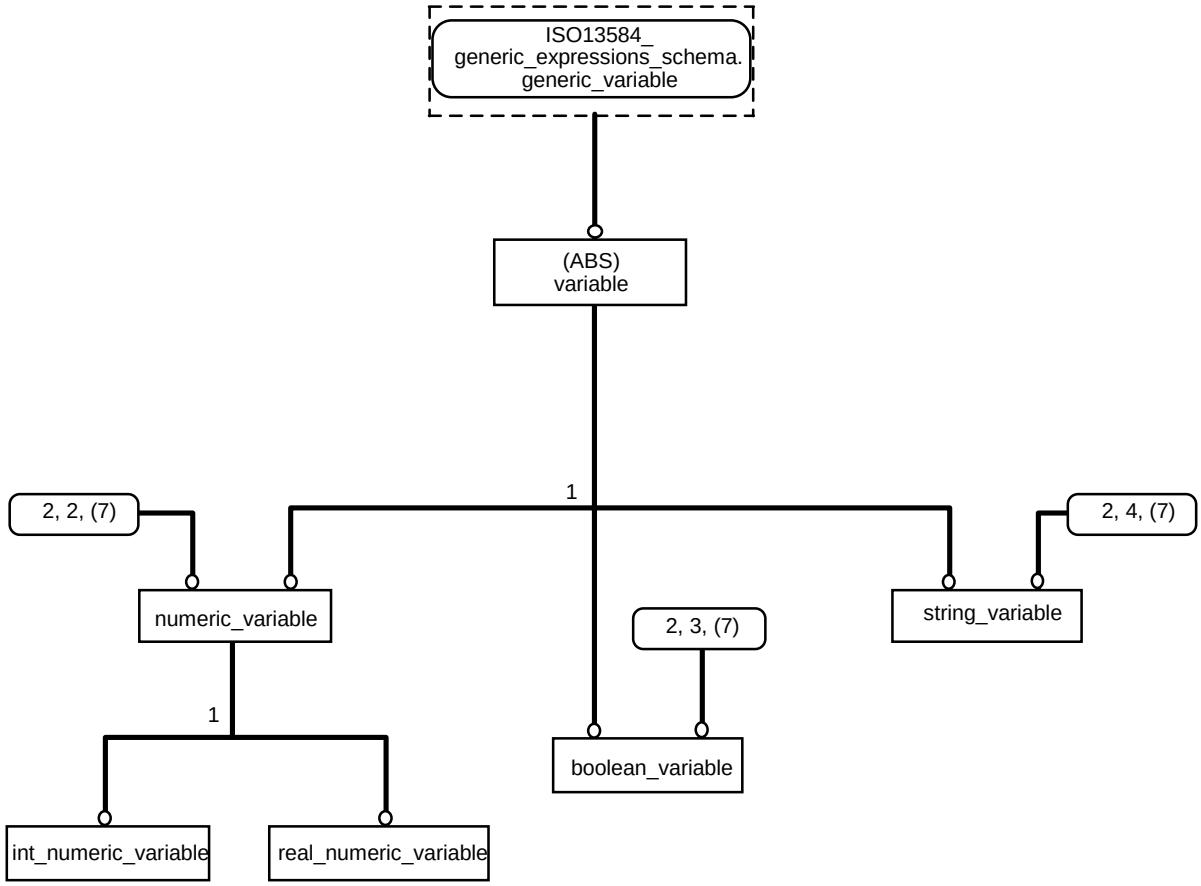

Figure C.3 — ISO13584_expressions_schema - EXPRESS-G diagram 2 of 12- Variables

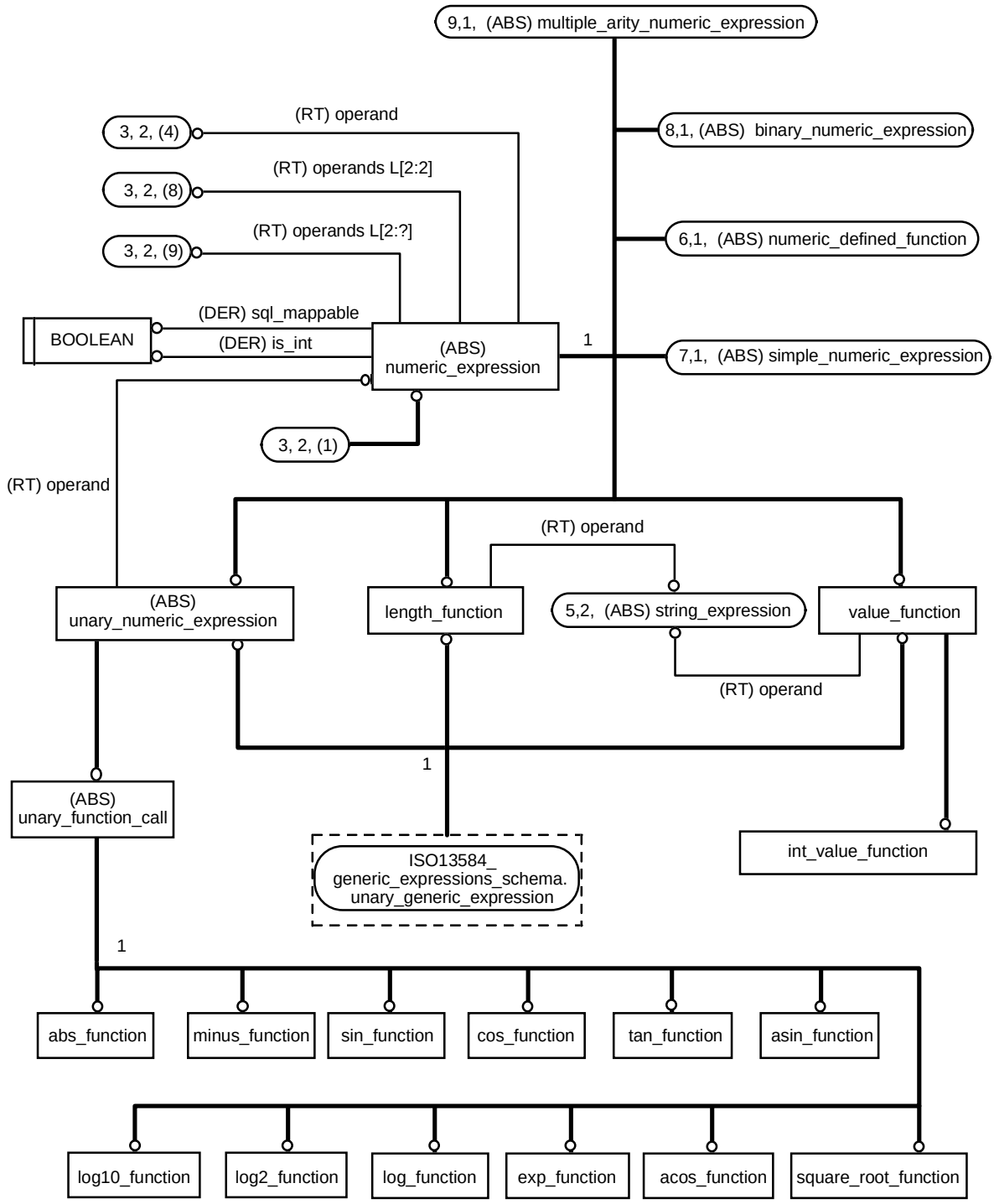

Figure C.4 — ISO13584_expressions_schema - EXPRESS-G diagram 3 of 12- Numeric expressions

