INTERNATIONAL STANDARD

ISO 10303-108

First edition
2005-02-01

Industrial automation systems and integration — Product data representation and exchange —

Part 108: Integrated application resource: Parameterization and constraints for explicit geometric product models

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits —

Partie 108: Ressources d'application intégrées: Paramétrage et contraintes pour les modèles de produits géométriques explicites

Reference number
ISO 10303-108:2005(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Figures

Tables

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 organizations, 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 standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this part of ISO 10303 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 10303–108 was prepared by Technical Committee ISO/TC184, *Industrial automation systems and integration*, Subcommittee SC4, *Industrial data.*

ISO 10303 consists of a series of parts, under the general title *Industrial automation systems and integration — Product data representation and exchange.* The structure of ISO 10303 is described in ISO 10303-1.

Each part of ISO 10303 is a member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the integrated application resources series. The integrated generic resources and the integrated application resources specify a single conceptual product data model.

A complete list of parts of ISO 10303 is available from the Internet:

<http://www.tc184-sc4.org/titles/>

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for archiving.

This part of ISO 10303 is a member of the integrated resources series. Major subdivisions of this part of ISO 10303 are:

— Parameterization schema;

— Explicit constraint schema;

— Variational representation schema;

— Explicit geometric constraint schema;

— Sketch schema.

This part of ISO 10303 provides representations of parameters and constraint relationships for use in models exchanged using ISO 10303, and specifies mechanisms for the association of such elements with other elements of the models in which they apply. Parameters and constraints are used in CAD systems to indicate, respectively, variant and invariant characteristics of a model under editing operations. This part of ISO 10303 also specifies representations for *sketches*, two-dimensional geometric configurations, possibly including parameters and constraints, that are often used by CAD systems as basic elements in constructional operations.

Models with explicitly represented parameterization and constraints are described as *variational*. The schemas provided are intended in the first instance to supplement the ISO 10303 integrated generic resources to allow the representation of product shape models enhanced with variational information. However, the basic mechanisms provided in the first three schemas can be used for representing parameterization and constraints in the context of any ISO 10303 model, whether of a product, a plan, a process or an organization.

Other resource parts of ISO 10303 provide capabilities for the representation and exchange of models having no associated variational information. That information, if present in the originating system, affects the behaviour of a model under editing operations there; it forms an important part of what is sometimes referred to as *design intent*. The use of this part of ISO 10303 for its capture and transfer potentially allows the reconstruction of key elements of design intent in a receiving system. The transferred variational information asserts relationships that already exist in the exchanged model; its purpose is to initiate processes in the receiving system that will ensure that those relationships are maintained if the model is modified there. This will assist in the efficient modification of the exchanged model in the receiving system, through the use of the original designer's scheme of parameterization and constraints. Such modification of models transferred using ISO 10303 has proved to be difficult or impossible in the absence of design intent information.

EXAMPLE It may be desired to optimize the model in the receiving system for structural integrity, or to modify it to make it cheaper to manufacture.

This part of ISO 10303 is intended to interoperate with other closely related parts, notably ISO 10303-55, which defines procedural and hybrid representations, specified in terms of the constructional operations

used in building a model. The primary forms of shape representation used by many modern CAD systems are of these types. ISO 10303-108 (this document) and ISO 10303-55 between them provide for the capture of the two major aspects of design intent. Procedural representations are outside the normative scope of the present document, but annex F provides some discussion of the role of parameterization and constraints in procedural and hybrid representations, and of the interplay between the explicit and procedural approaches to shape modelling.

A further aspect of modern CAD systems is their provision of feature-based design methods. This part of ISO 10303 does not address the topic of features, though it provides essential resources for the positioning and orientation of features in CAD models.

Three books providing further background on the topics covered by this part of ISO 10303 are given in the Bibliography [3,4,6].

The contents of the schemas making up this part of ISO 10303 are as follows:

parameterization_schema: Mechanisms for associating parameters with variable quantities in an instantiated EXPRESS model;

explicit_constraint_schema: Definitions of generic constraint relationships between elements of an instantiated EXPRESS model;

variational_representation_schema: Specification of the relationship between parameter and constraint information and the non-variational model with which it is associated;

explicit_geometric_constraint_schema: Specialization of the **explicit_constraint_schema** for the representation of commonly used geometric constraints (such as parallelism or tangency) between explicitly represented elements of geometric models;

sketch_schema: Means for the representation of two-dimensional sketches, which may contain variational elements, for use as basic elements in shape modelling operations.

The relationships of the schemas in this part of ISO 10303 to other schemas that define the integrated resources of ISO 10303 are illustrated in Figure 1 using the EXPRESS-G notation. EXPRESS-G is defined in annex D of ISO 10303-11. The internal relationships among ISO 10303-108 schemas are shown in Figure 2. The schemas occurring in Figure 1 are (with two exceptions that form part of ISO 13584) components of ISO 10303 integrated resources, and they are specified in the following resource parts:

`measure_schema`	ISO 10303-41
`product_property_representation_schema`	ISO 10303-41
`support_resource_schema`	ISO 10303-41
`geometric_model_schema`	ISO 10303-42
`geometry_schema`	ISO 10303-42
`topology_schema`	ISO 10303-42
`representation_schema`	ISO 10303-43
`mathematical_functions_schema`	ISO 10303-50
`ISO13584_generic_expressions_schema`	ISO 13584-20
`ISO13584_expressions_schema`	ISO 13584-20

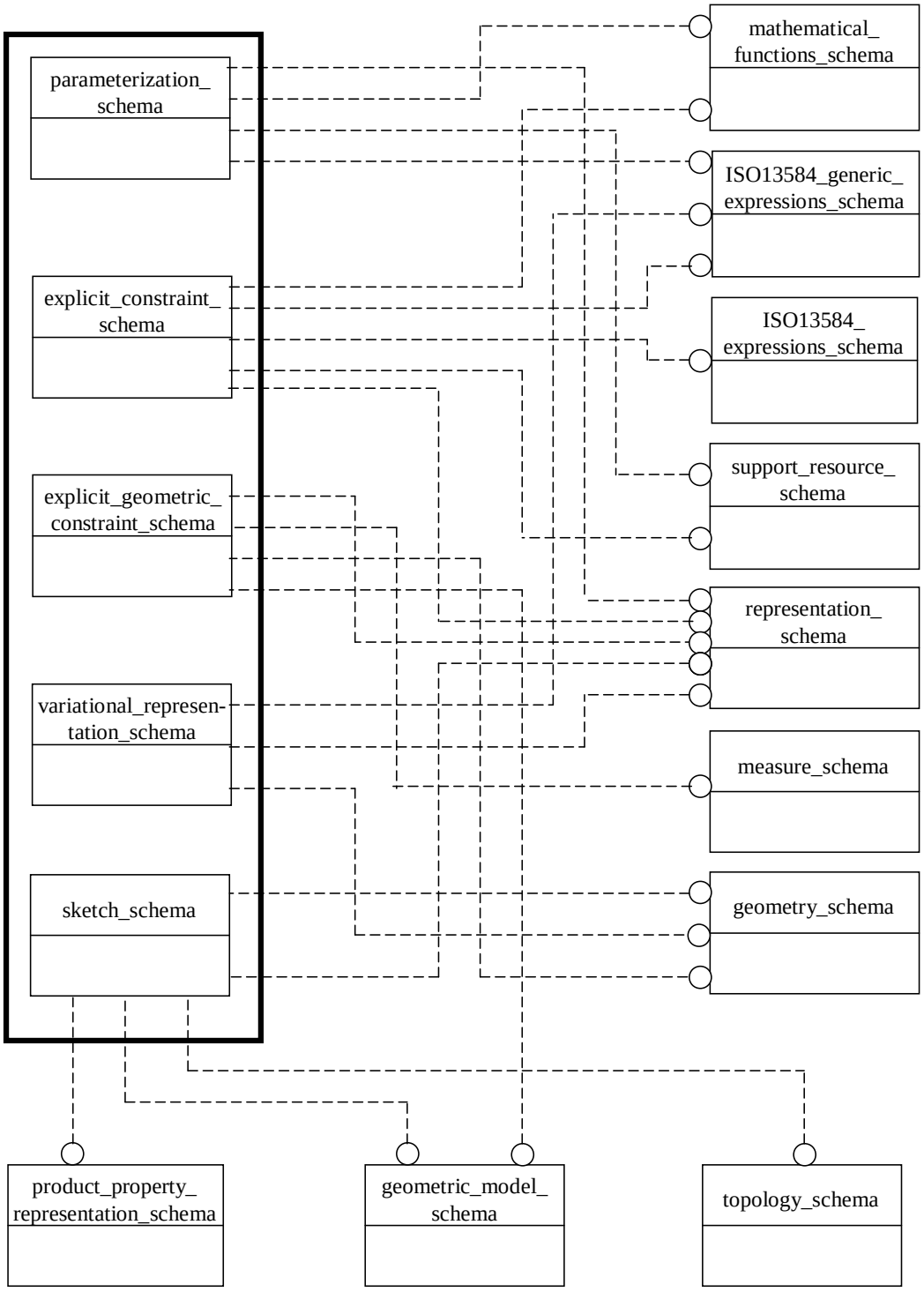

Figure 1 – Schema level diagram of relationships between ISO 10303-108 schemas (inside the box) and other resource schemas

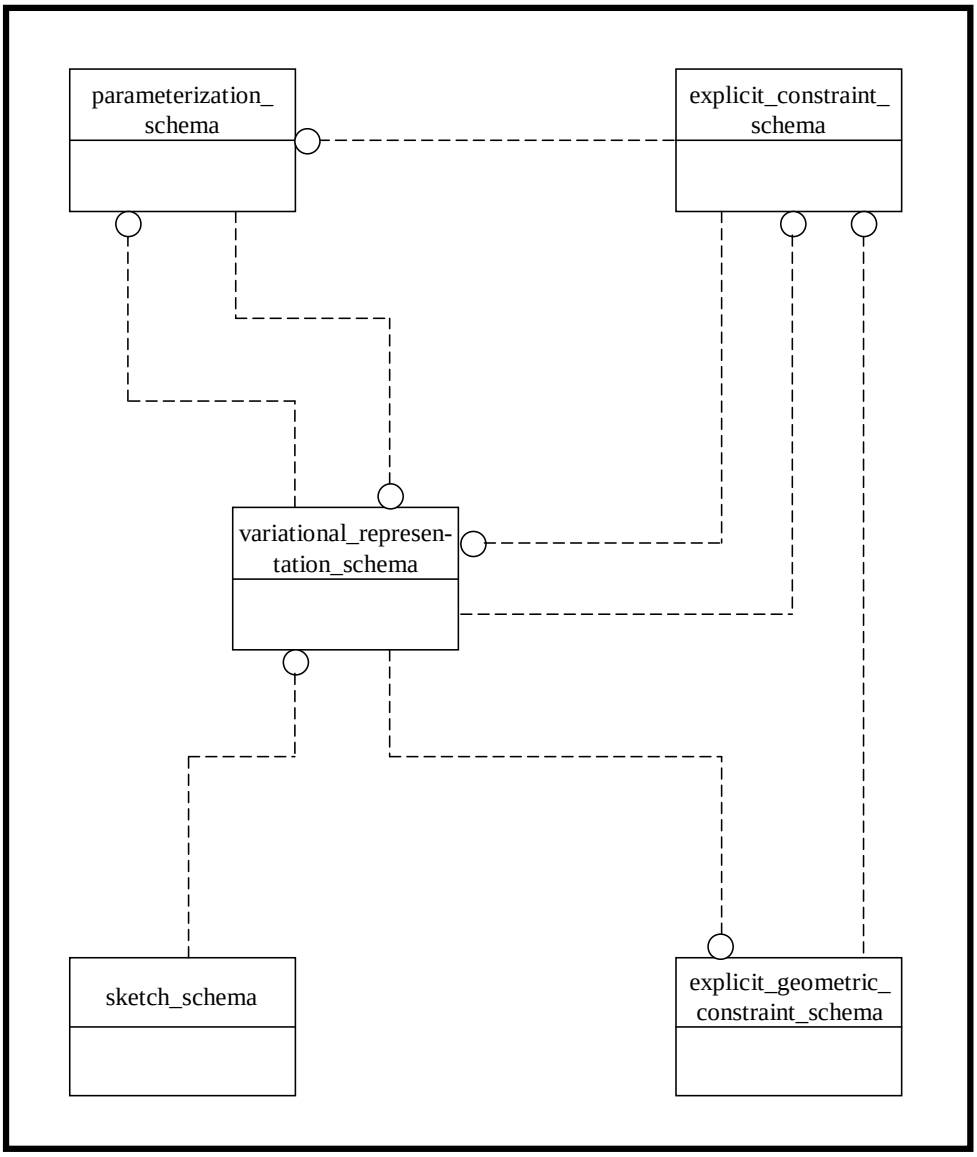

Figure 2 – Schema level diagram of relationships among ISO 10303-108 schemas

Industrial automation systems and integration — Product data representation and exchange —

Part 108: Integrated application resource: Parameterization and constraints for explicit geometric product models

1 Scope

This part of ISO 10303 specifies the resource constructs for the representation of model parameters and constraints, together with the mechanisms necessary for associating them with geometric or other elements of transferred models. The use of these capabilities potentially allows certain aspects of the behaviour of a model in its originating system to be conveyed together with the basic model itself. The intention in transferring this additional information is to provide the receiving system with data that will enable it to reconstruct corresponding behavioural characteristics in the model following the transfer. Ideally, this will enable the model to be edited in the receiving system just as as though it had been created there. That would not be possible without the exchange of what is known as *design intent* information. This part of ISO 10303 enables the capture and transfer of an important aspect of design intent.

Clause 4 defines a means for the association of model parameters with individual quantities in the model, to provide for the editing of dimensional values and other variable attributes. Clause 5 provides for the modelling of constraint relationships in terms of mathematical relationships among model parameters. Such constraints are not restricted to the modelling of product shape; they are designed to be applicable in any situation where it may be useful to capture and transfer mathematically specified relationships. Clause 5 also defines a class of descriptive (non-mathematical) constraints. Clause 6 defines a*variational representation*, containing a model together with all the associated parameters and constraints that potentially facilitate its effective editing following a transfer. Clause 7 defines specialized representations for a set of explicit geometric constraints applicable specifically to elements of shape models of the boundary representation and closely related types. These are subtypes of the descriptive constraints specified in clause 5. Finally, clause 8 provides representations for two-dimensional sketches, which form the basis of many shape construction methods in CAD modelling. Sketches are an important application area for the parameterization and constraint information defined in earlier clauses. The capabilities provided in this schema may also be used as a basis for the representation of parameterized drawings.

The following are within the scope of this part of ISO 10303:

— Parameterization of models through the association of variables with quantities occurring in them;

— Constraints defining mathematical relationships between parameters;

— Constraints on models expressed as relationships between their constituent elements or attributes of those elements;

— Association of parameters and constraints with models at the **representation** level, to create variational models of products, plans, processes or organizations;

— Specific representations for the geometric constraints commonly used in product shape modelling;

— Applications of parameterization and constraints to two- and three-dimensional shape models, in particular two-dimensional geometric sketches;

— Representation of models that are incompletely defined in the sense that certain values in the model may be regarded as not fully constrained.

NOTE 1 The scopes of the individual schemas comprising this part of ISO 10303 are specified in more detail in clauses 1.1 – 1.5.

The following are outside the scope of this part of ISO 10303:

— Procedural or history-based model representations, expressed in terms of sequences of constructional operations;

NOTE 2 Procedural or history-based representations are the subject of ISO 10303-55.

— Implicit or procedurally defined constraints;

— Solution methods for systems of constraint equations;

— Form features of shape models;

— Behaviour of a system in which a variational model is edited following a transfer;

NOTE 3 The information transmitted by the use of this part of ISO 10303 is intended to allow implementers to provide ‘reasonable’ or ‘intuitive’ behaviour by their systems in the circumstances mentioned, but this document does not prescribe the detailed nature of such behaviour or of its presentation to the system user.

— Considerations of accuracy in constraint satisfaction.

NOTE 4 Such considerations are essentially the same in this context as those arising in the geometric modelling of product shape. Limited general means are provided elsewhere in ISO 10303 for addressing accuracy issues.

Clauses 4, 5 and 6 of this part of ISO 10303 are applicable to any representation specified using the EXPRESS language defined in ISO 10303-11. The capabilities provided in those three schemas allow the association of parameters with attributes of entity data type instances and the specification of constraint relationships between such attributes in instantiated EXPRESS representations in general. These may include models of products, plans, processes or organizations, for example. Clauses 7 and 8 of this part of ISO 10303 are applicable more specifically to shape representations associated with discrete products. They provide additional specialization for the association of parameterization and constraints with geometric elements. The scopes of the individual clauses are given in more detail below.