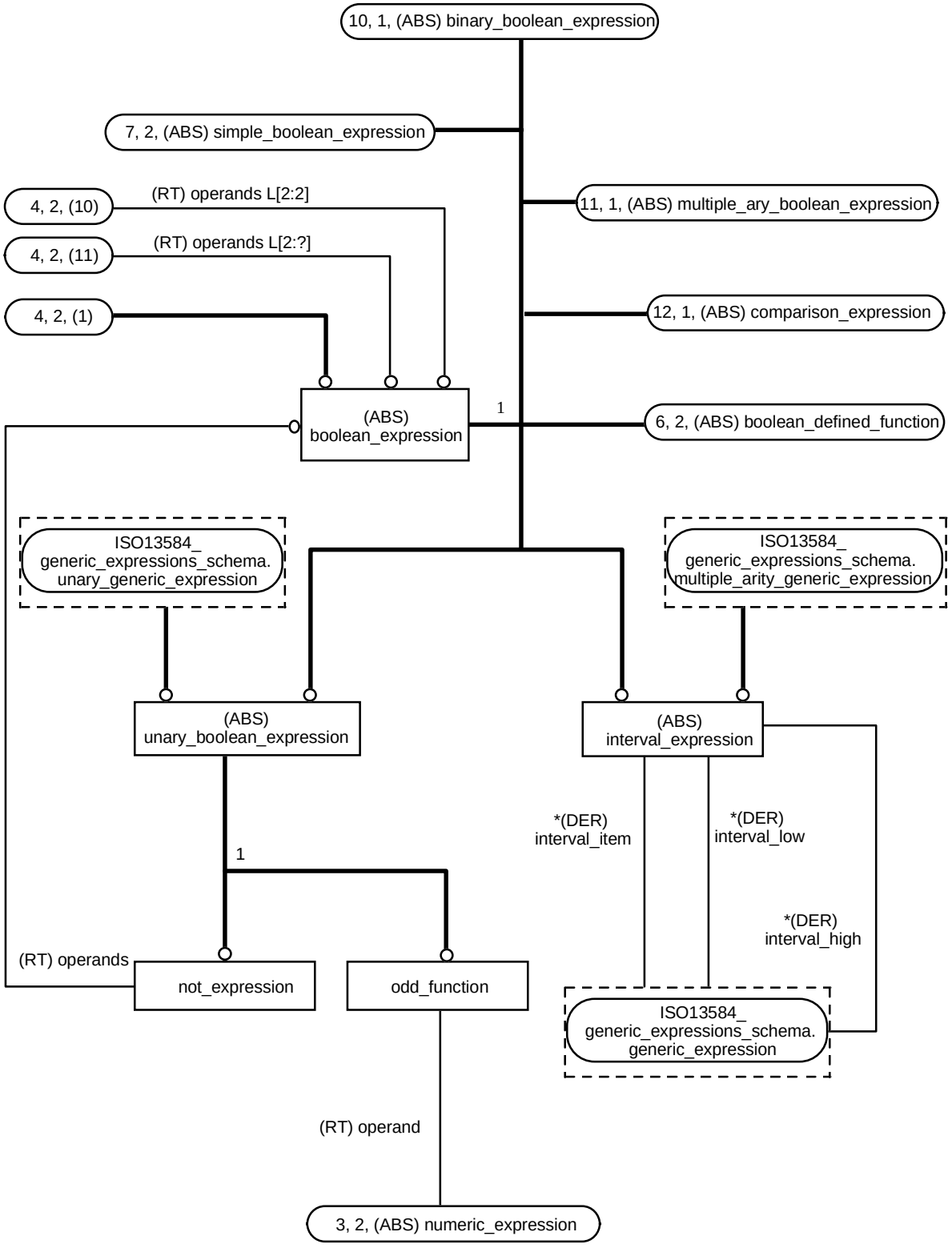

Figure C.5 — ISO13584_expressions_schema - EXPRESS-G diagram 4 of 12- Boolean expressions

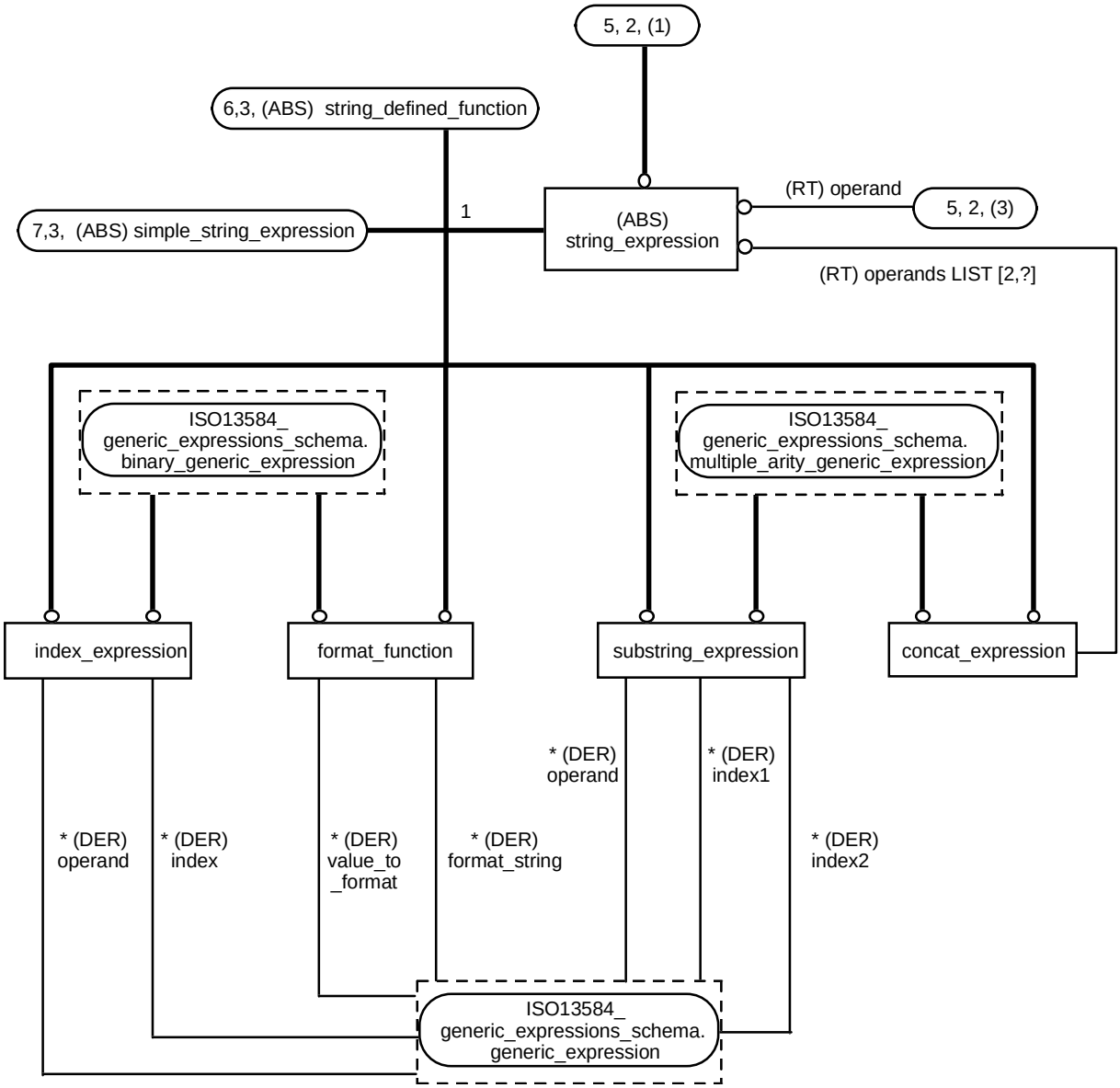

Figure C.6 — ISO13584_expressions_schema - EXPRESS-G diagram 5 of 12- String expressions