1.1 Parameterization schema

The following are within the scope of the parameterization schema:

— The representation of variable parameters in an ISO 10303 model;

— The association of such variable parameters with quantities in an ISO 10303 model expressed as values of attributes of entity data type instances in a populated schema;

— The specification of allowable domains for model parameters;

— The specification of entity data type instance attribute values that are required to be fixed, i.e., whose allowable domains are each restricted to a single value. The intention is that these values should not be modified in a receiving system following a transfer.

NOTE 1 The last provision allows, in particular, for dimensional values to be fixed in appropriate cases.

NOTE 2 The presence of parameters in a model, and of specified relationships between parameters, indicate possibilities for changes that are permissible after a model transfer. This preserves part of the intention of the model's creator, as expressed by the parameterization imposed on the model in the originating system.

1.2 Explicit constraint schema

The following are within the scope of the explicit constraint schema:

— Specification of constraint relationships between constituent elements in a model, expressing the intention that these relationships should be preserved on modification of the model following an ISO 10303 transfer;

— Constraint relationships of the following specific types:

a) Free-form constraints enabling the capture and transmission of explicitly defined mathematical relationships between quantities in a model;

b) Defined constraints, for the representation of relationships defined descriptively, with implied mathematical semantics;

NOTE A constraint imposed in an originating system typically encapsulates requirements for the preservation of specific aspects of the functionality or validity of the modelled object. The transmission of such constraint information in a model exchange is intended to enable these requirements to be maintained during any subsequent modification of the model in the receiving system.

— Specification of sets of constraints that are required to hold simultaneously in the model.

The following are outside the scope of the explicit constraint schema:

— The actual representation of mathematical expressions, functions or procedures (such representations are provided elsewhere in ISO 10303);

— Solution methods for systems of constraint equations;

— Sequential application of constraints;

— Implicit or procedurally defined constraints.

1.3 Variational representation schema

The following are within the scope of the variational representation schema:

— Characterization of parameterization and constraint information as*variational*;

— Specification of the relationship between variational information and the explicit model with which it is associated;

— Specification of auxiliary geometric elements for use in defining variational relationships.

1.4 Explicit geometric constraint schema

The following are within the scope of the explicit geometric constraint schema:

— Explicit geometric constraints as descriptive relationships between points, curves and surfaces in a product shape model, both in two and in three dimensions.

NOTE CAD systems provide two-dimensional constraints for use in defining planar profiles that are extruded or rotated to generate volumes. Such volumes are often used to provide the basic forms for protrusions or depressions in a shape model. Three-dimensional constraints play an important role in the definition of inter-feature geometric relationships in part models and inter-part geometric relationships in assembly models.

1.5 Sketch schema

The following are within the scope of the sketch schema:

— Sketches as planar geometric constructs for use in the generation of surfaces and solids by means of sweep operations and other operations of related types;

— Neutral sketches, defined in general two-dimensional coordinate systems;

— Positioned sketches, defined in three-dimensional model space;

— Relationships between neutral sketches and positioned sketches defined by mappings;

— The use of parameters and constraints to define variational sketches;

— Subsketches as partial sketches subject to special treatment;

— Geometric elements imported into the plane of a sketch but defined in terms of other model elements lying outside that plane, used for constraining sketch elements with respect to external geometry.

EXAMPLE An example of an imported curve is a curve defined by projecting an external curve onto the plane of a sketch. Sketch elements may be constrained with respect to imported elements lying in the same plane.

NOTE The entities defined in this schema are also suitable for the representation of parametric drawings. The generation of such drawings (for example, by projection from a three-dimensional CAD model) will require specialized system capabilities to handle the graphical aspects, the relationships between component models and assembly models, etc. However, the actual representational requirements of a sketch and of a parameterized drawing are very similar.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO10303-1, *Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.*

ISO10303-11, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

ISO 10303-21, *Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.*

ISO 10303-41, *Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support.*

ISO 10303-42, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation.*

ISO 10303-43, *Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures.*

ISO 10303-50, *Industrial automation systems and integration — Product data representation and exchange — Part 50: Integrated generic resource: Mathematical constructs.*

ISO 10303-55:—[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 55: Integrated generic resource: Procedural and hybrid representation.*

ISO 13584-20, *Industrial automation systems and integration — Parts library — Part 20: Logical resource: Logical model of expressions.*

3 Terms, definitions and abbreviations

3.1 Terms defined in ISO 10303-1

For the purposes of this document, the following terms defined in ISO 10303-1 apply.

— application;

— application context;

— application protocol (AP);

— assembly;

— component;

— data exchange;

— exchange structure;

[1)] To be published

— implementation method;

— integrated resource (IR);

— product;

— product data;

— structure.

3.2 Terms defined in ISO 10303-11

For the purposes of this document, the following terms defined in ISO 10303-11 apply.

— entity;

— entity data type;

— entity (data type) instance;

— instance;

— value.

3.3 Terms defined in ISO 10303-42

For the purposes of this document, the following terms defined in ISO 10303-42 apply.

— boundary representation solid model (B-rep);

— constructive solid geometry (CSG);

— coordinate space;

— curve;

— dimensionality;

— geometrically founded;

— model space;

— surface.

3.4 Terms defined in ISO 10303-43

For the purposes of this document, the following terms defined in ISO 10303-43 apply.

— context of representation;

— element of representation;

— representation.

3.5 Terms defined in ISO 10303-50

For the purposes of this document, the following terms defined in ISO 10303-50 apply.

— expression;

— function;

— function domain;

— variable.

3.6 Terms defined in ISO 13584-20

For the purposes of this document, the following terms defined in ISO 13584-20 apply. In all cases the context is that of a mathematical variable.

— environment;

— semantics;

— syntactic representation.

3.7 Other terms and definitions

For the purposes of this document, the following definitions apply. Where synonyms are given the first term listed is the one preferred for use in the remainder of the document.

3.7.1
auxiliary geometry
geometric elements such as centre-lines and centre-planes used for locational or tolerancing purposes, but excluding geometry used in the actual representation of a product shape

3.7.2
constrained elements
model elements that are controlled by a constraint if the model is edited following transfer into another system

3.7.3
constraint
relationship between two or more elements in a model, which should be maintained in any modifications made subsequent to a model transfer

3.7.4
constraint solution
solution of a system of one or more constraint relationships expressed as mathematical equations or inequalities

NOTE Geometric constraint systems are often nonlinear, possessing multiple constraint solutions. The user may have to choose one of them to obtain a unique design.

3.7.5
constraint solver
software system for solving sets of equations that are mathematical representations of constraint relationships

NOTE 1 Methods used by constraint solvers for determining solutions are outside the scope of this part of ISO 10303.

NOTE 2 In straightforward cases a constraint set will have the same number of unknowns as equations. Ideally, however, a constraint solver should also handle underconstrained and overconstrained constraint sets having, respectively, more unknowns and less unknowns than the number of equations. The specification of system response in such cases is also outside the scope of this part of ISO 10303, but it could include the generation of diagnostic information or of partial solutions as appropriate.

3.7.6
construction geometry
geometry used as an aid in the construction of a shape that is not an element of the shape itself

EXAMPLE This concept may be illustrated by the case of a 'spine' curve used in the generation of certain types of swept surfaces. A planar generating curve is defined, together with a reference point lying in its plane. The reference point is then swept along the spine curve, the plane of the generating curve being maintained normal to it. The generating curve sweeps out the desired surface. If the reference point does not lie on the generating curve the spine curve does not lie on the generated surface, and it is therefore classified as construction geometry.

3.7.7
current result
representative member of the family of models defined by a parametric model resulting from the assignment of their current values to all parameters

NOTE 1 In the shape modelling context a current result may be an explicit model such as a wireframe, surface or boundary representation solid model. Alternatively, it may be a procedural or hybrid model in which the values of all parameters are regarded as fixed.

NOTE 2 Use of an explicit current result will be essential in the exchange of procedural or history-based models, though that is outside the scope of this part of ISO 10303. In that case the current result will be transmitted as a secondary model of the boundary representation or some closely related type, and used to check the accuracy of the result of re-evaluating a transmitted procedural model in the receiving system.

NOTE 3 In the context of this part of ISO 10303, the current result may indicate which choice of solution was made in the sending system in the case of multiple constraint solutions.

3.7.8
current value
specific value associated with a variable or parameter in a parameterized model, the value associated with the corresponding current result model

3.7.9
defined constraint
constraint whose semantics are assumed to be understood by both the sending and receiving system so that its mathematical details do not need to be transmitted explicitly

NOTE Collections of defined constraints may be created for use in specific application areas. The explicit geometric constraints specified in clause 7 of this part of ISO 10303 are examples of defined constraints whose application area is shape modelling. They all define constraint relationships descriptively rather than algebraically.

3.7.10
design intent
intentions of the designer of a model with regard to how it may be instantiated or modified

NOTE The aspects of design intent relevant to this part of ISO10303 are concerned with the information represented in the parameters and constraints associated with a model. More generally, design intent also includes the procedural or construction history of a model, which is the subject of ISO 10303-55. All aspects of design intent influence the behaviour of a model under editing operations.

3.7.11
dimensional constraint
constraint on the value of a dimension in a shape model

NOTE There is a semantic difference between a *dimension* and a *dimensional constraint*. A dimension specifies a fixed quantitative relationship between elements in a shape model. It captures a static characteristic of the model, and has no behavioural semantics. A dimensional constraint also specifies a quantitative relationship between model elements, in this case not necessarily a fixed one, with the additional significance that *any logical relationship underlying the dimensional relationship should be maintained if the model is modified in the receiving system.* A dimensional constraint may specify a numerical value, or a mathematical relationship from which a numerical value may be calculated.

EXAMPLE In two dimensions, the specification of a distance between two lines only makes sense if the lines are parallel. If this distance is formulated as a dimensional constraint, then any change in the value of the distance requires that the underlying logical relationship of parallelism between the lines is maintained.

3.7.12
directed constraint; asymmetrical constraint
constraint defined with respect to one or more reference elements, which are not changed by the action of the constraint if it is invoked during the modification of a model

NOTE Following a model transfer, a directed constraint only allows the constrained elements to be modified in such a way that their specified relationship to the reference elements is maintained. On the other hand, modification of one or more reference elements (from *outside* the context of the constraint concerned) will lead to compatible changes in all the constrained elements it controls.

EXAMPLE 1 Suppose that Line 2 is constrained to be parallel to the reference element Line 1. However, Line 1 may be a constrained element in some other constraint. If the action of the second constraint causes a change in the direction of Line 1 then the first constraint should ensure that Line 2 changes correspondingly.

EXAMPLE 2 The *undirected* constraint corresponding to the first of the constraints in the previous example would simply assert that “Line 1 and Line 2 are parallel”.

3.7.13
element
in the context of this part of ISO 10303, an instance of a geometric entity data type that participates in a model or is involved in a constraint instance

3.7.14
explicit constraint; declarative constraint
a relationship between specific entity data type instances defined by a constraint instance in a populated EXPRESS schema

NOTE The constraint relationship may be a *defined* one, whose semantics are specified descriptively, or it may be a *free-form* one specified in terms of a mathematical equation or inequality.

3.7.15
explicit geometric constraint
specialized explicit constraint relating geometrical or topological elements occurring in a shape model, or auxiliary geometric elements associated with that model

NOTE Examples of explicit geometric constraints include relationships between geometric elements of a model, such as parallelism or tangency.

3.7.16
explicit model; declarative model; evaluated model
model whose full details are immediately available without the need for any form of calculation

NOTE In the case of product shape models, an explicit (or *evaluated*) shape model is a fully detailed model of the boundary representation or some related type, as defined in ISO 10303-42. More specifically, an explicit model is a model that is not of the procedural or hybrid type, which may contain little or no explicit geometry.

3.7.17
feature
local geometric configuration belonging to a product shape description, having significance in some application context associated with the product model

3.7.18
free-form constraint
constraint formulated by a system user for some specific purpose, expressed in terms of an explicit mathematical relationship between parameters

NOTE A free form constraint needs to be defined for any relationship not covered by the defined constraints provided in applicable EXPRESS schemas.

3.7.19
hybrid model
model combining the use of both explicit and procedural representation methods

3.7.20
implicit constraint; procedural constraint
constraint that is not explicitly represented but is inherent in the action of some constructional operation used in building a model

NOTE 1 An implicit constraint will be realized as a static relationship in the current result after the operation has been performed, but not generally recorded in the model as a corresponding explicit constraint.

EXAMPLE A hypothetical procedure '**create_parallel_line**' could be used in a procedural description of a model in two dimensions. Its input is Line 1 (assumed to exist before the procedure is used) and a distance. The output is Line 2, created by the procedure at the specified distance from Line 1. However, since Line 2 does not exist until after the procedure has been invoked (i.e., the model has been at least partially *evaluated*) the constraint is not explicit in the model defined purely in terms of the constructional procedure. It is merely implied by the nature of the procedure.

NOTE 2 Although implicit constraints are out of scope of this part of ISO 10303, they are defined here to make an important distinction between two fundamental approaches to constraint representation. The present resource is intended to be compatible with other resources for the exchange of procedurally defined models containing implicit constraints.

3.7.21
imported geometry
points and curves imported into the plane of a two-dimensional sketch (see the definition of**sketch**), but defined in terms of elements external to the sketch

EXAMPLE Examples of imported geometry are provided by projections of external points or curves onto the plane of a sketch. These elements are not subject to editing within the context of the sketch, since their defining elements lie outside it. Their purpose is to serve as reference elements for constraints on the sketch geometry.

NOTE This definition applies only in the restricted context of this part of ISO 10303; the concept of imported geometry may have a wider meaning in other application areas.

3.7.22
incidence
inclusion of one point set in another

NOTE In the context of the **explicit_geometric_constraint_schema** (see clause 7 of this part of ISO 10303) the points sets are points, curves and surfaces. An incidence constraint may, for example, require a line to lie on a surface, or a curve to pass through a set of points.

3.7.23
logical constraint
constraint involving no dimensional value

NOTE Logical geometric constraints include the assertion of such conditions as parallelism, perpendicularity and tangency. By contrast, dimensional constraints have associated numerical values.

3.7.24
model parameter
explicitly represented variable associated with some quantity in a model

NOTE 1 Assignment of different values to model parameters generates different members of a family of models. Model parameters therefore express design freedom in a model, according to the parameterization scheme imposed by its creator. Limitations may be defined on the allowable ranges of model parameters.

EXAMPLE The dimensions of a generic block may be represented by model parameters L (length), W (width) and H (height). Individual members of the family of blocks are specified by assigning numerical values to the three parameters independently. Alternatively, relationships may be defined between the model parameters, such as $L = 2W, H = 0.5W$, to restrict the size of the family and define it in terms of the single independent model parameter W.

NOTE 2 Distinction must be made between the use of the word **parameter** in this part of ISO 10303, in ISO 10303-11, in ISO 10303-42 and in ISO 10303-50. In ISO 10303-11 a parameter is used for the formal representation of an input to, or output from, a function or procedure defined in an EXPRESS schema. In ISO 10303-42 a parameter is a variable used to identify the position of a point on a curve or a surface, so that the parameter may be thought of as an input to a function whose output is a coordinate value. In ISO 10303-50 a parameter is defined as 'a free variable in an expression'. In this part of ISO 10303 the term **model parameter** is used for a variable that controls dimensions or other gross characteristics of a model, for example the overall shape of a product model. A model parameter may be thought of as an input to a procedure, in this case a procedure that computes one instance of a family of shape models. It is unfortunate that the word 'parameter' is in widespread current use for such a variety of purposes. Although at a broad conceptual level the usages within ISO 10303 are similar, there are significant differences in such matters as the way the functions or procedures are defined and in the scope of parameters in a model.

3.7.25
overconstrained
defined in terms of more constraints than are needed to determine a constraint solution

NOTE In an overconstrained model, one or more of the constraints will be either redundant (i.e., consistent with all the others but not further limiting possible solutions) or inconsistent (i.e., not compatible with the others). An inconsistent constraint system has no solution. A designer may intentionally create redundant constraints in a model, but then care is necessary when editing the model to avoid introducing inconsistency.

3.7.26
parameter
model parameter (in the context of this part of ISO 10303 — see the definition of **model parameter** given earlier)

3.7.27
point set
set of points in space

NOTE The point set concept permits a unified abstract view of shape elements; for example, a line may be regarded as an infinite set of points all satisfying a specified relationship to each other, and curves and surfaces may be viewed in a similar manner.

3.7.28
procedural model; generative model; history-based model
model described in terms of the operation of a sequence of procedures (which may include the solution of constraint sets), as opposed to an explicit or evaluated model which captures the end result of applying those procedures

NOTE Although procedural models are outside the scope of this part of ISO 10303, they are defined here to make an important distinction between two fundamentally different modelling approaches. The present resource is intended to be compatible with ISO 10303-55, which provides representations for the exchange of procedurally defined models.

3.7.29
reference element
model element to which constrained elements are related by a directed constraint

3.7.30
sketch
two-dimensional geometric construct used as input to certain three-dimensional modelling operations

NOTE 1 A sketch is made up of point and curve elements, which may be related by explicit constraints. Its elements may also be constrained with respect to **imported geometry** (see clause 3.7.21), which is invariant within the plane of the sketch. An example of a sketch is provided in annex F.

NOTE 2 The definition given above is restricted to the context of CAD operations, but the representations provided for sketches in this part of ISO 10303 are also suitable for the capture of parameterized drawings.

3.7.31
underconstrained
containing insufficiently many constraints to allow the determination of a specific constraint solution for a model

NOTE Essentially, one or more degrees of freedom still remain in an underconstrained model. This is a natural situation in the early stages of design where some decisions have been made and embodied in constraints, but not others. This part of ISO 10303 is designed to allow the exchange of underconstrained models (together with a current result as a representative example). The remaining flexibility may then be used in the receiving system, following a model exchange, to optimize the model for its intended purpose.

EXAMPLE The designer may create a design that (although fully constrained to meet functional requirements) leaves certain detail shape elements underconstrained. A manufacturing engineer can then use the remaining degrees of freedom to derive a final evaluated model by incorporating manufacturing constraints to minimize production costs.

3.7.32
undirected constraint; symmetrical constraint
constraint having no reference element, requiring the constrained condition to hold amongst all pairs of members of a set of constrained elements

NOTE 1 This term should not be confused with **symmetry_geometric_constraint**, which is a specific type of geometric constraint defined in clause 7.4.25.

3.7.33
unevaluated model
procedural model, hybrid model containing both explicit and procedural elements, or model containing constraint sets that have not yet been solved

3.7.34
variational
defined in terms of explicitly specified parameters and constraints, allowing variation by change of parameter values, subject to continued satisfaction of imposed constraints

EXAMPLE Consider a configuration of two distinct lines through a given fixed point, both tangent to a circle with a fixed centre, and suppose that a parameter is associated with the radius of the circle. The constraints in this configuration (apart from the invariance of the two fixed points) are

a) the requirement that both lines pass through the given point;

b) the requirement that both lines are tangent to the circle.

If now the radius of the circle is changed by varying the parameter value, a new configuration should result in which the circle and the two lines are all different from the originals but the constraints continue to be satisfied. This is an example of a variational configuration.

3.7.35
well constrained
defined in terms of a set of constraints that are both necessary and sufficient to allow the determination of a unique constraint solution

NOTE In the ideal situation, only one solution satisfies all the constraints. In practice multiple solutions often arise. In an ISO 10303 model transfer using the capabilities defined in this document, the current result may often be used to determine which solutions were chosen by the original designer. For example, in two dimensions the constraint “Line 2 is parallel to Line 1 at distance 3” has two solutions; Line 2 may lie 3 units either to the left or the right of Line 1. The choice of solution actually made in the sending system will be evident from the current model. Another method which can be used to distinguish chosen solutions of multi-solution constraints or constraint sets is the specification of ‘near-points’. These may be used, for example, to indicate the approximate desired location of a point of tangency. Several of the explicit geometric constraints defined in clause 7 of this part of ISO 10303 provide for the specification of near-points.

3.8 Abbreviations

AP	application protocol (of ISO 10303)
B-rep	boundary representation
CAD	computer aided design
CSG	constructive solid geometry
IR	integrated resource (of ISO 10303)

4 Parameterization

4.1 Introduction

The following EXPRESS declaration begins the parameterization schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA parameterization_schema;

REFERENCE FROM support_resource_schema             -- ISO 10303-41
  (identifier,
   label,
   text);

REFERENCE FROM representation_schema               -- ISO 10303-43
  (representation_item,
   using_representations);

REFERENCE FROM mathematical_functions_schema       -- ISO 10303-50
  (finite_space,
   maths_number,
   maths_value,
   maths_variable,
   member_of,
   positive_integer);

REFERENCE FROM variational_representation_schema   -- ISO 10303-108
  (variational_representation_item);

REFERENCE FROM ISO13584_generic_expressions_schema -- ISO 13584-20
  (environment,
   generic_variable,
   variable_semantics);
(*
```

NOTE 1 The schemas referenced above can be found in the following parts of ISO 10303 and ISO 13584:

`support_resource_schema`	ISO 10303-41
`representation_schema`	ISO 10303-43
`mathematical_functions_schema`	ISO 10303-50
`variational_representation_schema`	Clause 6 of this part of ISO 10303
`ISO13584_generic_expressions_schema`	ISO 13584-20

NOTE 2 See annex D, Figure D.1 – D.2, for a graphical presentation of this schema.

4.2 Fundamental concepts and assumptions

This schema provides representation methods for the following:

— Variables, represented by instances of the entity data types **bound_model_parameter** or **unbound_model_parameter**, expressing variation or design freedom in a representation or model;

— A means for binding a **bound_model_parameter** instance to an attribute of another entity data type instance in the same **representation**;

— Domains of validity for instances of **bound_model_parameter** and **unbound_model_parameter**;

— A means for fixing the values of attributes of specific entity data type instances in a model, equivalent to the use of **bound_model_parameter** instances with constant associated values.

These resource constructs are of general utility in the exchange and sharing of ISO 10303 models embodying

— the capability for variation of attribute values in a model following an exchange;

— the capture and transfer of constraint relationships defined in terms of mathematical expressions, functions or procedures. Specifically, model parameters can participate in instances of **free_form_constraint** as defined in clause 5.

Clause 6.3.1 of this part of ISO 10303 defines **variational_representation_item** as a subtype of the ISO 10303-43 entity data type **representation_item**. Model parameters are defined as subtypes of **variational_representation_item**, which is the supertype of all entity data types used to express the variational aspects of models with explicit parameterization and constraints. The type of **representation** in which they participate is a **variational_representation**, as defined in clause 6.3.3.

4.2.1 Model parameters

An abstract entity data type **model_parameter** is provided, with two instantiable subtypes, **bound_model_parameter** and **unbound_model_parameter**. These allow for the capture and transmission of permitted aspects of model variation that can be exploited in a receiving system. A **bound_model_parameter** is bound to an attribute of an entity data type instance in an ISO 10303 model, in which case it provides a syntactic representation of the value of that attribute, for example a dimensional value. By contrast, an **unbound_model_parameter** is not directly associated with any model attribute. Either kind of **model_parameter** may be used in mathematical relationships defined in free-form constraints. The current value of a **model_parameter** is specified by one of its attributes; in the bound case the value of this attribute is required by an informal proposition to be the same as the value of the attribute to which it is bound.

The entity data type **model_parameter** is defined as a subtype of **variational_representation_item**, and the scope of its instantiable subtypes is therefore defined by those instances of **variational_representation** in which they participate. It is also a subtype of the ISO 10303-50 entity data type **maths_variable**, from which it inherits an attribute **values_space**, of ISO 10303-50 type **maths_space**. This attribute specifies the domain of validity for values of the **model_parameter**. These may include domains corresponding to those of the EXPRESS data types REAL, INTEGER, BOOLEAN and STRING, together with various bounded subsets of the REAL and INTEGER domains. This part of ISO 10303 does not directly provide the use of parameters having values belonging to aggregate types, but applications may define such extensions if they are required.

EXAMPLE 1 Consider a rectangle, with length x units and width y units. Here x and y are variables or parameters. An explicit constraint relationship $x = y^2 + 2$ relates these dimensions. Valid parameter ranges $100 \leq x \leq 300$ and $2.0 \leq y \leq 5.0$ are defined. In this case the two variables correspond to instances of **bound_model_parameter**, both bound to physical quantities in the model, i.e., dimensional attributes of the rectangle. The parameterization and constraint information may be transmitted together with a 'current result' — an explicit model of a rectangle with length 18.0 units and width 4.0 units. These parameter values satisfy the con-

straint and fall within the required parameter ranges. When model transfer is complete, if one of the parameters is edited the other should adjust accordingly to maintain satisfaction of the constraint, provided the parameters remain within their valid ranges. It is assumed that the necessary functionality for parameter variation and constraint maintenance will be provided by the receiving system.

The following example illustrates the use of an **unbound_model_parameter**.

EXAMPLE 2 Suppose an instance of **right_circular_cylinder** (as defined in ISO 10303-42), has associated instances of **bound_model_parameter** associated with its **radius** and **height** attributes, here denoted by r and h respectively. A third parameter, denoted by t, may be used to control the values of both r and h according to the relationships $r = 3t - 2, h = t^2 + 1$. In the case when t is not bound to an attribute of any entity data type instance, it will appropriately be modelled in terms of an **unbound_model_parameter**.

4.2.2 Parameter binding to an instance attribute

A **bound_model_parameter** is associated with an attribute of an entity data type instance in a populated schema, whose value represents the value of the parameter. This association is defined through the use of an entity data type **instance_attribute_reference** that simply specifies the name of an attribute and the instance to which it belongs (see clause 4.4.6). A simple example is given below to illustrate the principle, and the intended usage of the mechanism is more fully documented in clause F.1 of annex F. Once the parameter binding has been established, the parameter may participate in a relationship that governs its value if the model is subsequently edited in a receiving system.

EXAMPLE For the purpose of the example, entity data types defined in the ISO 10303 integrated generic resources are treated as though they are instantiable elements in an application protocol.

It is desired to parameterize one dimension of a **block** solid, as defined in ISO 10303-42. This has three attributes, `x`, `y` and `z`, that prescribe its three principal dimensions. In any instantiation of the block these will have specific real numerical values. Consider now the following fragment of an ISO 10303-21 transfer file:

```
#290 = AXIS2_PLACEMENT_3D(...);
#300 = BLOCK('BLOCK1', #290, 4.0, 6.0, 8.0);
#310 = INSTANCE_ATTRIBUTE_REFERENCE
         ('GEOMETRIC_MODEL_SCHEMA.BLOCK.X', #300);
#320 = FINITE_REAL_INTERVAL(2.0, .CLOSED., 10.0, .CLOSED.);
#330 = BOUND_MODEL_PARAMETER
         ('XPARAM', #320, *, 'BLOCK X-DIMENSION', *);
#340 = BOUND_PARAMETER_ENVIRONMENT(#310, #330);
```

The instances represented above are explained as follows:

`#290:` defines an ISO 10303-42 axis placement (details omitted) for the next instance;

`#300:` the **block** instance. As a subtype of ISO 10303-43 **representation_item**, this inherits a **name** attribute of type **label**, whose value in this instance is `'block1'`. The block is defined with respect to the axis placement `#290` and has dimensions 4.0, 6.0 and 8.0 units;

`#310:` an instance of **instance_attribute_reference** in which the specified attribute name is given as `'geometric_model_schema.block.x'` and the owning **block** instance is `#300`. Note that the attribute name appears fully qualified with the name of the owning entity data type and its defining schema. This entry in the file identifies the particular instance whose specified attribute is to be associated with the **bound_model_parameter** instance;

`#320:` defines the domain of that parameter, a real interval closed at both ends, bounded below by 2.0 and above by 10.0. The entity data type **finite_real_interval** is defined in ISO 10303-50;

#330: specifies the **bound_model_parameter** itself, using the definition in 4.4.2 of this schema. Its attribute value list contains these entries:

— a label, 'xparam', corresponding to the **name** attribute of its **representation_item** supertype;

— a domain #320, corresponding to the **values_space** attribute of its **maths_variable** supertype;

— a derived attribute, representing the **label** 'xparam' again — this corresponds to the **name** attribute of the **maths_value** supertype, and its value is derived from the corresponding attribute of the **representation_item** supertype to ensure that the two labels are the same;

— a textual description 'block x-dimension';

— the value of the **model_parameter**, given as a derived value, although no formal method is available in EXPRESS for deriving it from instance #300;

#340: an instance of **bound_parameter_environment**, defined in 4.4.4, providing the link between the specified instance attribute, #310, and the parameter bound to it, #330.

At this point the binding of the parameter to the desired attribute is complete. The intention is that the value attribute of the bound **bound_model_parameter** instance #330 is equal to the value 4.0 associated with the 'x' attribute in the **block** instance, and lies within the domain of validity represented by #320. However, because the EXPRESS language provides no formal way of asserting this the value attribute of the parameter is recorded as indeterminate, and an informal proposition in 4.4.2 requires that the two values shall be equal on completion of the transfer. The achievement of this is the responsibility of the translation software. The parametric relationship having been captured in an exchange file as shown above, the x-dimension of the block may now be controlled in terms of the parameter associated with it if the model is edited following transfer into a receiving system.

4.3 Parameterization type definitions

4.3.1 attribute_identifier

The **attribute_identifier** type is a type of **identifier** whose values are restricted to valid possibilities for entity data type attribute names as defined in ISO 10303-11.

EXPRESS specification:

```
*)
TYPE attribute_identifier = identifier;
WHERE
   WR1: validate_attribute_id(SELF);
END_TYPE;
(*
```

Formal propositions:

WR1: The identifier shall represent a valid attribute identifier as specified in ISO 10303-11.

NOTE Permitted characters in an EXPRESS attribute name include the alphanumeric characters together with the special characters '_', '[', ']', '.', and '\'. This set is sufficient for fully qualified attribute names (see clause 9.2.3.4. of ISO 10303-11), including SELF references and specification of individual members of aggregate-valued attributes.

4.4 Parameterization entity definitions

4.4.1 model_parameter

The **model_parameter** entity data type is a type of **variational_representation_item** that represents a variable quantity in a **variational_representation** (see clause 6). It is also a type of **maths_variable** as defined in ISO 10303-50, and can therefore participate in mathematical relationships. Its attributes include an optional textual description of the significance of the parameter, and a current parameter value. A **model_parameter** instance inherits a **name** attribute from both its supertypes; the value of the **maths_variable** name is derived from the **representation_item** name to ensure consistency. It also inherits an attribute **values_space** from its **maths_variable** supertype, specifying the domain (permissible set of values) of the **model_parameter**.

NOTE 1 The fact that **model_parameter** is a type of **maths_variable** restricts its underlying domain of values to subsets of the real or integer numbers, Booleans or strings. Future editions of this part of ISO 10303 have the possibility of extending that spectrum of domains, if applications require it, by making use of the more general capabilities of ISO 10303-50.

Because **model_parameter** is a type of **maths_variable**, and hence ultimately of **generic_variable** as defined in ISO 13584-20, each instance of it is required to have an associated instance of the ISO 13584-20 entity data type **environment**, which links the parameter with its associated semantics. For that purpose, this schema provides appropriate subtypes of **environment**, namely **bound_parameter_environment** and **unbound_parameter_environment**, as defined in clauses 4.4.4 and 4.4.5 respectively. The key relationships between these ISO 10303-108 and ISO 13584-20 entity data types are shown in clause F.1 of annex F.

EXPRESS specification:

```
*)
ENTITY model_parameter
  ABSTRACT SUPERTYPE OF (ONEOF (bound_model_parameter,
                                unbound_model_parameter))
  SUBTYPE OF (variational_representation_item, maths_variable);
  parameter_description : OPTIONAL text;
  parameter_current_value : maths_value;
DERIVE
  SELF\maths_variable.name : label := SELF\representation_item.name;
WHERE
  WR1 : member_of(parameter_current_value,
    SELF\maths_variable.values_space);
END_ENTITY;
(*
```

Attribute definitions:

parameter_description: An optional description, for human interpretation, of the significance of the **model_parameter** instance.

parameter_current_value: The current value associated with the **model_parameter** instance.

SELF\maths_variable.name: The value of the **name** attribute from the **maths_variable** supertype, whose value is derived from that of the **representation_item** supertype, to ensure consistency.

SELF\representation_item.name: The value of the **name** attribute from the **representation_item** supertype.

SELF\maths_variable.values_space: The domain of validity of the current value associated with the **model_parameter** instance.

Formal propositions:

WR1: The current value of the **model_parameter** instance shall lie within the domain specified by the attribute **SELF\maths_variable.values_space**.

NOTE 2 No requirement has been imposed for the **name** attribute of a **model_parameter** instance to have a value that is unique in any **representation** it participates in. This is because it is not anticipated that **model_parameter** instances will be referred to by their name attributes. In general, **model_parameter** instances used in an exchange are referenced by ephemeral identifiers created during the translation process and discarded when the exchange is complete, by which time system-dependent identifiers have been generated in the receiving system. However, if uniqueness of name attributes is required for some application purpose the necessary restriction can be imposed in schemas that specialize definitions from this part of ISO 10303.

4.4.2 bound_model_parameter

The **bound_model_parameter** entity data type is a type of **model_parameter** whose instances can be bound to (associated with) explicit attributes of entity instances participating in a **variational_representation**. The current value of any instance of **bound_model_parameter** is indeterminate during an exchange, but is required by an informal proposition to be set equal in the receiving system to the value of the attribute to which it is bound. That attribute is therefore required to have an explicit value in a populated schema, which rules out the association of a **bound_model_parameter** with a derived or inverse attribute.

NOTE 1 This approach to the association of a value with a **bound_model_parameter** instance is necessary because the EXPRESS language provides no means for the formal derivation of the value from the referenced entity data type instance in a populated schema.

EXPRESS specification:

```
*)
ENTITY bound_model_parameter
  SUBTYPE OF (model_parameter);
DERIVE
  SELF\model_parameter.parameter_current_value : maths_value := ?;
WHERE
WR1 : 'PARAMETERIZATION_SCHEMA.BOUND_PARAMETER_ENVIRONMENT'
    IN TYPEOF(SELF\generic_variable.interpretation);
END_ENTITY;
(*
```

Attribute definitions:

parameter_current_value: The current value of the attribute to which the parameter is bound, always derived as indeterminate for an instance of this entity data type (see notes 1, 2 and 3).

SELF\generic_variable.interpretation: The instance of **bound_parameter_environment** that links a **bound_model_parameter** instance to a particular entity instance attribute.

Formal propositions:

WR1: Every instance of **bound_model_parameter** shall reference an instance of type **bound_parameter_environment**.

NOTE 2 The indeterminate value of the the attribute **parameter_current_value** does not give rise to a violation of WR1 of the **model_parameter** supertype, which for an instance of **bound_model_parameter** will evaluate to UNKNOWN rather than FALSE. Clause 9.2.2.2 of ISO 10303-11 states that this does not constitute a violation of the rule. In practice it is the responsibility of the translator software to check that the value of the referenced attribute lies within the domain of the parameter.

NOTE 3 Because the indeterminate value of **parameter_current_value** is derived — in this case, by a simple assigment — an instance of **bound_model_parameter** in an ISO 10303-21 exchange file will represent it by an asterisk, *, as illustrated in the examples in clause 4.2.2 and annex F.1.

NOTE 4 The ISO 13584-20 entity data type **environment** has two attributes, **semantics** and **syntactic_representation**. As a subtype of **environment**, **bound_parameter_environment** also possesses these attributes, which are treated as follows:

semantics: This attribute is of type **instance_attribute_reference** (see clause 4.4.6).

syntactic_representation: A WHERE rule applying to **bound_parameter_environment** requires that the value of this attribute shall be of type **bound_model_parameter**.

The ISO 13584-20 entity data type **generic_variable** has an inverse attribute **interpretation**, corresponding to the **syntactic_representation** attribute of **environment**. For **bound_model_parameter**, a subtype of **generic_variable**, this inverse attribute is required by WR1 to be of type **bound_parameter_environment**.

The entity data types **environment** and **variable_semantics** are subtyped in this part of ISO 10303 to satisfy a requirement of ISO 13584-20 regarding the binding of values to variables. An EXPRESS-G representation of their relationships with entity data types defined in this schema is given in clause F.1 of annex F.

Informal propositions:

IP1: The **parameter_current_value** attribute shall have the same value as the entity data type instance attribute referenced via the inverse attribute **SELF\generic_variable.interpretation**, and shall be type-compatible with it.

NOTE 5 It will be crucial for implementations to ensure that the foregoing informal proposition is satisfied. Means cannot be provided in this schema for checking its validity because no formal mechanism exists for accessing the value of the attribute with which the **bound_model_parameter** is associated.

NOTE 6 A local rule in the definition of **variational_representation** (see clause 6.3.3) ensures that any instance of **bound_model_parameter** shall belong to the same **variational_representation** as the entity data type instance to whose specified attribute it is bound.

NOTE 7 No restriction is imposed in this schema to prevent the binding of more than one **bound_model_parameter** to a single attribute of an entity data type instance. However, if such a restriction is required for some application purpose it can be specified in schemas that specialize definitions from this part of ISO 10303.

NOTE 8 The mechanism defined in this schema does not allow the direct association of a **bound_model_parameter** instance with more than one entity data type instance attribute. The effect of such a multiple binding can be achieved through the use of multiple **bound_model_parameter** instances related by the **equal_parameter_constraint** as defined in clause 5.4.3.

4.4.3 unbound_model_parameter

The **unbound_model_parameter** entity data type is a type of **model_parameter** representing a variable that is not bound to an attribute of any entity instance in the model. The value attribute of an **unbound_model_parameter** instance is specified explicitly, rather than by association with an attribute of some other instance in the model.

NOTE 1 An instance of **unbound_model_parameter** may be used in mathematical expressions in free-form constraints that govern values associated with instances of **bound_model_parameter**. Examples of this usage are given in clause 4.2.1 and clause F.1 of annex F.

EXPRESS specification:

```
*)
ENTITY unbound_model_parameter
  SUBTYPE OF (model_parameter);
WHERE
  WR1: 'PARAMETERIZATION_SCHEMA.UNBOUND_PARAMETER_ENVIRONMENT'
    IN TYPEOF(SELF\generic_variable.interpretation);
END_ENTITY;
(*
```

Attribute definitions:

SELF\generic_variable.interpretation: The instance of **unbound_parameter_environment** providing the link between the **unbound_model_parameter** instance and its associated instance of **unbound_model_parameter_semantics**. The definitions of these entities are given in Clauses 4.4.5 and 4.4.7.

Formal propositions:

WR1: Every instance of **unbound_model_parameter** shall be referenced by an instance of **unbound_parameter_environment**.

NOTE 2 The ISO 13584-20 entity data type **environment** has two attributes, **semantics** and **syntactic_representation**. As a subtype of **environment**, **unbound_parameter_environment** also possesses these attributes, which are treated as follows:

semantics: The value of this attribute is required to be of type **unbound_model_parameter_semantics** (see clause 4.4.7).

syntactic_representation: A WHERE rule applying to **unbound_parameter_environment** requires that the value of this attribute shall be of type **unbound_model_parameter**.

The ISO 13584-20 entity data type **generic_variable** has an inverse attribute **interpretation**, corresponding to the **syntactic_representation** attribute of **environment**. For **unbound_model_parameter**, a subtype of **generic_variable**, this inverse attribute is required by WR1 to be of type **unbound_parameter_environment**.

The entity data types **environment** and **variable_semantics** are subtyped in this part of ISO 10303 to satisfy a requirement of ISO 13584-20 regarding the binding of values to variables. An EXPRESS-G representation of their relationships with entity data types defined in this schema is given in clause F.1 of annex F.

4.4.4 bound_parameter_environment

The **bound_parameter_environment** entity data type is a type of **environment** as defined in ISO 13584-20. It provides a link between the syntactic and semantic aspects of a **bound_model_parameter** instance.

NOTE 1 ISO 13584-20 requires an instance of **environment** to be defined for every instance of **generic_variable**, and since **bound_model_parameter** as defined in this schema is a subtype of **generic_variable** it has been necessary to provide an appropriate subtype of **environment** in this schema.

EXPRESS specification:

```
*)
ENTITY bound_parameter_environment
  SUBTYPE OF (environment);
WHERE
  WR1: ('PARAMETERIZATION_SCHEMA.BOUND_MODEL_PARAMETER' IN
    TYPEOF(SELF\environment.syntactic_representation)) AND
    ('PARAMETERIZATION_SCHEMA.INSTANCE_ATTRIBUTE_REFERENCE' IN
    TYPEOF(SELF\environment.semantics));
END_ENTITY;
(*
```

Formal propositions:

WR1: For every instance of **bound_parameter_environment**, the **syntactic_representation** attribute of the **environment** supertype shall be of type **bound_model_parameter** and the **semantics** attribute shall be of type **instance_attribute_reference**.

NOTE 2 The relationships between the ISO 13584-20 entity data type **environment** and other entity data types defined in this schema are illustrated by the EXPRESS-G diagram in clause F.1 of annex F.

4.4.5 unbound_parameter_environment

The **unbound_parameter_environment** entity data type is a type of **environment** as defined in ISO 13584-20. It provides a link between the syntactic and semantic aspects of an **unbound_model_parameter** instance.

NOTE 1 ISO 13584-20 requires an instance of **environment** to be defined for every instance of **generic_variable**, and since **unbound_model_parameter** as defined in this schema is a subtype of **generic_variable** it has been necessary to provide an appropriate subtype of **environment** in this schema.

EXPRESS specification:

```
*)
ENTITY unbound_parameter_environment
  SUBTYPE OF (environment);
```

```
WHERE
  WR1: ('PARAMETERIZATION_SCHEMA.UNBOUND_MODEL_PARAMETER' IN
    TYPEOF(SELF\environment.syntactic_representation)) AND
    ('PARAMETERIZATION_SCHEMA.UNBOUND_MODEL_PARAMETER_SEMANTICS' IN
    TYPEOF(SELF\environment.semantics));
END_ENTITY;
(*
```

Formal propositions:

WR1: For any instance of **unbound_parameter_environment**, the **syntactic_representation** attribute of the **environment** supertype shall be of type **unbound_model_parameter** and the **semantics** attribute shall be of type **unbound_model_parameter_semantics**.

NOTE 2 The relationships between the ISO 13584-20 entity data type **environment** and other entity data types defined in this schema are illustrated by the EXPRESS-G diagram in clause F.1 of annex F.

4.4.6 instance_attribute_reference

The **instance_attribute_reference** entity data type is a type of **variable_semantics** (see ISO 13584-20). It identifies a named explicit attribute of a specific **representation_item** instance in a populated EXPRESS schema. The name of the attribute is specified as an **attribute_identifier** (see clause 4.3.1). Derived or inverse attributes shall not be referenced by the use of **instance_attribute_reference**.

NOTE 1 This entity data type is used in the definition of an association between a bound **model_parameter** and an attribute of an instance of a **representation_item**. It is defined as a subtype of the ISO 13584-20 entity data type **variable_semantics** to satisfy a requirement of that standard regarding the binding of values to variables. The intention is to provide an interpretation of the role of the variable in its modelling context.

Examples of the use of this entity data type are provided in clause F.1 of annex F.

EXPRESS specification:

```
*)
ENTITY instance_attribute_reference
  SUBTYPE OF (variable_semantics);
  attribute_name  : attribute_identifier;
  owning_instance : representation_item;
END_ENTITY;
(*
```

Attribute definitions:

attribute_name: An **attribute_identifier** specifying the name of the referenced attribute.

owning_instance: The **represention_item** instance owning the referenced attribute.

Informal propositions:

IP1: Any attribute referenced by an instance of **instance_attribute_reference** shall be specified as a fully qualified attribute in the form 'SCHEMA_NAME.ENTITY_NAME.ATTRIBUTE_NAME' (see ISO 10303-11).

NOTE 2 The foregoing informal proposition is imposed to ensure that ambiguities are avoided, for example in the case of complex instances, and to enable effective checking for compatibility of the referenced attribute identifier and its owning entity data type instance.

4.4.7 unbound_model_parameter_semantics

The **unbound_model_parameter_semantics** entity data type is a type of **variable_semantics** as defined in ISO 13584-20. It represents the semantics of an unbound **model_parameter**.

NOTE ISO 13584-20 requires a subtype of **variable_semantics** to be defined for any subtype of **generic_variable**. This is intended to provide an interpretation of the role of the variable in its modelling context. The definition of **unbound_model_parameter_semantics** given below reflects the fact that an unbound **model_parameter** has no semantics beyond what is implied by its use in an instance of **free_form_constraint** (see clause 5.4.4). It does not necessarily have any immediate physical significance in its own right, though it may indirectly control quantities that do have physical significance.

EXPRESS specification:

```
*)
ENTITY unbound_model_parameter_semantics
  SUBTYPE OF (variable_semantics);
END_ENTITY;
(*
```

4.4.8 fixed_instance_attribute_set

The **fixed_instance_attribute_set** entity data type is a type of **variational_representation_item** as defined in clause 6.3.1. It specifies a set of explicit attribute values in a populated schema whose values are required to remain fixed in the model after transfer to a receiving system. In particular, these fixed attributes may represent values of dimensions in a shape model. This entity data type shall not be used to constrain values of derived or inverse attributes.

NOTE Although this entity data type may appear to have the nature of a constraint, it has been included here rather than in the **explicit_constraint_schema** (clause 5) for the following reasons:

a) It makes use of **instance_attribute_reference** instances in the same manner as the **bound_model_parameter** entity data type defined in this schema;

b) The **explicit_constraint** entity data type defined in clause 5 constrains instances of **representation_item** rather than individual attributes of such instances;

c) A fixed attribute may be considered to be parameterized with a domain restricted to a single value.

EXPRESS specification:

```
*)
ENTITY fixed_instance_attribute_set
  SUBTYPE OF (variational_representation_item);
  fixed_attributes : SET[1:?] OF instance_attribute_reference;
WHERE
  WR1: SIZEOF(QUERY(q <* using_representations(SELF) |
    SIZEOF(QUERY(r <* q.items |
    'PARAMETERIZATION_SCHEMA.FIXED_INSTANCE_ATTRIBUTE_SET'
    IN TYPEOF(r))) > 1)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

fixed_attributes: The set of entity data type instance attributes that have fixed values in the model.

Formal propositions:

WR1: The current instance shall be the only instance of **fixed_instance_attribute_set** in any **representation** that contains it.

4.4.9 generated_finite_numeric_space

The entity data type **generated_finite_numeric_space** is a type of **finite_space** as defined in ISO 10303-50. It defines a finite space of numerical values whose members are not enumerated but are specified in terms of a starting value and an increment, corresponding to practice in many computer programming languages.

NOTE 1 The constructed set may be used as the domain of a numerical **model_parameter**.

EXPRESS specification:

```
*)
ENTITY generated_finite_numeric_space
  SUBTYPE OF (finite_space);
  start_value      : maths_number;
  increment_value  : maths_number;
  increment_number : positive_integer;
DERIVE
  SELF\finite_space.members : SET [2:?] OF maths_number
    := make_numeric_set(start_value, increment_value, increment_number);
WHERE
  WR1: increment_value <> 0.0;
END_ENTITY;
(*
```


Attribute definitions:

start_value: The base member of the constructed set.

increment_value: The numerical difference between successive members added to the set.

increment_number: The number of members added to the set after specification of its base member.

SELF\finite_space.members: The evaluated members of the set of **maths_number** instances defining the space.

Formal propositions:

WR1: The size of the increment shall be nonzero, and the number of increments shall be greater than zero.

NOTE 2 The number of members of the set is greater by one than the value of **increment_number**.

NOTE 3 The entity data type **generated_finite_numeric_space** is not referenced elsewhere in this part of ISO 10303, but is provided as a resource for schemas that may make use of the **ISO10303_parameterization_schema** or some specialization of it.

4.5 Parameterization function definitions

4.5.1 make_numeric_set

The **make_numeric_set** function creates a set containing two or more members of type **maths_number** (an ISO 10303-50 named type equivalent to the EXPRESS simple type NUMBER), from an initial value, an increment and an integer specifying the number of increments.

EXPRESS specification:

```
*)
FUNCTION make_numeric_set(start, delta : maths_number;
                          incs         : positive_integer)
                                       : SET [2:?] OF maths_number;

  LOCAL
    i : INTEGER;
    numeric_set : SET[2:?] OF maths_number := [start, (start + delta)];
  END_LOCAL;

  IF incs > 1 THEN REPEAT i := 2 TO incs;
                     numeric_set := numeric_set + (start + (i*delta));
                   END_REPEAT;
  END_IF;
  RETURN(numeric_set);
END_FUNCTION;
(*
```

Argument definitions:

start: The value of the initial member of the generated set of values.

delta: The increment between successive values generated.

incs: The number of increments (i.e., one less than the number of members of the generated set).

4.5.2 validate_attribute_id

The **validate_attribute_id** function checks whether the **attribute_identifier** specified in an instance of **attribute_reference** is valid in the sense that its first character is a letter and all the others are letters, digits, '_', '[',']','.' or '\'. The last two characters allow references to attributes inherited from supertypes where this is appropriate, and the two preceding ones to individual members of aggregate instances.

NOTE The characters that may occur in valid EXPRESS identifiers are specified in clause 7 of ISO 10303-11. The use of '.' and '\' in qualified attributes is specified in clause 9.2.3.4 of the same document.