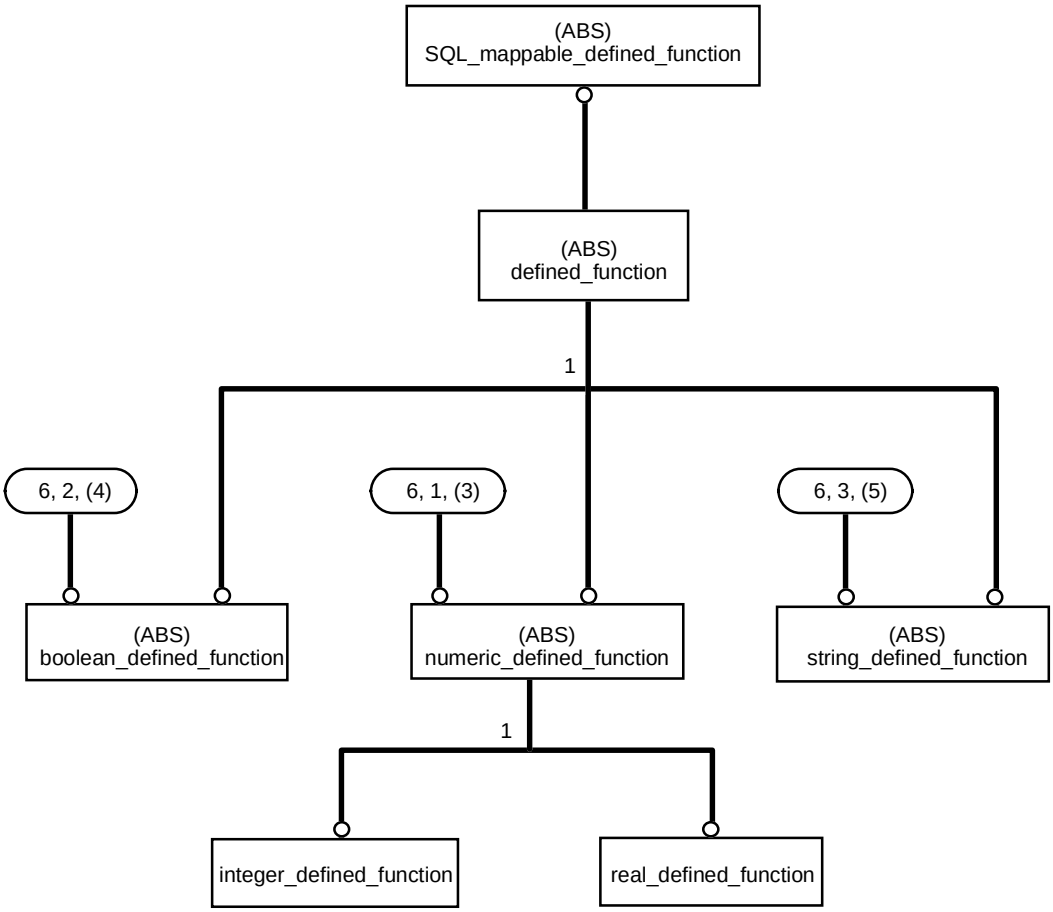

Figure C.7 — ISO13584_expressions_schema - EXPRESS-G diagram 6 of 12 - defined_function

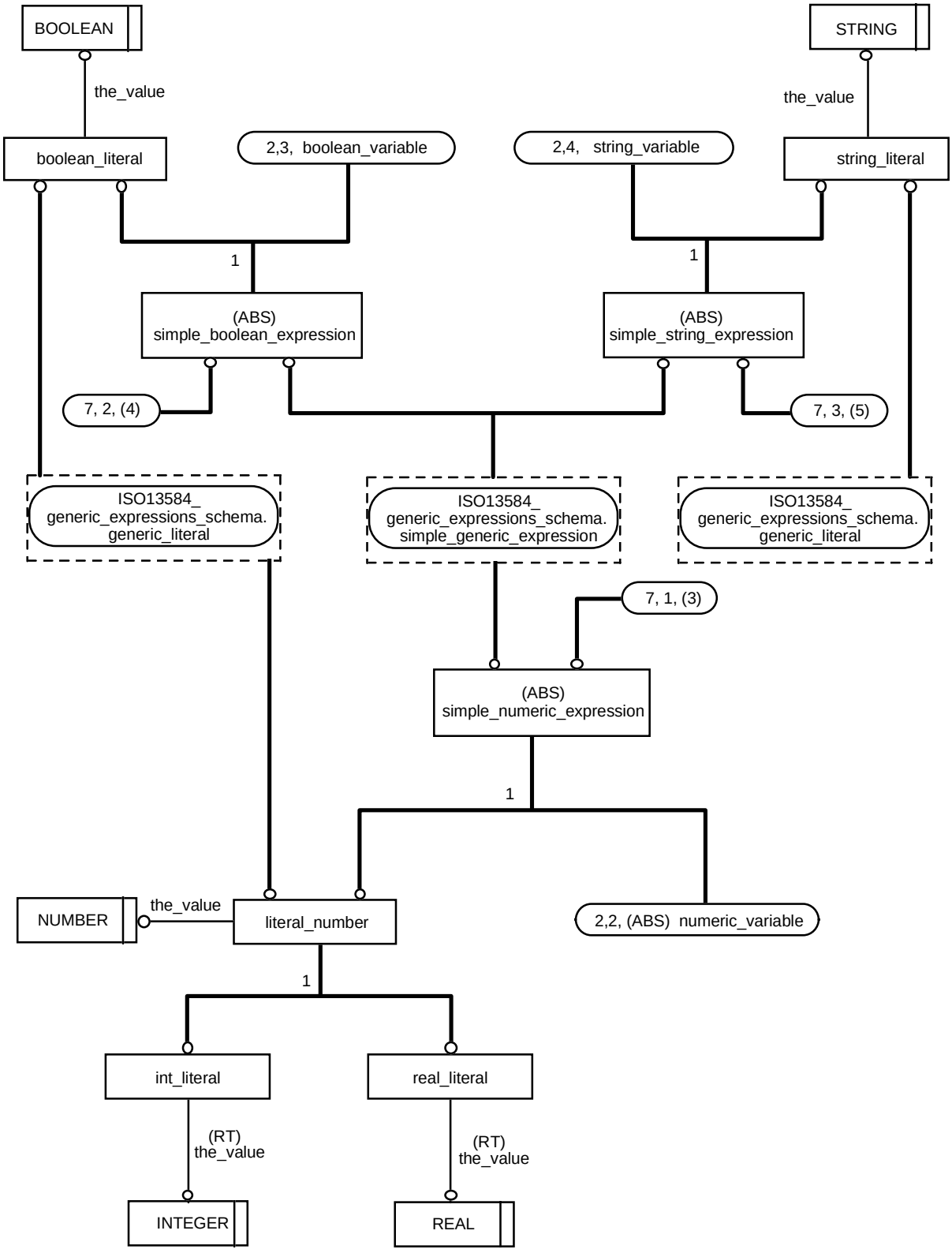

Figure C.8 - ISO13584_expressions_schema - EXPRESS-G diagram 7 of 12- Subtypes of simple_generic_expression