EXPRESS specification:

```
*)
FUNCTION validate_attribute_id(attid : attribute_identifier) : BOOLEAN;

  CONSTANT
    letters      : SET[52:52] OF STRING :=
      ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p',
       'q','r','s','t','u','v','w','x','y','z','A','B','C','D','E','F',
       'G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V',
       'W','X','Y','Z'];
    numbers_etc  : SET[15:15] OF STRING :=
      ['0','1','2','3','4','5','6','7','8','9','_','[',']','.','\'];
    valid_chars  : SET[67:67] OF STRING := letters + numbers_etc;
  END_CONSTANT;

  LOCAL
    id_length : INTEGER := LENGTH(attid);
    id_valid  : BOOLEAN := TRUE;
    i         : INTEGER;
  END_LOCAL;

  -- check that indentifier starts with a letter

  IF NOT (attid[1] IN letters) THEN id_valid := FALSE;
  END_IF;

  -- check that no invalid characters occur subsequently

  REPEAT i := 2 TO id_length WHILE id_valid = TRUE;
    IF NOT (attid[i] IN valid_chars) THEN id_valid := FALSE;
    END_IF;
  END_REPEAT;
  RETURN(id_valid);

END_FUNCTION;
(*
```

Argument definitions:

attid: The attribute identifier whose validity is to be checked.

EXPRESS specification:

```
*)
END_SCHEMA; -- parameterization_schema
(*
```

5 Explicit constraint

5.1 Introduction

The following EXPRESS declaration begins the explicit constraint schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA explicit_constraint_schema;

REFERENCE FROM support_resource_schema            -- ISO 10303-41
  (text);

REFERENCE FROM representation_schema              -- ISO 10303-43
  (item_in_context,
   representation_item,
   using_representations);

REFERENCE FROM mathematical_functions_schema      -- ISO 10303-50
  (compatible_spaces,
   maths_variable,
   values_space_of);

REFERENCE FROM parameterization_schema            -- ISO 10303-108
  (bound_model_parameter,
   model_parameter,
   unbound_model_parameter);

REFERENCE FROM variational_representation_schema  -- ISO 10303-108
  (variational_representation_item);

REFERENCE FROM ISO13584_generic_expressions_schema -- ISO 13584-20
  (used_variables);

REFERENCE FROM ISO13584_expressions_schema         -- ISO 13584-20
  (boolean_expression,
   expression);
(*
```

NOTE 1 The schemas referenced above can be found in the following Parts of ISO 10303 and ISO 13584:

support_resource_schema	ISO 10303-41
representation_schema	ISO 10303-43
mathematical_functions_schema	ISO 10303-50
parameterization_schema	clause 4 of this part of ISO 10303
variational_representation_schema	clause 6 of this part of ISO 10303
ISO13584_generic_expressions_schema	ISO 13584-20
ISO13584_expressions_schema	ISO 13584-20

NOTE 2 See annex D, Figure D.3, for a graphical presentation of this schema.

5.2 Fundamental concepts and assumptions

This schema provides representations for general explicit constraints, applicable to a wide variety of models. Such constraints may be used, following a model exchange, to control the behaviour of the model if it is modified in the receiving system. They capture an important aspect of the originator's intent in creating the model, namely the requirement that certain of its characteristics should remain invariant if it is edited. This invariance is usually required to ensure the continued validity or functionality of whatever the model represents.

Explicit *geometric* constraints are provided in most current CAD systems in the context of product shape models, for the reasons given above. The representations defined in this schema, apart from their many other applications, provide the necessary foundations for the explicit geometric constraints specified in clause 7 of this part of ISO 10303.

In a static sense an explicit constraint is redundant, since its presence has no effect unless an attempt is made to change the model. The satisfaction of the constraint should ideally be checked during a model exchange, however, to ensure that the model and its constraints are mutually consistent. If an attempt is made to change the model following a data transfer, the effect of the constraint will become apparent, provided the receiving system has the requisite constraint functionality. The intended effect is that this system should cause the possibilities for change to be restricted to those for which the constraint remains valid.

EXAMPLE Consider an ISO 10303-21 file exchange of a product shape model, which contains two concentric circular edges. The concentricity may result from the fact that the axial lines of the circular curves underlying both edges happen to coincide. However, that does not imply any restriction on how the model may be changed following the transfer. Now suppose that a concentricity constraint is applied between the two circles concerned. This causes no changes to the geometry or topology of the shape model, as stated above. It asserts something that is already true in the model, but additionally it expresses the requirement that the condition should *remain* true after reconstruction in a receiving system, where modifications may be made. The explicit constraint is therefore not part of the shape description of the model as constructed, but governs what changes may subsequently be made to it.

An explicit constraint is usually an assertion of a specified relationship between two or more entity data type instances in a populated schema, as in the preceding example. The relationship may be defined in terms of values of *attributes* of entity data type instances, and it is then also possible to define relationships between different attributes of a single entity data type instance. The provisions of clause 4 allow the association of instances of **bound_model_parameter** with attributes of other instances in a populated schema, which enables the representation of such constraint relationships in terms of mathematical relations between parameters. In what follows, the word *element* will be used to refer to any form of entity data type instance involved in an explicit constraint. There should be no confusion with the usage of the same term in the context of finite element analysis.

Clause 6.3.1 of this part of ISO 10303 defines **variational_representation_item** as a subtype of the ISO 10303-43 entity data type **representation_item**. Explicit constraints are defined as subtypes of **variational_representation_item**, which is the supertype of all entity data types used to express the variational aspects of models with parameterization and constraints. The type of **representation** in which they participate is a **variational_representation**, as defined in clause 6.3.3 of this part of ISO 10303.

NOTE Many CAD systems use a procedural description of the model, expressed in terms of the operations used in constructing it. In the exchange of models of this kind, model regeneration in a receiving system will be achieved by performing those operations. In such a case *implicit* constraints may be present in the transferred procedural model. These are inherent in the use of certain constructional procedures. For example, a function may

be provided for the construction of a rectangular block with specified dimensions. As constructed, the block will have three pairs of opposite parallel faces, and adjacent faces will be perpendicular to each other. This will be the case regardless of the dimensions of the block; the parallelism and perpendicularity constraints are built into the operation of the creation function. Constraints defined in this way, by procedures, are outside the scope of this schema. However, this part of ISO 10303 is compatible with ISO 10303-55, the integrated generic resource for the representation of procedural and hybrid models.

5.2.1 Free-form and defined constraints

Explicit constraints may be formulated as mathematical relationships between attributes of entity data type instances. This schema defines a **free_form_constraint** entity data type, in which those mathematical relationships are specified in precise detail through a mechanism using subtypes of the **model_parameter** entity data type to represent attribute values.

A **defined_constraint** entity data type is also provided. This does not specify relationships explicitly; instead, it uses a descriptive form of definition. A receiving system is expected to understand the semantics of a defined constraint and to reformulate it mathematically in whatever way is most suitable for its geometric constraint solver. Only one very simple type of defined constraint is specified in this schema, the **equal_parameter_constraint** (see 5.4.3). However, clause 7 of this part of ISO 10303 specifies many constraints of the defined type.

5.2.2 Simultaneous groups of constraints

In some models, groups of explicit constraints may be required to hold simultaneously. An entity data type **simultaneous_constraint_group** is provided in this schema for the representation of such groups. A rule is provided to ensure that a given **explicit_constraint** cannot belong to more than one **simultaneous_constraint_group**; otherwise, all the sets containing the constraint in question could be assembled into a larger simultaneous set.

NOTE The sequential application of constraints is outside the scope of this part of ISO 10303. Constraint instances in a populated schema that do not belong to a **simultaneous_constraint_group** may have been applied sequentially in the originating system, but such time-dependent model aspects must be captured through the use of the procedural representation resource ISO 10303-55.

Hierarchical structuring of constraint groups is permitted, by allowing instances of **simultaneous_constraint_group** to be members of other instances of the same type.

5.2.3 Use of the current result in the resolution of ambiguities

CAD systems implementing simultaneous constraint sets must be capable of solving systems of simultaneous equations to determine the parameter values and other details of constrained geometric configurations. Often the equations concerned are nonlinear, and this leads to the possibility of multiple solutions. However, all CAD systems at present generate a single fully explicit model, which implies that the designer (or the system) has made specific choices of constraint solutions where multiple possibilities have arisen. The transfer of the designer's explicit model (referred to as the 'current result'), together with the parameterized model containing the constraint system, may serve to indicate the particular choices of constraint solutions made in the originating system. The association of a current result with its controlling parameter and constraint information constitutes a **variational_representation**, as defined in in clause 6 of this part of ISO 10303.

NOTE It should be reiterated that the methods used for solving systems of constraint equations are outside the scope of ISO 10303. The intention of the standard is limited to the transmission of information necessary to construct such a system of equations in a receiving system.

5.2.4 Directed and undirected constraints

Two types of explicit constraint may be distinguished. A *directed* constraint asserts a relationship between the items in a set of constrained elements and one or more specified *reference elements*. An *undirected* constraint has no reference element; it asserts a relationship between all pairs of items in the set of constrained elements, which in this case usually has only two or three members. Provision is made in this part of ISO 10303 to handle both directed and undirected constraints.

NOTE The terms *symmetric* and *asymmetric* are often used for undirected and directed constraints, respectively.

EXAMPLE 1 A parallelism constraint specifying that a set of lines is parallel to a given line is directed; the given line is the reference element. On the other hand, a constraint specifying that three planes are mutually perpendicular is undirected; there is no reference element, and the relationship exists between all pairs of members of the set of constrained items.

EXAMPLE 2 The case where a line is constrained to be tangent to two fixed circles is an example of the use of *multiple* reference elements (i.e., the two circles).

5.2.5 Roles of model parameters in free-form constraints

There are important differences between the semantics of directed and undirected free-form constraints, and these are highlighted by the intended behaviour of participating **model_parameter** instances, as explained below.

A **model_parameter** may play either of two roles in a directed **free_form_constraint**:

reference element: In this case the value of the parameter is regarded as an input when the constraint expression is evaluated;

constrained element: Here the value of the parameter is regarded as an output, being determined in terms of the values associated with the input or reference elements.

Thus, if a directed constraint holds the configuration may only be modified, following a model transfer, by editing reference elements. This should cause consistent modifications to all the constrained elements so that the constraint continues to be satisfied. No direct editing of constrained elements is permitted in this case.

In the undirected case, where no reference element is present, the constrained elements play a dual role — any one of them may be edited following a model transfer, and all the rest should change accordingly, so as to ensure the continued satisfaction of the constraint.

EXAMPLE The relation $L * W = A$ gives the area A of a rectangle in terms of its length L and width W. The use of this relation in a constraint may be restricted by specifying L and W as reference parameters and A as a constrained parameter. With this distinction, it should be impossible to edit the value of A directly following a model transfer, since that might contravene the design intent.

NOTE It should be noted that if a **model_parameter** is used as a constrained element in Constraint A and a reference element in Constraint B there is an implied ordering of these constraints; A must be solved before B. More complex hierarchies are clearly possible. However, the *explicit* representation of the temporal ordering of constraint evaluation is outside the scope of this part of ISO 10303.

Two subtypes of **free_form_constraint** are defined in this schema. The first, the **free_form_assignment**, is always a directed constraint. An expression involving reference elements is evaluated and the resulting

value assigned to a constrained element. The second subtype is the**free_form_relation**, which in its most general form allows the representation of hybrid constraints, having an undirected relationship amongst a set of constrained elements, where the details of this relationship are governed by one or more reference elements. Example 1 in 5.4.6 provides an illustration of such a hybrid case.

5.2.6 Accuracy of constraint satisfaction

The accuracy of constraint satisfaction in an ISO 10303 information model depends upon certain characteristics of the system where it was generated. This is also true of other aspects of model accuracy, particularly in the case of shape models. The level of agreement between the topology and geometry of a boundary representation model as defined in ISO 10303-42 provides an example. In this context, topological information might specify that two edges connect, but the geometrical information may give slightly different positions for their end-points, though close enough for them to be regarded as coincident by the internal standards of the system. Such discrepancies typically arise from computational errors incurred in geometric calculations.

Analogously, a shape model may contain a constraint asserting that two lines are perpendicular, when the lines as actually represented meet at a fraction of a degree less than$90°$. Errors in geometry/topology agreement and in constraint satisfaction are therefore very similar in nature.

No specialized means is provided in this schema for handling the problem of accuracy of constraint satisfaction in exchanged models. As indicated above, the problem is pervasive, particularly in geometric or shape modelling. The general capabilities provided elsewhere in ISO 10303 for addressing model accuracy may be used in application protocols making use of this part of the standard.

5.3 Explicit constraint type definitions

5.3.1 constraint_group_member

The **constraint_group_member** type allows a selection between entity data types that are valid members of an instance of **simultaneous_constraint_group** as defined in clause 5.4.7.

EXPRESS specification:

```
*)
TYPE constraint_group_member = SELECT
  (explicit_constraint,
   simultaneous_constraint_group);
END_TYPE;
(*
```

5.4 Explicit constraint entity definitions

5.4.1 explicit_constraint

The **explicit_constraint** entity data type is a type of **variational_representation_item**. It asserts a relationship between model elements that the receiving system is expected to maintain when the model is modified. This entity data type is the generic supertype of all explicit constraints. Such constraints may occur in either of two forms:

directed: in which all members of a set or list of constrained elements are constrained with respect to one or more reference elements;

undirected: in which there is no reference element and the constraint is required to hold between all possible pairs of a set of constrained elements.

A further distinction, as mentioned in 5.2.1, is that between a defined and a free-form constraint. Both are provided as subtypes of **explicit_constraint**.

EXPRESS specification:

```
*)
ENTITY explicit_constraint
  ABSTRACT SUPERTYPE OF (ONEOF(defined_constraint,
                               free_form_constraint))
  SUBTYPE OF (variational_representation_item);
  description : OPTIONAL text;
  constrained_elements : SET[1:?] OF representation_item;
  reference_elements   : SET[0:?] OF representation_item;
WHERE
  WR1: SIZEOF(constrained_elements * reference_elements) = 0;
END_ENTITY;
(*
```

Attribute definitions:

description: An optional textual description of the semantics of the constraint instance.

constrained_elements: The set of constrained elements.

reference_elements: The set of reference elements upon which the constrained elements are dependent.

Formal propositions:

WR1: There shall be no members in common between the set of **reference_elements**, if any are defined, and the set of **constrained_elements**.

Informal propositions:

IP1: Every **explicit_constraint** in an ISO 10303 model shall be satisfied at the time of model transfer, as judged by the internal numerical accuracy criteria of the originating system.

5.4.2 defined_constraint

The **defined_constraint** entity data type is a type of **explicit_constraint**, and the abstract supertype of a class of constraints defined in descriptive terms, with no specification of a precise mathematical relationship between parameters and entity data type instance attributes. The semantics of a **defined_constraint** instance shall be understood by any system interpreting it, which will then formulate the constraint mathematically in whatever manner is convenient for its own purposes.

NOTE By contrast, a free-form constraint is formulated in terms of a precise mathematical relationship between parameters and entity data type instance attributes. In this case the relationship as represented in the originating system shall be reproduced exactly in the receiving system.

EXAMPLE The constraints defined in clause 7 are all of the defined type, asserting various types of geometric relationships. To give just one illustration, the **parallel_geometric_constraint** of clause 7 may be used to constrain two lines to be parallel. This is a descriptive specification of the constraint. An example of a precise mathematical representation of the relationship between the two lines is the requirement that the cross-product of their direction vectors be zero, expressed in terms of equations involving the components of those vectors. The constraint may or may not actually be formulated in this form for solution by the constraint solver in a particular CAD system.

EXPRESS specification:

```
*)
ENTITY defined_constraint
  ABSTRACT SUPERTYPE OF (equal_parameter_constraint)
  SUBTYPE OF (explicit_constraint);
END_ENTITY;
(*
```

5.4.3 equal_parameter_constraint

The **equal_parameter_constraint** entity data type is a type of **defined_constraint** that constrains a set of **model_parameter** instances to have equal value attributes. There are directed and undirected forms:

directed: All the constrained elements are required to have the same value as a single reference element;

undirected: There is no reference element, and all the constrained elements are required to share a common value.

EXAMPLE 1 It may be desired to constrain all constant radius blends in a geometric model to have the same radius. In this case, a **model_parameter** may be associated with each blend radius and the undirected form of **equal_parameter_constraint** used to assert the equality of value of all these parameters.

NOTE The **free_form_relation** constraint, defined in 5.4.6 allows a the value of a parameter to be constrained through its participation in a mathematical relationship. The directed form of the **equal_parameter_constraint** can achieve the effect of constraining a *set* of parameters according to a relationship in which only one of them participates, by using as its reference element the parameter controlled by the **free_form_relation**.

EXAMPLE 2 Suppose that **model_parameter** instances corresponding to variables x, y, z are constrained by a **free_form_relation** constraint to satisfy the relationship $x^2 + y^2 = z^2$. If it is additionally desired that parameters corresponding to variables q, r, s always have values equal to that of z, then a directed **equal_parameter_constraint** may be used, in which the set of **constrained_elements** contains the **model_parameter** instances corresponding to q, r, s, while z is used as a **reference_element**.

The **equal_parameter_constraint** is an elementary example of the **defined_constraint** class. It asserts that the values of a set of parameters are equal, but the relationship is expressed descriptively rather than in explicit mathematical terms as would be the case in a **free_form_constraint**.

EXPRESS specification:

```
*)
ENTITY equal_parameter_constraint
  SUBTYPE OF (defined_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF model_parameter;
  SELF\explicit_constraint.reference_elements   :
    SET[0:1] OF model_parameter;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.constrained_elements +
    SELF\explicit_constraint.reference_elements) >= 2;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of **model_parameter** instances whose values are constrained to be equal.

SELF\explicit_constraint.reference_elements: A set of zero or one **model_parameter** instances; if a reference element is present its value is assigned to all the constrained elements.

Formal propositions:

WR1: The total number of **model_parameter** instances involved in an **equal_parameter_constraint** shall not be less than two.

5.4.4 free_form_constraint

The **free_form_constraint** entity data type is a type of **explicit_constraint** defining a special purpose constraint that cannot be modelled with the defined constraint entities available in any particular application context.

As with other explicit constraints, the constrained elements are those controlled by the constraint, and the reference elements specify, for the case of directed constraints, elements upon which they are constrained to be dependent. The dependency generally involves relations between values of attributes of entity data type instances, and the types of the constrained and reference elements are therefore both specified as **model_parameter**. Two specialized subtypes of this constraint are defined below.

EXPRESS specification:

```
*)
ENTITY free_form_constraint
  ABSTRACT SUPERTYPE OF (ONEOF(free_form_assignment, free_form_relation))
  SUBTYPE OF (explicit_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF model_parameter;
  SELF\explicit_constraint.reference_elements   :
    SET[0:?] OF model_parameter;
```

```
  constraining_expression : expression;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of constrained instances of **model_parameter**.

SELF\explicit_constraint.reference_elements: The set of **model_parameter** instances used as reference elements.

constraining_expression: The **expression** used to define a relationship between the **model_parameter** instances involved in the constraint.

NOTE 1 The **expression** entity data type from ISO 13584-20 allows the modelling of mathematical expressions whose values are of types NUMBER, BOOLEAN or STRING.

NOTE 2 Any application protocol making use of this schema will need to USE a set of domains from the ISO 10303-50 **mathematical_functions_schema** that is appropriate for the **model_parameter** value types employed.

5.4.5 free_form_assignment

The **free_form_assignment** entity data type is a type of **free_form_constraint** that represents the assignment of a value, the current value of a mathematical expression involving **model_parameter** instances, to one or more specified **model_parameter** instances. The value of the expression in the exchange model is obtained by evaluating it with the current values of all parameters involved in it. In the context of this entity data type, **model_parameter** instances involved in the expression are regarded as reference elements. The value of the expression is assigned to the value attribute of one or more constrained **model_parameter** instances.

EXAMPLE Suppose that real-valued **model_parameter** instances are associated with variables t_1, t_2, t_3, x_1, x_2, where the parameters corresponding to t_1, t_2, t_3 are constrained elements, and those corresponding to x_1, x_2 are reference elements. Then the reference parameters may be used in the **constraining_expression** representation of the expression $\sin^2 x_1 + \sin^2 x_2$, and the resulting value of the expression, evaluated for the current values of those parameters, assigned to the parameters in the set of **constrained_elements**, corresponding to variables t_1, t_2, t_3. Thus the three constrained parameters are all constrained to have the value of the expression involving the two reference parameters.

NOTE 1 A **free_form_assignment** is always a directed constraint. In editing a model containing such a constraint, only the values of the reference elements may be changed by the system user — the values of the constrained elements are then determined by the system in accordance with the new set of reference values.

EXPRESS specification:

```
*)
ENTITY free_form_assignment
  SUBTYPE OF (free_form_constraint);
WHERE
  WR1: SIZEOF(QUERY(q <* SELF\free_form_constraint.constrained_elements |
    q IN used_variables
    (SELF\free_form_constraint.constraining_expression))) = 0;
  WR2: SIZEOF(QUERY(q <* SELF\free_form_constraint.reference_elements |
```



```
    NOT (q IN used_variables(
    SELF\free_form_constraint.constraining_expression)))) = 0;
  WR3: SIZEOF(SELF\free_form_constraint.reference_elements) >= 1;
  WR4: SIZEOF(QUERY(q <* SELF\free_form_constraint.constrained_elements |
    NOT (compatible_spaces(values_space_of(
    SELF\free_form_constraint.constraining_expression),
    q\maths_variable.values_space)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: No member of the set of **SELF\explicit_constraint.constrained_elements** shall occur in the **SELF\free_form_constraint.constraining_expression**.

WR2: Every member of the set of **SELF\free_form_constraint.reference_elements** shall occur in the **SELF\free_form_constraint.constraining_expression**.

WR3: The set of **SELF\free_form_constraint.reference_elements** shall contain at least one member — otherwise the **constraining_expression** has a constant value and the use of this constraint is inappropriate.

WR4: The possible range of values of the **constraining_expression** shall be compatible with the specified domains of values of all the constrained **model_parameter** instances.

NOTE 2 The compatibility referred to in the description of **WR4** requires that the specified domain of each constrained instance of **model_parameter** has a non-null intersection with the values space of the **constraining_expression**. This ensures type compatibility of the values concerned.

NOTE 3 An example of the use of **free_form_assignment** is given in Example 2 of clause F.1.

5.4.6 free_form_relation

The **free_form_relation** entity data type is a type of **free_form_constraint** representing a BOOLEAN-valued relation between the values of **model_parameter** instances. The constraint is satisfied if the relation has the value TRUE when evaluated with current values of all the parameters involved at the time of model exchange. Within the scope of this type of constraint, **model_parameter** instances listed as reference elements are regarded as constants used in determining values of the**constrained_elements**.

The use of this constraint to control *sets* of **model_parameter** instances to have the same associated value requires the **equal_parameter_constraint** of 5.4.3 to be employed.

NOTE 1 A **free_form_relation** may represent a directed constraint, an undirected constraint or some combination of the two (hybrid constraint). If only one constrained element is present, the constraint will always be directed. If no reference element is present the constraint will be undirected. However, if more than one constrained element is present, together with one or more reference elements, some freedom will generally remain in the determination of the values of the constrained elements after satisfaction of the constraint relation. In such a case, what remains is an undirected relationship in which one or more of the constrained element values may be changed and the others should adjust accordingly. The following example illustrates such a situation.

EXAMPLE 1 Three real-valued **model_parameter** instances correspond to variables x, y and z. A **free_form_relation** instance is defined in which the parameters corresponding to x and y are constrained elements, and that corresponding to z is a reference element. An **expression** in this instance, corresponding to the relation $\sqrt{(x^2 + y^2)} < z^2 + 3.0$, specifies the relationship between the three parameters.

In geometric terms, if x, y, z represent cartesian coordinates, the constraint requires x and y to lie inside a circle of radius $z^2 + 3$. The set of all such points form the interior of the solid resulting from rotation of the parabola $x = z^2 + 3$ about the z-axis.

This example illustrates a hybrid constraint, directed with respect to the **model_parameter** representing z but undirected as regards the relationship between x and y.

The behaviour is clear following modification of one constrained element — the other constrained element changes accordingly. However, there is no uniquely defined behaviour when the reference element is modified. In such cases a fully constrained situation may be achieved through the presence of other constraints or through user interaction following transfer of the model.

EXPRESS specification:

```
*)
ENTITY free_form_relation
  SUBTYPE OF (free_form_constraint);
WHERE
  WR1: 'ISO13584_EXPRESSIONS_SCHEMA.BOOLEAN_EXPRESSION' IN TYPEOF
    (SELF\free_form_constraint.constraining_expression);
  WR2: SIZEOF(QUERY(q <* (SELF\free_form_constraint.constrained_elements +
    SELF\free_form_constraint.reference_elements) |
    NOT (q IN (SELF\free_form_constraint.constraining_expression)))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The **SELF\free_form_constraint.constraining_expression** attribute shall be of type **boolean_expression** as defined in ISO 13584-20.

WR2: All **model_parameter** instances belonging to the sets **SELF\explicit_constraint.constrained_elements** and **SELF\explicit_constraint.reference_elements** shall participate in **SELF\free_form_constraint.constraining_expression**.

NOTE 2 A **boolean_expression** is defined in ISO 13584-20 to be an expression whose range is the BOOLEAN data type defined in ISO 10303-11 (i.e., the expression evaluates to TRUE or FALSE).

NOTE 3 A single **free_form_relation** instance can capture both aspects of a reciprocal dimensional relationship. Assume that two **bound_model_parameter** instances have been defined and associated with the radius attributes of two different **circle** instances, whose values will be denoted here by r_1 and r_2. Suppose now that that the radius of one of the circles must always be twice that of the other. This relationship may be modelled through the use of a constraint relation corresponding to $r_1^2 = 2.0 * r_2^2$. If both model parameters are specified as constrained elements then this is an undirected constraint defining a reciprocal relationship. It will ensure the desired behaviour in the receiving system, whichever of the two radii is modified.

An example of this application of the **free_form_relation** is given in Example 1 of clause F.1.

5.4.7 simultaneous_constraint_group

The **simultaneous_constraint_group** entity data type is a type of **variational_representation_item**, defining a set of constraints that are required to hold simultaneously. Such constraint sets may have multiple solutions. Instances of **simultaneous_constraint_group** are also permitted to be members of

the set, and this allows recursive structuring of constraint groups. The embedding of instances of **simultaneous_constraint_group** may reflect the sequence of stages of model creation.

EXAMPLE 1 A collection of constraints applying to a two-dimensional sketch may be represented by an instance of **simultaneous_constraint_group**. The sketch may then be used as the basis for the creation of a three-dimensional object participating in an assembly. A group of assembly constraints may be used to position and orient the component in the assembly, and these form a constraint group at a higher level. However, the lower level sketch constraints are still present in the overall definition, and if a parametric change is made at that level they will need to be solved first, and the solution then taken into account at the higher level. For this application it is appropriate to embed the lower level sketch constraint group in the higher-level assembly constraint group.

NOTE The time-dependent sequencing of constraints is outside the scope of this part of ISO 10303, except insofar as it is implied by the embedding of constraint groups within each other.

EXPRESS specification:

```
*)
ENTITY simultaneous_constraint_group
  SUBTYPE OF (variational_representation_item);
  constraint_group : SET[2:?] OF constraint_group_member;
WHERE
  WR1: SIZEOF(QUERY(q <* using_representations(SELF) |
    SIZEOF(QUERY(r <* q.items |
    ('EXPLICIT_CONSTRAINT_SCHEMA.SIMULTANEOUS_CONSTRAINT_GROUP'
    IN TYPEOF(r)) AND (SIZEOF(QUERY(s <* constraint_group |
    (s IN r.constraint_group) AND NOT (r :=: SELF))) > 0))) > 0)) = 0;
  WR2: SIZEOF(QUERY(q <* using_representations(constraint_group[1]) |
    (SIZEOF(QUERY(r <* constraint_group |
    item_in_context(r,q.context_of_items)))
    = SIZEOF(constraint_group)))) > 0;
  WR3: SIZEOF(QUERY(q <* constraint_group |
    (('EXPLICIT_CONSTRAINT_SCHEMA.EXPLICIT_CONSTRAINT' IN TYPEOF(q))
    AND (SIZEOF(QUERY(r <* q.constrained_elements |
    SIZEOF(QUERY(s <* constraint_group |
    r IN s.reference_elements)) > 0)) > 0)))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

constraint_group: A set with members of type **explicit_constraint** or **simultaneous_constraint_group**.

Formal propositions:

WR1: Any constraint in the group shall be a member of no other **simultaneous_constraint_group**.

WR2: There shall exist at least one **representation_context** that contains all members of the group of explicit constraints (including members of embedded groups).

WR3: No instance that serves as a constrained element in any one constraint in the group shall serve as a reference element in any other. This restriction shall apply at each level in the constraint hierarchy, but not between the levels defined by embedding of constraint groups.

NOTE 1 In the absence of WR1, a given instance of **explicit_constraint** could belong to two or more instances of **simultaneous_constraint_group**. This would imply that all the constraint groups containing that constraint could be assembled into a single, larger, constraint group. WR1 ensures that all such constraint groups are maximal, and that such overlaps in membership cannot occur. Additionally, it ensures that a constraint that is a member of an embedded group cannot also be a member at any other level in a constraint hierarchy.

NOTE 2 An important consequence of WR2 concerns the dimensionality of elements in shape models. The geometric subtypes of **explicit_constraint** defined in clause 7 of this part of ISO 10303 are also subtypes of **geometric_representation_item** (see ISO 10303-42), and have an associated dimensionality. If one or more explicit constraints in the group are **geometric_representation_item** instances, then any common **representation_context** shared by the members must be a **geometric_representation_context** — this is required by a local rule in the definition of **geometric_representation_item**. Then WR2 ensures that all geometric constraints in an instance of **simultaneous_constraint_group** necessarily have the same dimensionality, consistent with the dimensionality of their shared **geometric_representation_context**. The fact that WR2 applies equally to embedded constraint groups ensures that complete hierarchies of geometric constraints are dimensionally consistent.

NOTE 3 In the absence of **WR3** a situation could arise such as the following:

— A **model_parameter** with name `'x'` is a reference element in a free-form constraint F, which has a **model_parameter** with name `'y'` as a constrained element;

— Conversely, free-form constraint G uses the **model_parameter** named `'y'` as a reference element and that named `'x'` as a constrained element.

Here constraint F forbids the value associated with `'y'` to be changed to achieve constraint satisfaction, while constraint G similarly forbids any change in the value associated with `'x'`. Hence the value attributes of both **model_parameter** instances must remain fixed within the context of the **simultaneous_constraint_group**. This and similar situations are avoided by requiring that no instance shall play the role of both reference and constrained element in constraints occurring at the same level in the group. Thus any instance involved in constraints at a given level of the constraint hierarchy must play the role of either reference element or constrained element to the group as a whole at that level, but not both.

The situation is different in the following cases:

— It is permitted that a constrained element from an embedded constraint group at a lower level shall be a reference element for a constraint group at a higher level;

— In a sequentially imposed set of constraints, a constrained element from one constraint may play the role of a reference element in a succeeding constraint.

However, the representation of time-dependent sequences of constraints is out of scope of this part of ISO 10303.

EXPRESS specification:

```
*)
END_SCHEMA; -- explicit_constraint_schema
(*
```

6 Variational representation

6.1 Introduction

The following EXPRESS declaration begins the variational representation schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA variational_representation_schema;

REFERENCE FROM geometry_schema                          -- ISO 10303-42
  (geometric_representation_item);

REFERENCE FROM representation_schema                    -- ISO 10303-43
  (representation,
   representation_item,
   representation_relationship,
   using_representations);

REFERENCE FROM parameterization_schema                  -- ISO 10303-108
  (bound_model_parameter,
   fixed_instance_attribute_set,
   instance_attribute_reference,
   unbound_model_parameter);

REFERENCE FROM explicit_constraint_schema               -- ISO 10303-108
  (explicit_constraint,
   free_form_constraint);

REFERENCE FROM explicit_geometric_constraint_schema     -- ISO 10303-108
  (explicit_geometric_constraint);