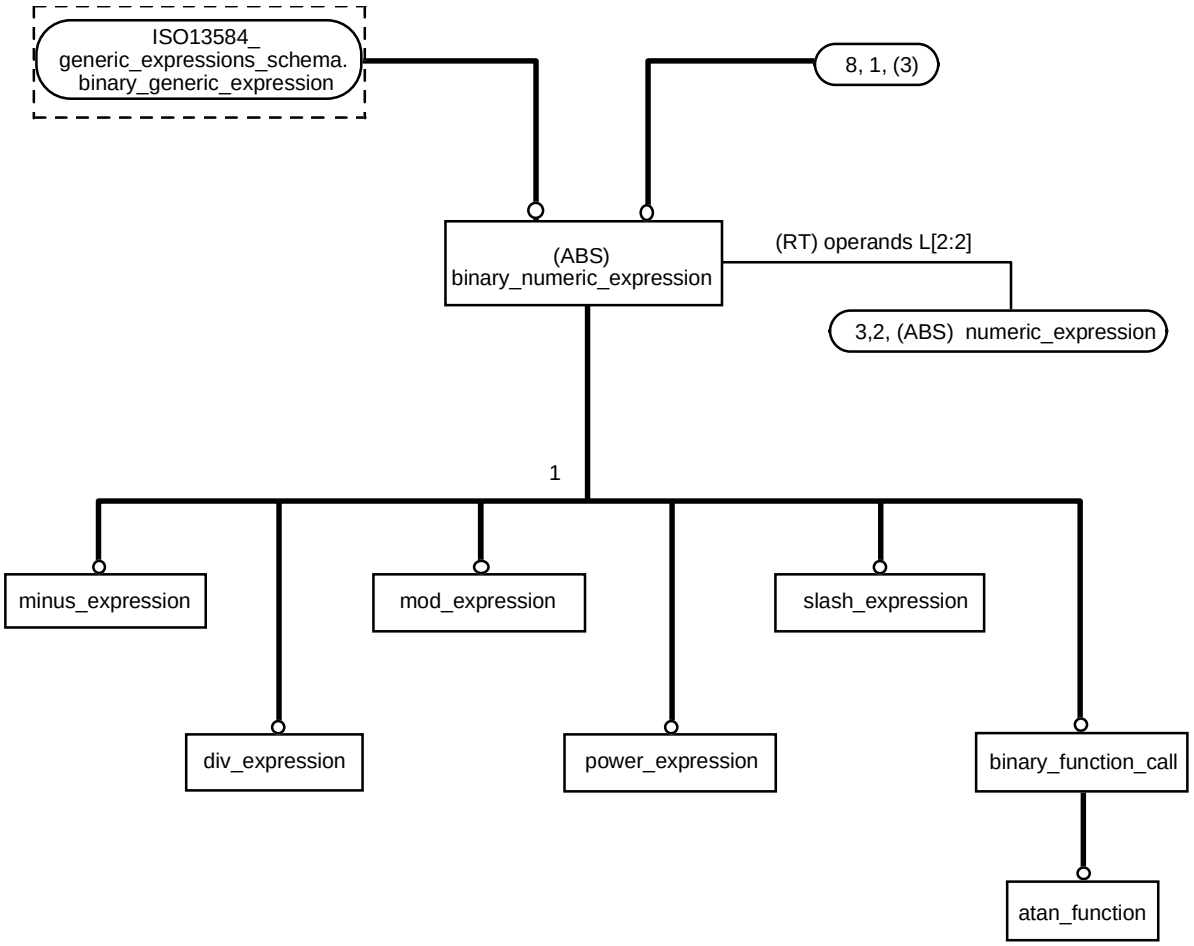

Figure C.9 — ISO13584_expressions_schema - EXPRESS-G diagram 8 of 12- Binary numeric expressions

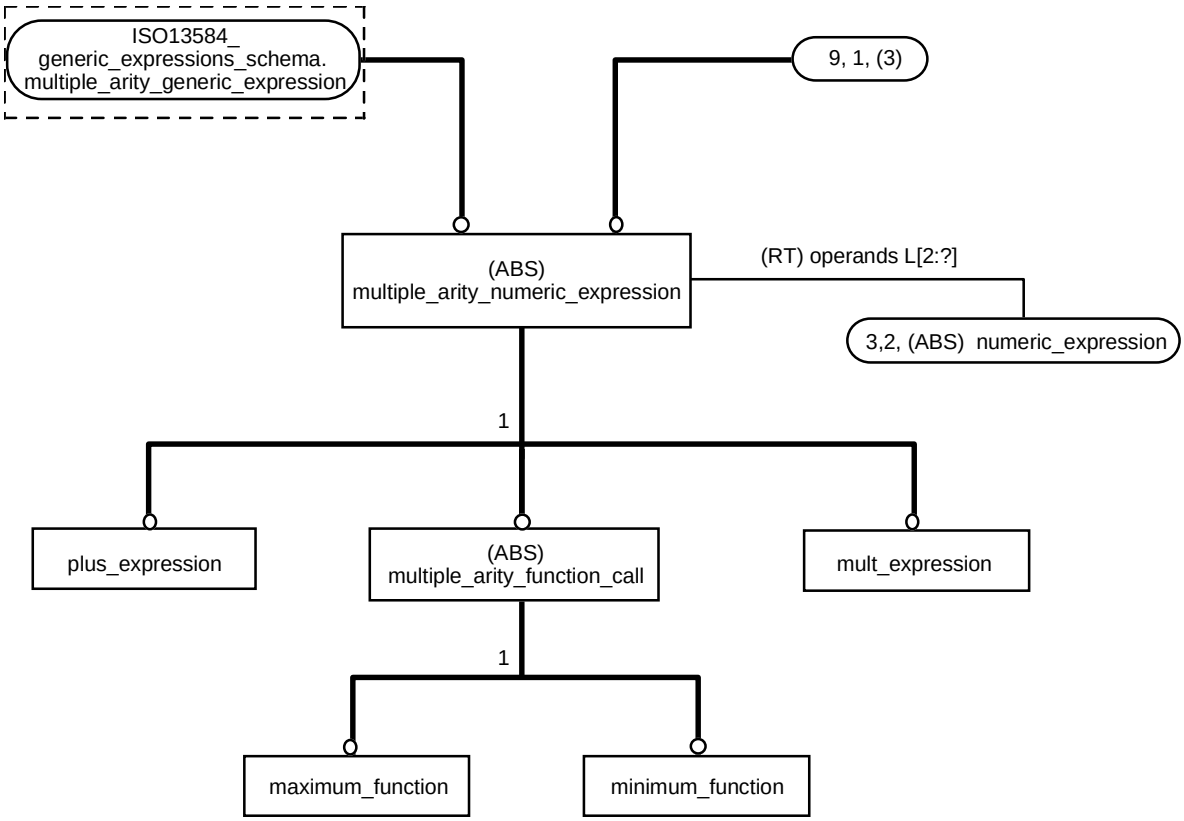

Figure C.10 - ISO13584_expressions_schema - EXPRESS-G diagram 9 of 12- multiple-arity

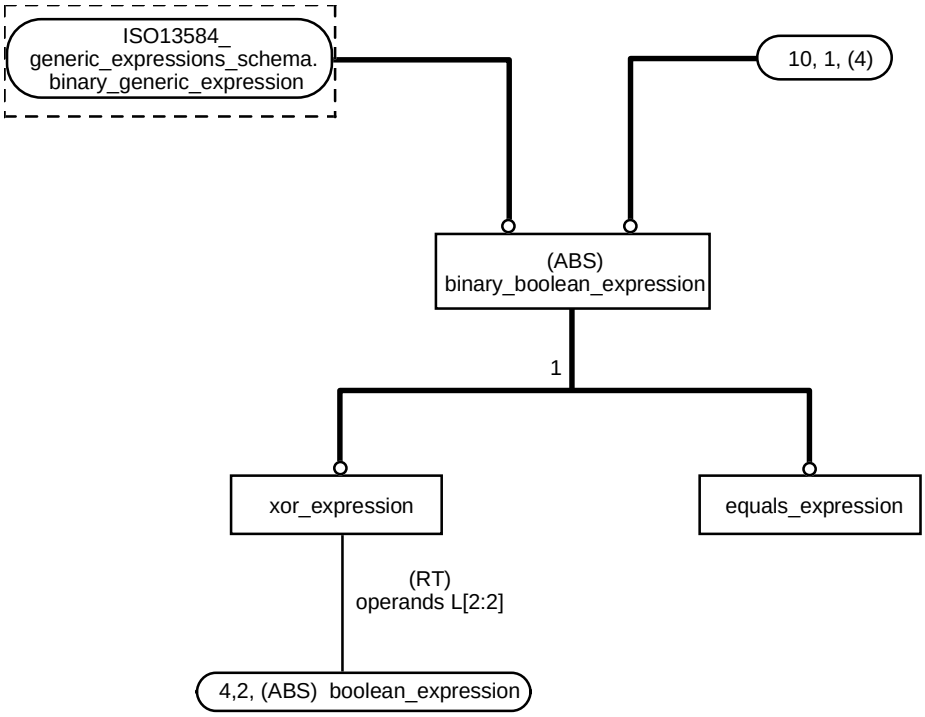

Figure C.11 — ISO13584_expressions_schema - EXPRESS-G diagram 10 of 12- Binary Boolean expressions