REFERENCE FROM ISO13584_generic_expressions_schema      -- ISO 13584-20
  (generic_variable);
(*
```

NOTE 1 The schemas referenced above can be found in the following Parts of ISO 10303:

`geometry_schema`	ISO 10303-42
`representation_schema`	ISO 10303-43
`parameterization_schema`	clause 4 of this part of ISO 10303
`explicit_constraint_schema`	clause 5 of this part of ISO 10303
`explicit_geometric_constraint_schema`	clause 7 of this part of ISO 10303
`ISO13584_generic_expressions_schema`	ISO 13584-20

NOTE 2 See annex D, Figure D.4, for a graphical presentation of this schema.

6.2 Fundamental concepts and assumptions

This schema provides the entity data type **variational_representation** for the representation of a variational model, characterized by the presence of explicitly represented parameters and constraints. Such a model may be considered to represent a family of related non-variational models. A ‘current result’

is associated with a variational model; it is that member of the represented family corresponding to the current values of all the parameters. Different members of the family can be derived by variation of those parameter values, subject to the imposed constraints.

NOTE 1 It is important to distinguish between explicitly represented parameters and constraints and the implicit parameters and constraints inherent in procedural models. In a procedural model, parameters occur as input arguments of constructional operations, and constraints are inherent in the nature of those operations.

The relationship between an instance of **variational_representation** and its current result is that the second is entirely contained within the first. Alternatively, the information regarding parameters and constraints may be considered to constitute a wrapper surrounding the current result, which may take either of two forms:

a) An explicit model represented in terms of its constituent elements;

b) A procedural or construction history model represented in terms of the operations used to construct it. Such representations are defined in ISO 10303-55 (see also annex E of this part of ISO 10303). In most such cases the model will be of the hybrid type in which some constructional operations are performed on explicitly defined elements such as the sketches specified in clause 8 of this part of ISO 10303, which may have associated variational information. In other cases the constructional operations will be defined in terms of explicit supporting elements such as points and directions, and variational relationships may be imposed between such elements.

A new abstract subtype of the ISO 10303-43 entity data type **representation_item** has been introduced to distinguish between variational and non-variational data, namely **variational_representation_item**, as defined in clause 6.3.1 of this schema. Its instantiable subtypes include **model_parameter**, **explicit_constraint** and other related entities. In an instance of **variational_representation**, all instances of **representation_item** occurring in the wrapper are required also to be of type **variational_representation_item**. Conversely, no instance of **representation_item** used directly or indirectly by the current result is permitted to have the type **variational_representation_item**. This makes a clear distinction between the non-variational current result model being transferred and the variational information used to control its behaviour if it is edited in the receiving system.

NOTE The structure of a **variational_representation** is shown diagrammatically in Figure 3.

The relationship between a variational representation and its embedded current result, both of which shall be separately instantiated, is captured by the entity data type **variational_current_representation_relationship** (see clause 6.3.4), a subtype of **representation_relationship** as defined in ISO 10303-43.

The additional entity data type **auxiliary_geometric_representation_item** is defined in clause 6.3.2 for the representation of geometric elements that are used as reference elements in constraints in the variational representation but that do not belong to the current result.

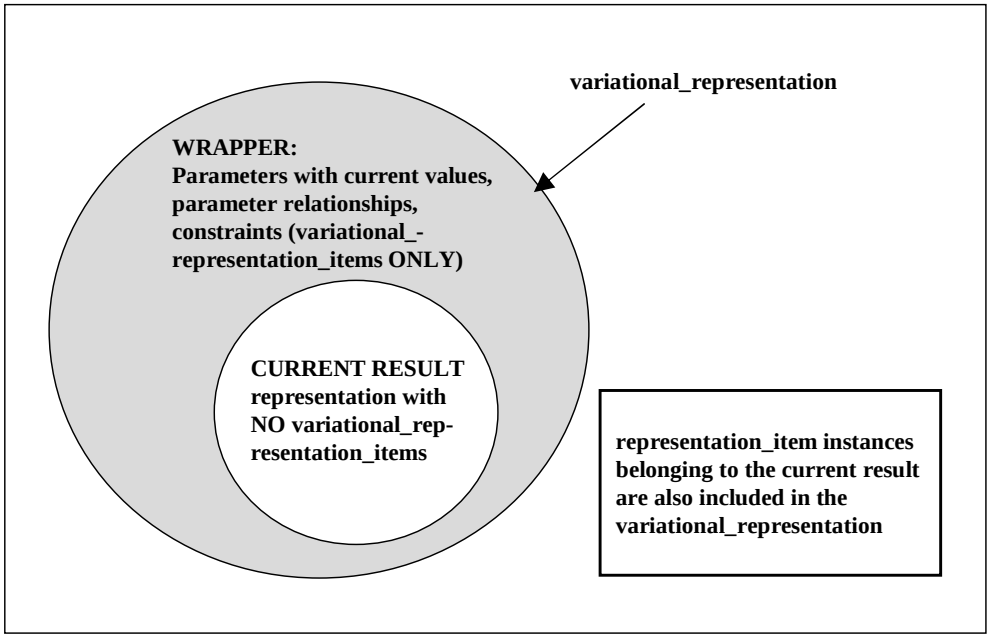

Figure 3 – Embedding of a current result representation in a variational_representation

6.3 Variational representation entity definitions

6.3.1 variational_representation_item

The abstract entity data type **variational_representation_item** is a type of **representation_item**. It defines an element of a representation that does not affect the static characteristics of a transferred model at the time of transfer, but that has the potential to control its behaviour when the model is edited in a receiving system following a transfer.

EXAMPLE Instantiable subtypes of **model_parameter** as defined in clause 4 and **explicit_constraint** as defined in clause 5 are examples of **variational_representation_item**.

No instance of **variational_representation_item** shall occur in any representation that is not of type **variational_representation**. Every instance of **variational_representation_item** used by an instance of **variational_representation** shall be a member of the set of **items** of that representation.

EXPRESS specification:

```
*)
ENTITY variational_representation_item
  ABSTRACT SUPERTYPE OF (auxiliary_geometric_representation_item)
  SUBTYPE OF (representation_item);
WHERE
  WR1: SIZEOF(QUERY(q <* using_representations(SELF) |
```

```
    NOT ('VARIATIONAL_REPRESENTATION_SCHEMA.VARIATIONAL_REPRESENTATION'
    IN TYPEOF(q)))) = 0;
  WR2: SIZEOF(QUERY(q <* using_representations(SELF) |
    NOT (SELF IN q.items))) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: No instance of **variational_representation_item** shall occur in any representation that is not of type **variational_representation**.

WR2: Every instance of **representation** that uses a given instance of **variational_representation_item** shall reference it directly, i.e., shall include it as a member of its **items** attribute.

6.3.2 auxiliary_geometric_representation_item

The entity data type **auxiliary_geometric_representation_item** is a type of **geometric_representation_item** and also a type of **variational_representation_item**. It provides a representation for geometric elements that exist in a **variational_representation** for use as reference elements in constraints but are not part of the current representation.

EXAMPLE A dimensional constraint may specify the half-width of a rectangular-section slot as the distance of each wall from the mid-plane of the slot. In this case the plane concerned is not part of the geometry of the slot, or of the part it exists on, but is used as a reference element in the dimensional constraint. The current result contains details of the part geometry, but not the specification of the slot mid-plane. That is provided in the variational wrapper as an instance of **auxiliary_geometric_representation_item**, together with the dimensional constraint using it as a reference element.

NOTE The imported points and curves defined in clause 8 of this part of ISO 10303 provide further examples of **auxiliary_geometric_representation_item**.

EXPRESS specification:

```
*)
ENTITY auxiliary_geometric_representation_item
  SUBTYPE OF (geometric_representation_item,
              variational_representation_item);
END_ENTITY;
(*
```

6.3.3 variational_representation

The **variational_representation** entity data type defines a type of representation containing parameterization and constraint information that may be used to edit the model, following a transfer, in a manner consistent with the designer's original intent.

EXAMPLE 1 A model of a table has a parameter named L associated with its length. It is required that the maximum unsupported length between pairs of legs is 1.5 metres. Thus the number of legs supporting the table top will depend upon the value of L. A model is transferred for which L has the value 1.2 metres and the table has four legs — this is the 'current result'. Associated with the current result is the definition of the parameter L and the mathematical relation involving L that is used to determine the number and (equal) spacing of pairs of

legs. The model as received following a transfer is that of the current result. However, the associated information allows this model to be edited in the receiving system by varying L, in which case the number of legs should adjust automatically, and the current result will change accordingly. In both the initial and the final state the **variational_representation** includes the current result model together with variational information permitting it to be edited intelligently.

EXPRESS specification:

```
*)
ENTITY variational_representation
  SUBTYPE OF (representation);
INVERSE
  cm_link : variational_current_representation_relationship FOR rep_1;
WHERE
  WR1: SIZEOF(QUERY(q <* SELF\representation.items |
    'VARIATIONAL_REPRESENTATION_SCHEMA.VARIATIONAL_REPRESENTATION_ITEM'
    IN TYPEOF(q))) > 0;
  WR2: SIZEOF(QUERY(q <* (SELF\representation.items -
    cm_link.rep_2.items) | invalidate_vrep_item(q))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

cm_link: The instance of **variational_current_representation_relationship** linking a **variational_representation** instance with its associated current result.

Formal propositions:

WR1: At least one instance of **representation_item** referenced by a **variational_representation** shall be of type **variational_representation_item**.

WR2: Every instance of **variational_representation_item** occurring in a variational representation shall be valid in the sense specified in clause 6.4.1, where the function **invalidate_vrep_item** is defined.

NOTE 1 A WHERE rule of the entity **variational_representation_item** (see clause 6.3.1) requires that any instance of **variational_representation_item** occurring in an instance of **variational_representation** is directly referenced as a member of the **items** attribute of that representation.

NOTE 2 In the case of a shape model the current result may include instances of **geometric_representation_item**. In that case a WHERE rule of **geometric_representation_item** requires the associated **representation_context** shared by the **variational_representation** and its embedded current result to be of type **geometric_representation_context**. These entity data types are defined in ISO 10303-42.

EXAMPLE 2 An instance of **neutral_sketch_representation** (see clause 8.4.9) defines a rectangle with length 12 units and width 7 units, in terms of low-level geometric elements. This is the 'current result'. Associated with the rectangle are instances of **pgc_with_dimension** (see clause 7.4.4) enforcing parallelism between opposite pairs of sides of the rectangle, and **perpendicular_geometric_constraint** (see clause 7.4.21) relating a pair of adjacent sides. Further, instances of **bound_model_parameter** (see clause 4.4.2) are bound to the **displacement_value** attributes of the two parallelism constraints. These parameter and constraint instances constitute the variational information associated with the sketch. The sketch as received after a transfer corresponds to the current result. However, the associated variational information ensures that if the sketch is edited following the transfer, by vari-

ation of the values of the model parameters, it retains a rectangular form. The current result and the variational information together comprise an instance of **variational_shape_representation**. Because the current result contains explicit geometric elements and is two-dimensional, both it and its containing **variational_representation** share a **geometric_representation_context** whose **coordinate_space_dimension** attribute has the value 2.

6.3.4 variational_current_representation_relationship

The **variational_current_representation_relationship** entity data type is a type of **representation_relationship** that defines the relationship between a **variational_representation** and its embedded non-variational 'current result' representation.

EXPRESS specification:

```
*)
ENTITY variational_current_representation_relationship
  SUBTYPE OF (representation_relationship);
  SELF\representation_relationship.rep_1
                : variational_representation;
  current_result : representation;
UNIQUE
  UR1: current_result;
WHERE
  WR1: QUERY(q <* SELF\representation_relationship.rep_1.items | NOT
    ('VARIATIONAL_REPRESENTATION_SCHEMA.VARIATIONAL_REPRESENTATION_ITEM'
    IN TYPEOF(q))) = SELF\representation_relationship.rep_2.items;
  WR2: SELF\representation_relationship.rep_1.context_of_items :=:
    SELF\representation_relationship.rep_2.context_of_items;
  WR3: SIZEOF(QUERY(q <* SELF\representation_relationship.rep_2.items |
    'VARIATIONAL_REPRESENTATION_SCHEMA.VARIATIONAL_REPRESENTATION_ITEM'
    IN TYPEOF(q))) = 0;
  WR4: TYPEOF(SELF\representation_relationship.rep_1) -
       TYPEOF(SELF\representation_relationship.rep_2) =
       ['VARIATIONAL_REPRESENTATION_SCHEMA.VARIATIONAL_REPRESENTATION'];
  WR5: current_result :=: SELF\representation_relationship.rep_2;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_relationship.rep_1: The **variational_representation**.

SELF\representation_relationship.rep_2: The embedded current result representation.

current_result: The representation defining the current result model.

Formal propositions:

UR1: The **representation** instance referenced by the attribute **current_result** shall not occur as the current result of any other instance of **variational_current_representation_relationship**.

WR1: The set of instances of **representation_item** that are directly referenced by the current result representation shall be identical with the set of non-variational instances of **representation_item** that are directly referenced by the containing **variational_representation**.

WR2: The related representations shall share the same **representation_context**.

WR3: The instances of **representation_item** that are directly referenced by the current result **representation** shall contain no instances of **variational_representation_item**.

WR4: The type list of the variational representation shall be identical to that of the related current result except that it shall also include **variational_representation**.

WR5: The unique **representation** instance referenced by the attribute **current_result** shall be identical with the representation referenced by the attribute **rep_2** of the supertype **representation_relationship**.

NOTE 1 The combination of UR1 and WR5 ensure that no two variational representations share the same current result model. This is appropriate because the current result is a representative example of the class of models defined by the **variational_representation**, and in particular it is determined by the current values of all parameters in that representation. It is unlikely that any two different instances of **variational_representation** will yield identical current results, and even if this were to happen it would probably be by chance rather than by design.

NOTE 2 This part of ISO 10303 makes no provision for the specification of relationships between

— different instances of **variational_representation** derived from an original instance of **variational_representation** by modification of parameter values, i.e., between different members of the same part family;

— instances of **variational_representation** in which different current results have resulted from identical variational information but different choices of constraint solutions.

Such relationships may be defined in other parts of ISO 10303 that use or specialize entities from this schema.

6.4 Variational representation function definitions

6.4.1 invalidate_vrep_item

The **invalidate_vrep_item** function determines whether an instance of **variational_representation_item** participating in a **variational_representation** is invalid. Valid cases are defined as follows:

a) A valid instance of **bound_model_parameter** shall be bound to an attribute of some instance used by all the representations in which the model parameter participates.

b) A valid instance of **unbound_model_parameter** shall participate as a reference element or a constrained element in at least one instance of **free_form_constraint** occurring in all the representations in which the model parameter participates.

c) A valid instance of **fixed_instance_attribute_set** shall reference no attribute of any instance that is not used by all the representations in which the fixed instance attribute set participates.

d) A valid instance of **explicit_constraint** shall specify no reference element or constrained element that is not an instance used by all the representations in which the explicit constraint participates.

e) A valid instance of **auxiliary_geometric_representation_item** shall participate as a reference element in at least one instance of **explicit_geometric_constraint**, for every representation in which the auxiliary geometric representation item participates.

NOTE An instance of **auxiliary_geometric_representation_item** may also participate as a *constrained* element in an instance of **explicit_geometric_constraint** in a case where, for example, it is positioned with respect to another instance of **auxiliary_geometric_representation_item**. It is therefore possible to define chains of auxiliary geometric elements in which each member except the first is constrained with respect to its predecessor.

The function tests for all the above conditions in the order given, and returns TRUE as soon as one of them is found not to be satisfied. If all are satisfied the function returns FALSE and the tested instance of **variational_representation_item** is accepted as valid.

EXPRESS specification:

```
*)
FUNCTION invalidate_vrep_item(item : variational_representation_item)
                                   : BOOLEAN;

  LOCAL
    reps    : SET[1:?] OF representation := using_representations(item);
    svri    : SET[1:?] OF variational_representation_item;
    iar     : instance_attribute_reference;
    i       : INTEGER;
    n       : INTEGER := HIINDEX(reps);
  END_LOCAL;

  IF ('PARAMETERIZATION_SCHEMA.BOUND_MODEL_PARAMETER' IN TYPEOF(item))
  THEN
    IF 'PARAMETERIZATION_SCHEMA.INSTANCE_ATTRIBUTE_REFERENCE'
      IN TYPEOF(item\generic_variable.interpretation.semantics)
    THEN
      BEGIN
        iar := item\generic_variable.interpretation.semantics;
        IF (reps <> using_representations(iar.owning_instance))
        THEN
          RETURN(TRUE);
        END_IF;
      END;
    ELSE RETURN(TRUE); -- parameter not attached to an instance attribute
    END_IF;
  END_IF;

  IF ('PARAMETERIZATION_SCHEMA.UNBOUND_MODEL_PARAMETER' IN TYPEOF(item))
  THEN
    BEGIN
      REPEAT i := 1 TO n;
        svri := QUERY(q <* reps[i].items |
          'EXPLICIT_CONSTRAINT_SCHEMA.FREE_FORM_CONSTRAINT' IN TYPEOF(q));
        IF SIZEOF(QUERY(r <* svri |
          item IN (r.reference_elements + r.constrained_elements))) = 0
        THEN
          RETURN(TRUE);
        END_IF;
      END_REPEAT;
    END;
  END_IF;

  IF ('PARAMETERIZATION_SCHEMA.FIXED_INSTANCE_ATTRIBUTE_SET'
    IN TYPEOF(item))
  THEN
    REPEAT i := 1 TO SIZEOF(item.fixed_attributes);
      IF (reps <> using_representations(item.fixed_attributes[i]))
      THEN
        RETURN(TRUE);
```



```
      END_IF;
    END_REPEAT;
  END_IF;

  IF ('EXPLICIT_CONSTRAINT_SCHEMA.EXPLICIT_CONSTRAINT' IN TYPEOF(item))
  THEN
    IF SIZEOF(QUERY(q <*
      (item.reference_elements + item.constrained_elements) |
      reps <> using_representations(q))) > 0
    THEN
      RETURN(TRUE);
    END_IF;
  END_IF;

  IF ('VARIATIONAL_REPRESENTATION_SCHEMA.
       AUXILIARY_GEOMETRIC_REPRESENTATION_ITEM' IN TYPEOF(item))
  THEN
    BEGIN
      REPEAT i := 1 TO n;
        svri := QUERY(q <* reps[i].items |
          'EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA.
           EXPLICIT_GEOMETRIC_CONSTRAINT' IN TYPEOF(q));
        IF SIZEOF(QUERY(r <* svri |
          item IN r.reference_elements)) = 0
        THEN
          RETURN(TRUE);
        END_IF;
      END_REPEAT;
    END;
  END_IF;

  RETURN(FALSE); -- no invalid cases have been found

END_FUNCTION;
(*
```

Argument definitions:

item: The instance of **variational_representation_item** to be tested for invalidity.