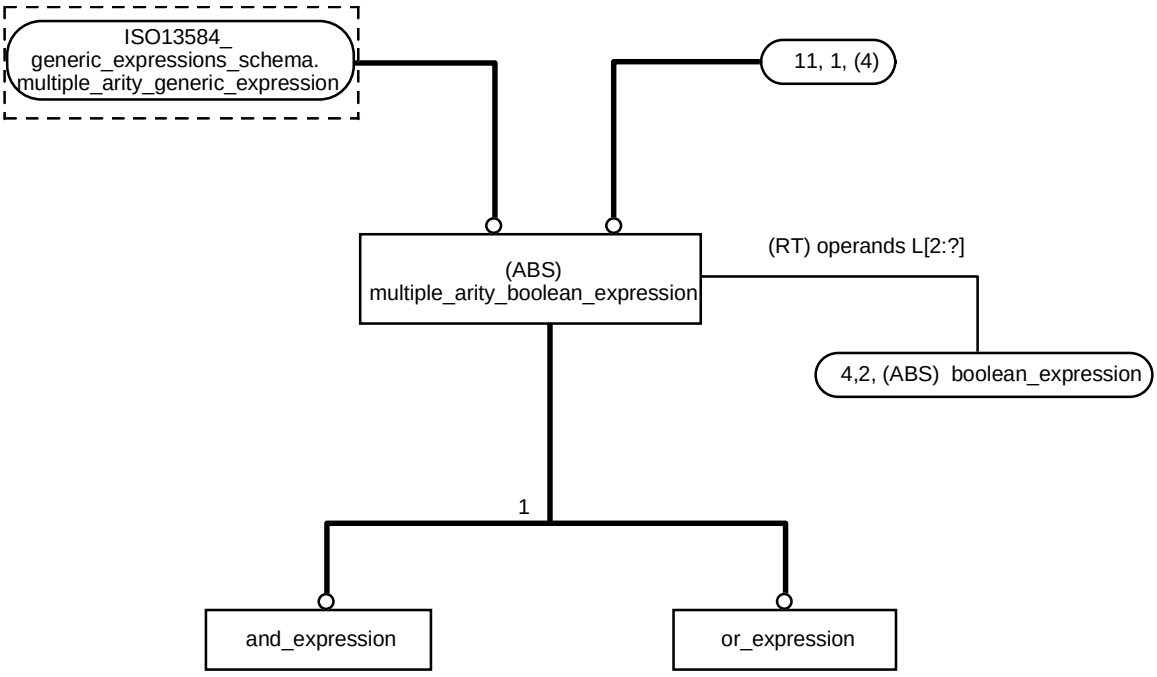

Figure C.12 — ISO13584_expressions_schema - EXPRESS-G diagram 11 of 12- multiple-arity Boolean expressions

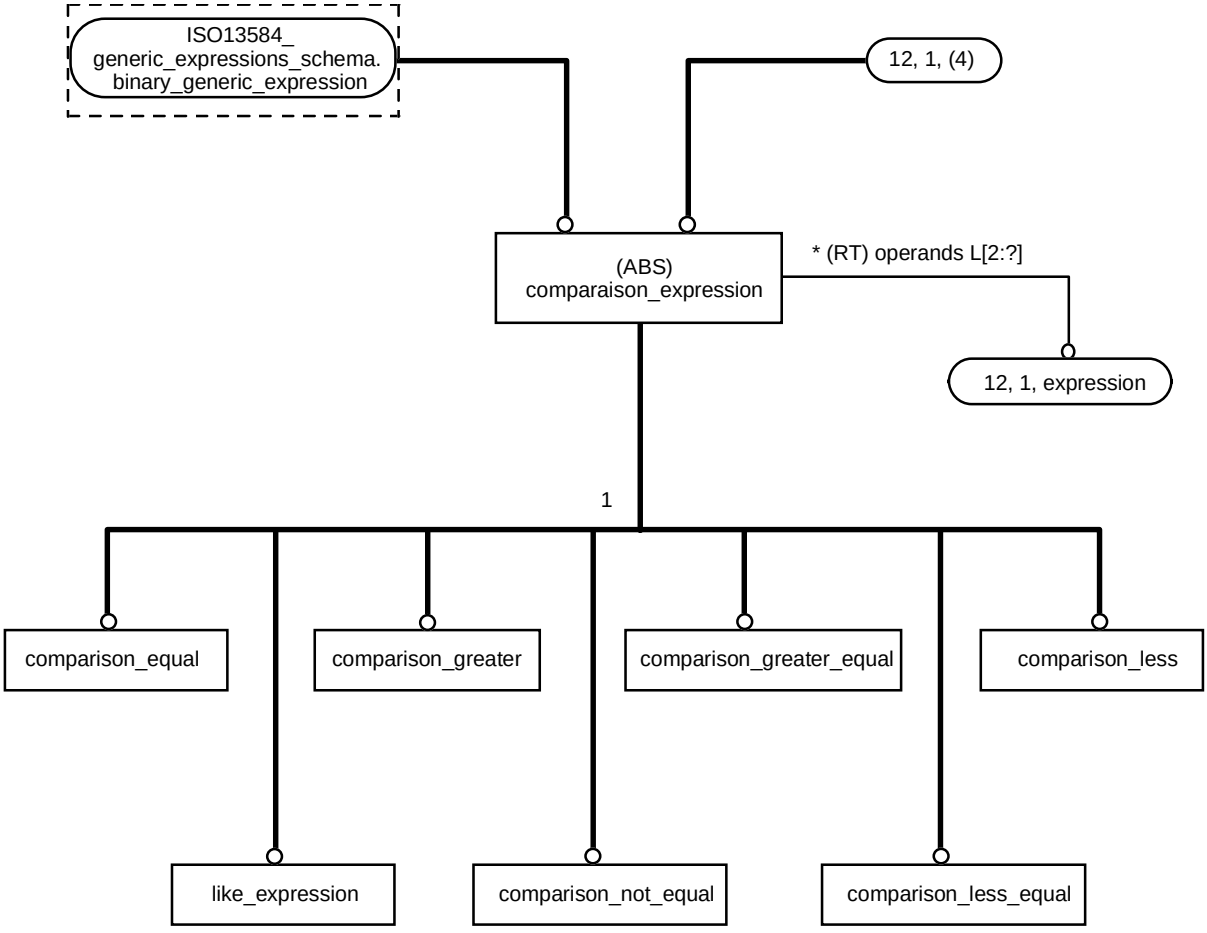

Figure C.13 — ISO13584_expressions_schema - EXPRESS-G diagram 12 of 12- Comparison expressions

Annex D

(informative)

Use of the ISO13584_expressions_schema

D.1 Introduction

The schemata presented in this part of ISO 13584 are generic resources intended to be used wherever expressions should be modelled in the EXPRESS language.

This annex discusses the different possible usage's and applications of these generic resources.

The **ISO13584_expressions_schema** encodes the expressions that can be built from the combination of the simple expression operators (taken from the EXPRESS language), of constants and of variables.

For each of these elements (operators, constants and variables), a semantics must be given each time an expression is used.

The semantics of the operators are specified in conformance with the one defined for each operator in the EXPRESS language reference manual.

When using the expression schema, the only semantics that shall be defined is the semantics of variable, i.e., the interpretation function that bounds values with variables.

D.2 Interpretation function and variable semantics

Note that the semantics of operators and constants are already defined in this part of ISO 13584.

Constants have a simple semantics since they are interpreted by the value they represent.

In expressions, variables stand for concepts that are associated with values. Expressions using these variables stand for relationships between these current values.

When using the **ISO13584_expressions_schema**, the following shall be defined for each variable:

What is the concept a variable stands for;

In what context this concept may be used, and

What is the mechanism (interpretation function) that bounds a value with the variable.

In this part of ISO 13584, the concept a variable stands for (its semantics) is represented by a **variable_semantics** entity. Therefore, the three above points shall be defined by the definition of a subtype of this abstract entity data type.

D.3 Representation of the interpretation function in ISO 13584 Part 20

As stated above, variables shall have an interpretation function in order to be given a semantics. In ISO 13584, this goal is achieved by combining three entities : **variable**, **environment**, and **variable_semantics**. The **variable** entity defines the syntactic representation of the semantic concept

which is represented by the **variable_semantics** entity. The **environment** entity links these two entities in order to exactly associate one semantic to a given variable.

The following planning model recalls how these three entities are defined.

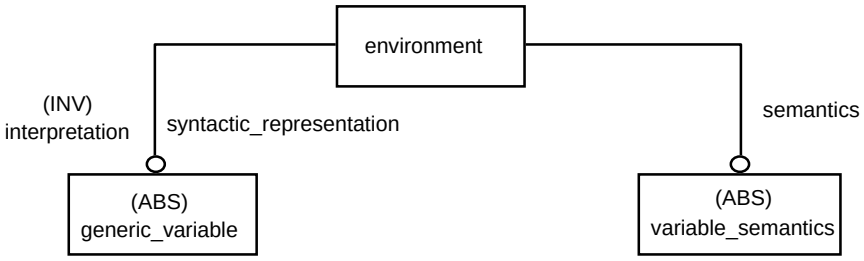

Figure D.1 — Syntax and semantics association for variables

D.4 Use of the variable_semantics entity to define the semantic of new variables

We assume that the concept that is intended to be represented by a variable already exists as an entity within the data model. Therefore, at the data model level, there exist two approaches to bind this concept with a variable :

— the first one uses a particular subtype of the **variable_semantics** entity that references this entity by means of attribute(s);

— the second one defines a common subtype of the **variable_semantics** entity and of this entity data type.

Note that this information model defines only the first of the above two points (See D.2.). The other point shall be specified in the documentation of this model.

We illustrate these two approaches on two examples.

D.4.1 Use of a particular subtype of the variable_semantics entity

This approach for semantic definition consists in introducing a subtype of the **variable_semantics** that references, through attribute(s), the concept the variable stands for.

The following examples show how this approach works. We did not choose toy examples in order to show the feasibility of this approach in practice.

EXAMPLE Definition of a SELF variable.

Remember that a SELF variable is this particular variable occurring in object oriented approaches. It is used within the context of an instance of a class to represent this particular instance and, possibly, the value of one particular attribute of this particular instance.

In ISO 13584 Part 24, that follows the object oriented approach to model families of parts, the representation of SELF variables proved necessary to allow expressions involving this kind of variables. The following EXPRESS_G diagram shows the technique used to represent this semantics using a simple inheritance of the **variable_semantics** entity.

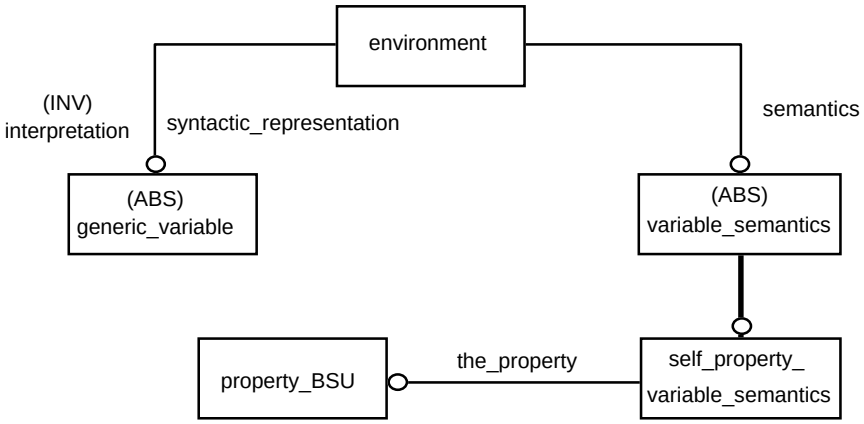

Figure D.2 — Specialisation of the semantics by subtyping of the variable_semantics entity

NOTE **Property_BSU** is defined in ISO 13584-42. It enables the identification of one particular property.

A **self_property_variable_semantics entity** is defined as a subtype of the **variable_semantics** entity. It has an attribute called *the_property* that identifies a particular property of the embedding class (represented by a **property_BSU** entity). So, this entity carries the semantics of the *the_property* of the SELF instance.

Note that the documentation of the **self_property_variable_semantics** specifies that:

— the corresponding variable shall be used in the context of a PLib class instance, and

— this variable stands for the value of the relevant property of this class instance.

D.4.2 Multiple inheritance of the variable_semantics entity and of another entity

Multiple inheritance allows the gathering, in a common entity, of the features of different super-types.

Based on this concept of sharing, one can define a semantics of a variable as the sharing of the **variable_semantics** entity (to say that we are dealing with the semantics of a variable), and any other entity we want to represent by a variable.

EXAMPLE Definition of a **variable_semantics** associated to a **material_property_representation** described in ISO 10303-45.

By subtyping the resource constructs from the **material_property_representation_schema** documented in ISO 10303-45, the value of a **material_property** may be associated with a **numeric_expression** that specifies the dependence between the value of this **material_property** and other **property_definitions** referenced by their **material_property_ representation**.

The following planning model shows how this mechanism works.

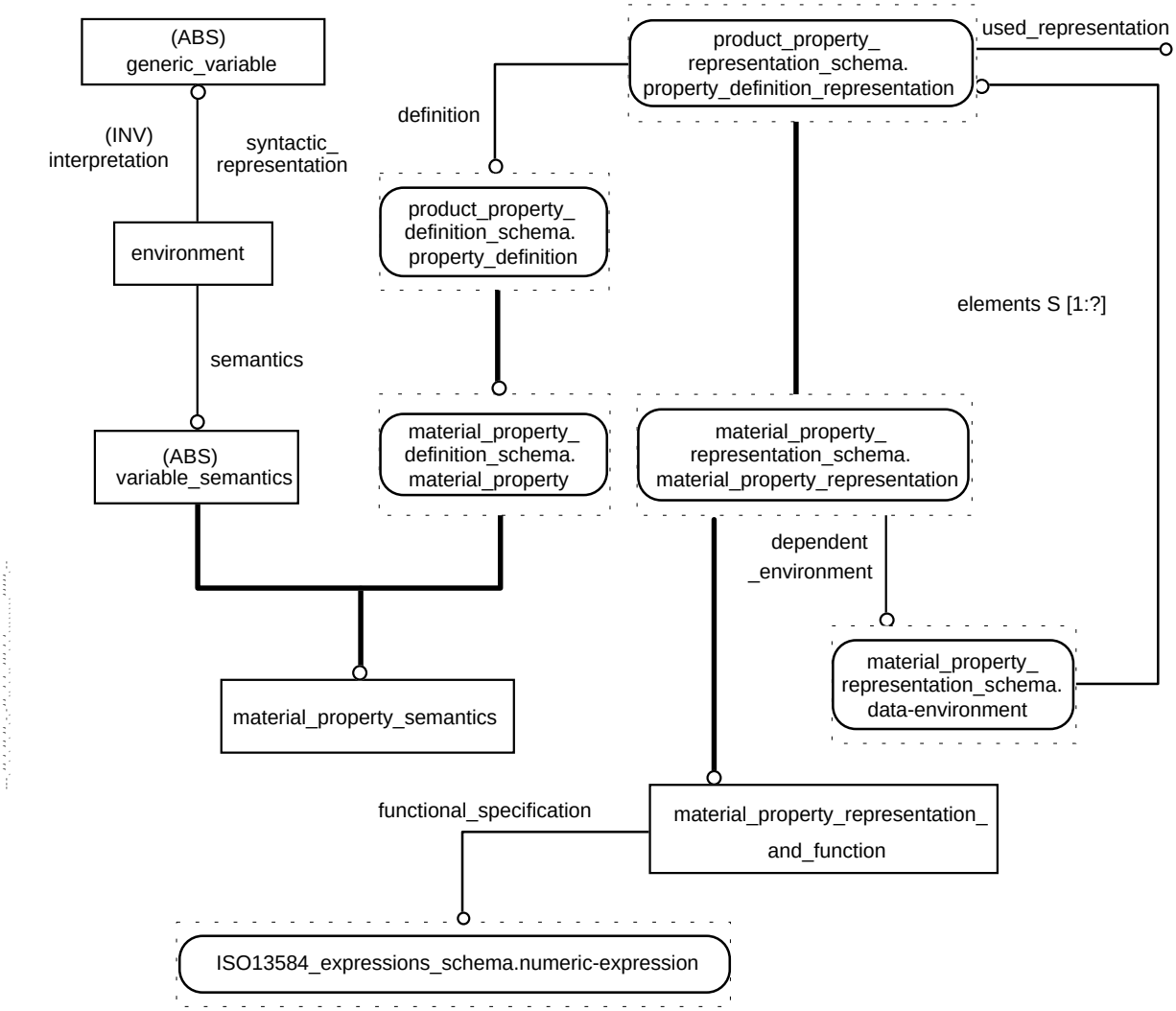

Figure D.3 — Specialisation of the semantics by subtyping the variable_semantics entity and another entity

A WHERE rule in the **material_property_representation_and_function** entity should ensure that the **functional_specification numeric_expression** involves as variables only those variables:

a) that are related through the **environment** entity to a **material_property_semantics**, and

b) that all these **material_property_semantics** correspond to the set of properties that model the environment (i.e., each of these **material_property_semantics** is referenced for some *i*, by **SELF.dependent_environment.elements[i].definition**).

Using the **material_property_representation_and_function** subtype, a **material_property** is associated to both a value (that corresponds to some particular values of the properties on which they depend on), and to a function that defines the functional dependency between these values.

Note that this approach does not require the use of an extra attribute in the **material_property_semantics** entity. The inheritance mechanism propagates all the attribute of the **material_property** entity (i.e., the identification of the property).

Note also that, in this example, the documentation shall describe the context of use of the variable (for one particular product, for any product of some set, ...) and the interpretation function.

D.4.3 Defining a concept not represented in the model

Some variable semantics may correspond to concepts that are not already represented explicitly in a model. In this case, a complete information model of this concept shall be built as a subtype of **variable_semantics**.

The following example shows how might be defined two variables that stand for the *x* and *y* coordinate in some particular 2D space. Such variables may then be involved in any numeric expression that define the exact (mathematical) shape of a particular **curve** entity.

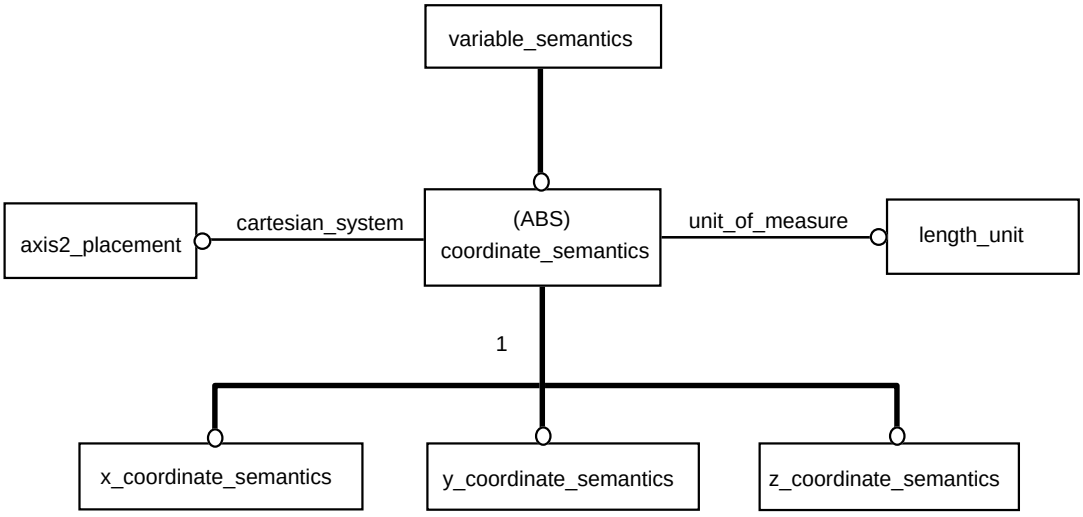

Figure D.4 — Example of the definition of a concept not represented in the model : coordinates

Annex E

(informative)

Specialisation of the schemata

E.1 Introduction

The definition the **ISO13584_generic_expressions_schema** and of the **ISO13584_expressions_schema** are generic resources in the sense that they can be specialised without affecting the already defined resources. Indeed, these schemata have been defined in order to be specialised or extended for the introduction and the modelling of any kind of expression. The goal of this annex is to discuss how specialisation of the two schemata may be performed.

E.2 Specialisation of the ISO13584_generic_expressions_schema

ISO13584_generic_expressions_schema defines the structure of any kind of expression, without introducing any particular data type.

Specialisation of the **ISO13584_generic_expressions_schema** requires the introduction of a particular data type(s). Such data type(s) shall be introduced by subtyping the existing EXPRESS resources and may be referenced by the new operators that manipulate this new introduced data type.

EXAMPLE In **ISO13584_expressions_schema** schema, only *integer, real, Boolean* and *string* concrete types are defined. Therefore, only these types are associated with operators that compute values.

When a new data type is introduced, the following EXPRESS entities are concerned with this specialisation process:

— `generic_expression`
— `generic_variable`
— `variable_semantics`
— `generic_literal`
— `unary_generic_expression`
— `binary_generic_expression`
— `multiple_arity_generic_expression`

E.3 Specialisation of the ISO13584_expressions_schema

Specialisation of the **ISO13584_expressions_schema** enables the enlargement of the set of operators either that combine values that belong to the base data type of this schema (i.e., number, real, integer, Boolean and string) or that produce values that belong to these base data types (e.g., distance between two points).

Indeed, such a specialisation shall ensure the semantic soundness and correctness of the resulting expressions and shall not affect the correctness of the generic resources provided as EXPRESS functions. These resources shall still be compliant with their abstract descriptions (e.g., the i**s_int_expr** shall be able to compute whether or not the data type of an expression is the INTEGER data type).

The following EXPRESS entities were specifically introduced to support such a specialisation process:

— `boolean_defined_function`
— `integer_defined_function`

— numeric_defined_function
— real_defined_function
— string_defined_function
— SQL_mappable_defined_function

E.4 Methodology for specialisation of ISO 13584 part 20

This section suggests the different steps that should be followed in order to define different specialisations of the schemata introduced in this part of ISO 13584.

1) Identify the concrete data type to be processed by the expressions. If only *integer, real, Boolean* and *string* concrete types are to be used, specialise the **ISO13584_expressions_schema**, else specialise the **ISO13584_generic_expression_schema**.

2) Identify the semantic concepts to be carried by the defined expressions. Define the required subtypes of the **variable_semantics** entity (see: Annex D).

3) Specialisation of the **ISO13584_generic_expressions_schema**.

   a) Define the root of all expressions of the new data type by subtyping the **generic_expression** entity (or the **numeric_expression**, **Boolean_expression** or **string_expression** as appropriate).

   b) Define the variables of these expressions by subtyping the **generic_variable**s entity. It is linked by the **environment** entity to the semantics defined in step 2.

   c) Define the constants of these expressions by subtyping the **generic_literal** entity.

   d) Define the operators which combine variables and constants of the new data type. These operators are obtained by subtyping the **unary_generic_expression, binary_generic_expression**, and the **multiple_arity_generic_expression** entities according to their arity. Add the rules that ensure the semantic soundness and correctness of these operator.

1) Integration with the **ISO13584_expressions_schema** .

   d) If any operator that produces values of the new data type from **expression**s (e.g., a complex defined by two reals) exists, then

      — Reference the **ISO13584_expressions_schema**

      — Define this operator as a common data type of the root entity defined in step 3.1, and of one of the following entities : **unary_generic_expression, binary_generic_expression**, and the **multiple_arity_generic_expression** according to the arity of this operator. Add the rules that ensure the semantic correctness of this operator.

   b) If there exist any operator on the new data type that generates values of one of the base data type of the **ISO13584_expressions_schema** then for each operator or function returning the *integer, real, Boolean* or *string* types, subtype the corresponding **defined_function** entity to define the corresponding function and one of the entities **Unary_generic_expression, binary_generic_expression**, and the **multiple_arity_generic_expression** according to the arity of the operator.

This process ensures the following:

— any expression conforming to the specialised schema is semantically correct, and

— all the functions provided as resources by this part ISO 13584 (e.g., **is_int_expr**, **used_variables**, **is_SQL_mappable**, ...) comply with their abstract description.

NOTE In case of specialisation of the **ISO13584_expressions_schema** schema, the step 3 is skipped.

E.5 Example of specialisation of the ISO13584_generic_expressions_schema schema

Notice that in this example, we have voluntarily kept the EXPRESS notations as simple as possible in order to make the specialisation understandable.

Let us consider the specialisation of the **ISO13584_generic_expressions schema** to handle expressions dealing with complex numbers. These expressions are obtained by subtyping of the **ISO13584 _generic_expressions_schema.**

The following EXPRESS specifications give the kernel of such a schema. We will deal only with few expressions.

— STEP 1: the defined data type is complex.

— STEP 2: the semantic concept to be defined by complex expressions. Assume that we would like to associate these expressions with the coordinate pair of 2D **cartesian_point**s (as defined in ISO 10303-42).

One can define the following semantics intended to be associated with complex variables.

EXPRESS specification :

```
*)
ENTITY position_variable_semantics
SUBTYPE OF (variable_semantics, cartesian_point);
WHERE
     SELF\geometric_representation_item.dimension=2;
END_ENTITY ;
(*
```

The semantics shall be a 2D point and the documentation shall state that the real part is *x* coordinate and that the imaginary part is the *y* coordinate.

Note that we assume, in this example, that we do not need **point_expression**s.

— STEP 3 : definition of the **complex_expression** by specialisation of the **ISO13584_generic_expression_schema**

EXPRESS specification :

```
*)
ENTITY complex_expression
SUBTYPE OF (generic_expression)
ASTRACT_SUPERTYPE OF (ONEOF(      simple_complex_expression,
                                  unary_complex_expression,
                                  binary_complex_expression
                                  multiple_arity_complex_expression)) ;
END_ENTITY ;
```

```
ENTITY simple_complex_expression
ABSTRACT SUPERTYPE(OF ONEOF(     complex_variable,
                                          complex_literal))
SUBTYPE OF (complex_expression, simple_generic_expression) ;
END_ENTITY ;

ENTITY complex_variable ;
SUBTYPE OF (simple_complex_expression, generic_variable) ;
END_ENTITY ;

ENTITY complex_literal
SUBTYPE OF (simple_complex_expression, generic_literal) ;
 real_part, imag_part : REAL ;
END_ENTITY ;

ENTITY unary_complex_expression
.......
END_ENTITY ;

ENTITY binary_complex_expression
.....
END_ENTITY ;

ENTITY multiple_arity_complex_expression
......
END_ENTITY ;

ENTITY position_cartesian_constructor
SUBTYPE OF (complex_expression, binary_generic_expression);
 x, y : REAL
END_ENTITY;

ENTITY position_polar_constructor
SUBTYPE OF (complex_expression, binary_generic_expression);
 ro, theta : REAL;
END_ENTITY;
(*
```

— STEP4: the specialisation of the **ISO13584_expressions_schema** is given by defining the following entities.

EXPRESS specification :

```
*)
ENTITY real_part_defined_function
SUBTYPE OF (real_defined_function, unary_generic_expression);
 the_complex : complex_expression;
END_ENTITY ;

ENTITY imaginary_part_defined_function
SUBTYPE OF (real_defined_function, unary_generic_expression)
 the_complex : complex_expression;
```

```
END_ENTITY ;


ENTITY angle_defined_function
SUBTYPE OF (real_defined_function, unary_generic_expression);
END_ENTITY ;

ENTITY module_defined_function
SUBTYPE OF (real_defined_function, unary_generic_expression) ;
END_ENTITY ;
(*
```

Note that, this schema is voluntarily left incomplete. It is not the purpose of this annex to provide a standardised schema for complex expressions.

E.6 Example of specialisation of the ISO13584_expressions_schema schema

This example corresponds to the case where STEP 3 is skipped.

One wants to introduce an operator that computes the distance between two points in a cartesian space. This function returns a real number. The entity corresponding to this function is obtained by specialisation. The **real_defined_function** is subtyped. The following entity is defined :

EXPRESS specification :

```
*)
ENTITY Distance_function
SUBTYPE OF(real_defined_function) ;
 pt1, pt2 : point ;
END_ENTITY ;
(*
```

Annex F

(informative)

Static analysis of expressions

F.1 Introduction

In the context of this part of ISO 13584, expressions are considered. Two possible views can be defined according to the defined concepts and assumptions made in clause 4 : exchange time and evaluation time.

This part of ISO 13584 deals with the exchange of expressions only. Only static properties of expressions can be inferred at this stage. It does not model either the evaluation function nor how this evaluation is performed.

This annex reviews the analysis that may be performed on expressions at exchange time, and gives some explanations about these processes.

At the generic expressions level, computations of the used variables and functions are possible. Moreover, the acyclicity of the graph representing an expression may be checked as well.

At the expressions level, it is possible to compute the data type of an expression.

F.2 is_acyclic function

Remember that:

a directed graph is a collection of nodes connected together by some directed links.

a directed graph is said to be acyclic if there is no path from a node back to itself.

As stated in the concepts and assumptions of this part of ISO 13584, the underlying structure of a graph is a directed acyclic graph. This structure is mandatory for the traversal of an expression. Indeed, it ensures that all the functions which perform a traversal of such a graph terminates when it reaches the node from which there is no link arriving to this node.

The **is_acyclic** function checks whether an expression satisfies this condition.

Notice that a tree is a particular directed cyclic graph.

F.3 Used_variables and used_functions functions

The **used_variables**, respectively the **used_functions**, functions are the functions which returns all the entities representing a variable, respectively a function, in a given expression.

These functions, make a traversal of the directed acyclic graph that represents the expression, as for evaluation and data type synthesis, and they return the variables and the functions occurring in this expression.

F.4 Is_SQL_mappable function

The **Is_SQL_mappable** function checks if a given expression is mappable to the SQL language for querying databases.

This function, makes a traversal of the directed acyclic graph that represents the expression, as for evaluation and data type synthesis, and returns TRUE when the expression is mappable.

In the context of ISO 13584-24, this function has been used to process data structures related to tables and to relational databases. It has been used to check the mappability of several expressions occurring in constraints.

F.5 Type control and type synthesis

Type control is the process that allows the checking of the correct typing of an expression.

Type synthesis is the process that allows the synthesis of the data type of an expression.

Note that in the **ISO13584_expressions_schema** defined in this part of ISO 13584, type control is ensured by the constraints of this schema. Type synthesis results from the application of the TYPEOF EXPRESS function on the entity which carries the data type.

Index

ICS 25.040.40

Descriptors: automation, automation engineering, computer applications, industrial products, components, data, sets of data, data representation, data exchange, libraries.

Price based on 87 pages