EXPRESS specification:

```
*)
END_SCHEMA; -- variational_representation_schema
(*
```

7 Explicit geometric constraint

7.1 Introduction

The following EXPRESS declaration begins the explicit geometric constraint schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA explicit_geometric_constraint_schema;

REFERENCE FROM measure_schema             -- ISO 10303-41
  (length_measure,
   plane_angle_measure,
   positive_length_measure);

REFERENCE FROM geometry_schema;           -- ISO 10303-42

REFERENCE FROM geometric_model_schema     -- ISO 10303-42
  (extruded_area_solid,
   extruded_face_solid,
   revolved_area_solid,
   revolved_face_solid,
   right_circular_cone,
   right_circular_cylinder,
   sphere,
   swept_area_solid,
   swept_face_solid,
   torus);

REFERENCE FROM representation_schema      -- ISO 10303-43
  (representation_item_relationship);

REFERENCE FROM explicit_constraint_schema -- ISO 10303-108
  (defined_constraint,
   explicit_constraint);
(*
```

NOTE 1 The schemas referenced above can be found in the following Parts of ISO 10303:

measure_schema	ISO 10303-41
geometry_schema	ISO 10303-42
geometric_model_schema	ISO 10303-42
representation_schema	ISO 10303-43
explicit_constraint_schema	clause 5 of this part of ISO 10303

NOTE 2 See annex D, Figures D.5 – D.14, for a graphical presentation of this schema.

7.2 Fundamental concepts and assumptions

This schema provides resources for the specification of explicit geometric constraints, including dimensional constraints, between geometric elements of a shape model. These elements are instances of sub-

types of **geometric_representation_item** (as defined in ISO 10303-42), the particular set of valid subtypes differing from one constraint type to another. In most cases they are simple geometric types such as points, curves, surfaces and their specializations. Constraints between topological elements may be defined indirectly, in terms of their associated geometry. More generally, the fundamental concepts and assumptions stated in 5.2 of the **explicit_constraint_schema** (see clause 5) apply in this schema also.

EXAMPLE 1 Examples of the relations that can be asserted between geometrical elements include parallelism of a line with a plane, tangency of two circles, and symmetry of two general surfaces with respect to a plane.

NOTE Geometric constraints may define relationships between elements that either

— belong to the product shape model;

— define auxiliary or constructional geometry, used in constructing that shape but not part of it.

An important use is the assertion of relationships between these two kinds of elements.

All the constraints defined in this schema are subtypes of **explicit_geometric_constraint** as defined in clause 7.4.1, which is itself defined as a subtype of the ISO 10303-42 entity data type **geometric_representation_item**. These constraints therefore have an associated dimensionality, which is required in any particular constraint instance to be the same as that of all the geometric elements involved.

The entity data type **explicit_geometric_constraint** is also a subtype of **defined_constraint**, as specified in clause 5.4.2. Such a constraint is specified in descriptive geometric terms, the assumption being that it is universally implemented in shape modelling systems, which have a precise understanding of its semantics. Thus when a model with explicit geometric constraints is transferred into a receiving system, the transmitted descriptive form can be reformulated in the explicit mathematical manner best suited to the nature of the receiving system's constraint solver.

EXAMPLE 2 The statement that a line is tangent to two nonconcentric circles is an example of a descriptive geometric constraint. It can be captured and transferred with a shape model using an instance of the explicit geometric constraint **tangent_geometric_constraint** specified in clause 7.4.24 of this schema. This instance simply provides references to the line and the two circles involved; some additional information can be provided to select one of the four possible cases of a line tangent to two given circles. However, in mathematical terms the line and the two circles are expressible by algebraic equations. The constraint may therefore be formulated as an explicit mathematical relationship involving the coefficients of these three equations, requiring that the distance of the line from the centre of each circle is equal to the circle's radius. The resulting equation may or may not give the appropriate representation of the constraint for the receiving system. That system is expected to interpret the descriptive form of the constraint and reformulate it automatically in a convenient form for its internal constraint solver, for solution together with other constraints applying in the particular geometric configuration concerned.

The constraints defined in this schema are valid in both two and three dimensions. At the time of writing, two-dimensional constraints are used in CAD systems mainly in the representation of two-dimensional sketches (the representation of sketches is dealt with in clause 8 of this part of ISO 10303). Three-dimensional constraints are applicable in 3D sketches defined on general planes in model space, in the positioning and orientation of features in part models and in the positioning and orientation of part models in assembly models. Except in the assembly model case, individual geometric elements within a part model are constrained with respect to each other, and the model may therefore deform as a result of parametric changes made to it. In the assembly case the constraints apply between geometric elements belonging to distinct part models, each regarded as a rigid ensemble of such elements. In an assembly model it is therefore not the part models that deform, but rather the overall configuration of assembly constituents. The constraints specified in the present schema are intended to serve for either purpose.

The schema includes representations for specific kinds of dimensional constraints, which are explained in some detail in clause 7.2.1.

Abbreviated names are used in the identifiers of certain of the entity data types declared in this schema. Prefixes used in these identifiers have the following meanings:

pgc	**parallel_geometric_constraint**
pdgc	**point_distance_geometric_constraint**
cdgc	**curve_distance_geometric_constraint**
sdgc	**surface_distance_geometric_constraint**
rgc	**radius_geometric_constraint**
clgc	**curve_length_geometric_constraint**
pogc	**parallel_offset_geometric_constraint**
agc	**angle_geometric_constraint**

The abbreviations are used to shorten the identifiers of dimensional subtypes of the entity data types occurring in the right-hand column, to facilitate their use in domain (WHERE) rules.

7.2.1 Dimensional constraints

The dimensional capabilities provided in this schema are valid in either two or three dimensions. Their two dimensional cases cover the requirements of parametric drawings. The use of the entity data type **fixed_instance_attribute_set** specified in clause 4.4.8 also allows the representation of fixed dimensions in traditional non-parametric drawings in a manner consistent with existing standards on draughting conventions.

All the constraints defined in this part of ISO 10303 are logical in the sense that they make assertions that are either TRUE, FALSE or UNKNOWN. In principle, only the first two of those values apply, though in practice the limitations of computational processes may sometimes make it impossible to judge the validity of a constraint. However, a distinction is made in this schema between purely*logical* constraints and *dimensional* constraints. The former require some unquantified condition to apply (such as parallelism of lines or tangency of curves), while the latter prescribe a numerical value for a dimension. In this schema, some dimensional constraints are specified as subtypes of logical constraints because the logical condition must first apply before the dimensional value has any significance. Thus the supertype asserts the logical relationship between elements, and the dimensional subtype quantifies that relationship.

EXAMPLE 1 A constraint on the distance between two planes is meaningless unless those planes are parallel. Here parallelism is the logical condition.

For the purposes of this schema, a constraint is regarded as dimensional if it specifies an explicit numerical value for a dimension. All others are logical constraints.

EXAMPLE 2 The requirement for a line to be tangent to a circle is an example of a logical constraint, and the requirement that a circle has a radial dimension of 10 units is an example of a dimensional constraint.

While the distinction between logical and dimensional constraints is adequate for the purposes of this part of ISO 10303, it is recognized to be slightly arbitrary, as shown by the following example.

EXAMPLE 3 Circle 1 is required to have the same radius as Circle 2, which has a numerically specified radius dimension. The constraint can be viewed in either of two lights, as follows:

a) The radius of Circle 2 has a numerical value, and hence the constraint is equivalent to specifying a value for the radius of Circle 1. Then this is a dimensional constraint;

b) Alternatively, the radii of Circles 1 and 2 are required to be equal, and this is either true, false (or possibly unknown) in the model. Then the constraint is logical.

NOTE A futher distinction made in the dimensioning and tolerancing community is that between *intrinsic* and *relational* dimensions. These are sometimes referred to alternatively as *dimensions of size* and *dimensions of location*, respectively. In the first only one element is involved, and the dimension is an inherent property of that element. In the second, two elements are involved and the dimension specifies a relationship between them.

Both types of dimension occur in the present schema. For example,

rgc_with_dimension: This constraint, defined in 7.4.14, is a subtype of **radius_geometric_constraint**. It specifies a dimension of size, the radius of a circle. It makes no reference, implicit or explicit, to any element that is not part of the definition of the circle;

pgc_with_dimension: This constraint, defined in 7.4.4, is a subtype of **parallel_geometric_constraint** that can be used to specify a dimension of location. In its directed form, a reference line or plane will act as a datum element and the dimension will control the location of another line or plane with respect to the datum.

Currently there is no specialized treatment of these two types of dimension in this part of ISO 10303, but more prominence may be given to the distinction between them in future editions if that is found to be desirable.

7.2.2 Semantics of dimensional constraints

This part of ISO 10303 defines a dimension in terms of the numerical value of an attribute of certain entity data type instances. The type of the numerical value may be one of the SELECT values of **measure_value** as defined in ISO 10303-41.

A dimensional attribute may have an associated **bound_model_parameter** or belong to a **fixed_instance_attribute_set** (see clause 4).

If referenced by an instance of **bound_model_parameter**, the dimensional attribute is potentially subject to modification in the receiving system, subject to restrictions imposed by

— the parameter's specified domain;

— any free-form constraints in which the parameter participates;

— any explicit geometric or other constraints that affect the entity instance to which the attribute belongs.

Alternatively, if a numerical attribute defining a dimension is a member of a **fixed_instance_attribute_set**, the intention is that it should be constrained to be invariant in the receiving system.

EXAMPLE 1 A bound **model_parameter** may be associated with a dimensional value in a shape model, but an unbound **model_parameter**, by definition, has no such direct association. For example, the **x**, **y** and **z** dimensional attributes of a **block** solid, as defined in ISO 10303-42, may be associated with bound **model_parameter** instances with name attributes `'L','W'` and `'H'` respectively. Then relations $L = T^2$, $W = 2T$ and $H = 2 + \cos T$ may be modelled by **free_form_assignment** constraints, in which T represents an unbound **model_parameter** named `'T'`. This unbound parameter will not be directly associated with a dimension, though it will indirectly control the values of three dimensions.

EXAMPLE 2 In the previous example, the dimensions of the block can only be changed by editing the parameter named `'T'`. This is because the values of the parameters named `'L','W','H'` are specified through free-form assignments in which `'T'` is the reference element and they are constrained elements. In such a case the values of the constrained elements can only be edited by changing the value of the reference element. This shows that the

value of a dimensional attribute having a bound parameter cannot always be edited directly. Account must always be taken of other constraints in which the parameter participates.

Finally, it should be emphasized once again that logical or dimensional constraints in a shape model are redundant for purposes of static model transfer. They do not affect the shape of the exchanged model. The assertions they make are expected to be true when a model is exchanged, their effect only becoming apparent when that model is modified in the receiving system. In that event they are intended to govern the nature of the permissible changes that may be made to it.

7.2.3 Constraints on procedurally defined model elements

ISO 10303-42 contains representations for certain types of geometric modelling elements that have a procedural nature; that is, their geometry is not explicitly defined. Examples include the following:

offset curves and surfaces: These are defined in terms of a base element (a curve or a surface) and an offsetting operation;

swept surfaces and solids: these are similarly defined in terms of a base element (a curve or a face) and a sweep operation.

If such an element is captured and transferred into a receiving system, provided the instance of the base entity data type and the nature of the operation performed on it are transferred satisfactorily, its exchange should in principle present no problems. However, difficulties potentially arise when explicit representations are computed from the implicit or procedural representations. There is no provision in ISO 10303-42 to ensure that the editing of the base element, or of the parameters of the operation performed upon it, will give rise to appropriate changes in the calculated explicit representation. For this reason, several constraints are defined in this schema to ensure the maintenance of compatibility between implicit and explicit representations under such circumstances. They include **parallel_offset_geometric_constraint**, **swept_point_curve_geometric_constraint** and **swept_curve_surface_geometric_constraint**, as defined in 7.4.17, 7.4.26 and 7.4.27 respectively.

7.3 Explicit geometric constraint type definitions

7.3.1 geometric_constraint_element

The **geometric_constraint_element** type allows a selection between the major subtypes of **geometric_representation_item** that are subject to explicit geometric constraints, namely points, curves, surfaces, vectors and directions.

EXPRESS specification:

```
*)
TYPE geometric_constraint_element = SELECT
  (point,
   curve,
   surface,
   vector,
   direction);
END_TYPE;
(*
```


7.3.2 point_curve_or_surface_constraint_element

The **point_curve_or_surface_constraint_element** type allows a selection between entity data types that represent points, curves or surfaces.

EXPRESS specification:

```
*)
TYPE point_curve_or_surface_constraint_element = SELECT
  (point,
   curve,
   surface);
END_TYPE;
(*
```

7.3.3 curve_or_surface_constraint_element

The **curve_or_surface_constraint_element** type allows a selection between entity data types that represent curves or surfaces.

EXPRESS specification:

```
*)
TYPE curve_or_surface_constraint_element = SELECT
  (curve,
   surface);
END_TYPE;
(*
```

7.3.4 linear_geometry_constraint_element

The **linear_geometry_constraint_element** type allows a selection between entity data types representing lines, planes, directions and vectors.

EXPRESS specification:

```
*)
TYPE linear_geometry_constraint_element = SELECT
  (line,
   plane,
   direction,
   vector);
END_TYPE;
(*
```

7.3.5 radial_geometry_constraint_element

The **radial_geometry_constraint_element** type allows a selection between entity data types representing circles, cylindrical surfaces, spherical surfaces, right circular cylinders and spheres. All of these are

characterised in terms of a single radius value except for

— **conical_surface**, which has an additional **semi_angle** attribute;

— **right_circular_cylinder**, which has an additional **height** attribute;

— **right_circular_cone**, which has both additional attributes.

EXPRESS specification:

```
*)
TYPE radial_geometry_constraint_element = SELECT
  (circle,
   cylindrical_surface,
   conical_surface,
   spherical_surface,
   right_circular_cylinder,
   right_circular_cone,
   sphere);
END_TYPE;
(*
```

7.3.6 axial_geometry_constraint_element

The **axial_geometry_constraint_element** type allows a selection between entity data types representing points, lines, circles, and surfaces and solids having an axis of rotational symmetry.

EXPRESS specification:

```
*)
TYPE axial_geometry_constraint_element = SELECT
  (point,
   line,
   circle,
   plane,
   cylindrical_surface,
   conical_surface,
   spherical_surface,
   toroidal_surface,
   surface_of_revolution,
   sphere,
   right_circular_cone,
   right_circular_cylinder,
   torus,
   revolved_face_solid,
   revolved_area_solid);
END_TYPE;
(*
```

7.3.7 swept_surface_or_solid

The **swept_surface_or_solid** type allows a selection between all the procedurally defined swept surfaces and volumes specified in ISO 10303-42.

EXPRESS specification:

```
*)
TYPE swept_surface_or_solid = SELECT
  (swept_surface,
   swept_face_solid,
   swept_area_solid);
END_TYPE;
(*
```

7.3.8 tangent_contact_type

The type **tangent_contact_type** enumerates the different types of tangency relationships that are possible between curves and surfaces.

EXPRESS specification:

```
*)
TYPE tangent_contact_type = ENUMERATION OF
  (point_contact,
   curve_contact,
   surface_contact);
END_TYPE;
(*
```

Enumerated item definitions:

point_contact: Tangency occurs at an isolated point;

curve_contact: Tangency occurs along a finite line or curve segment;

surface_contact: Tangency occurs over a finite surface region.

EXAMPLE The second of these conditions could occur if two different composite curves each happened to have a segment lying on same parent curve. An analogous situation for composite surfaces could give rise to the third condition.

7.3.9 parallel_offset_type

The type **parallel_offset_type** enumerates the different types of parallel offset relationships defined in ISO 10303-42.

EXPRESS specification:

```
*)
TYPE parallel_offset_type = ENUMERATION OF
  (curve_2d_offset,
   curve_3d_offset,
   surface_offset);
END_TYPE;
(*
```

Enumerated item definitions:

curve_2d_offset: The parallel offset relationship applies to planar curves, all defined in the same plane;

curve_3d_offset: The parallel offset relationship applies to three-dimensional curves (see the explanation in 7.4.17);

surface_offset: The parallel offset relationship applies to surfaces.

7.3.10 non_negative_length_measure

The **non_negative_length_measure** type defines a type of **length_measure** whose value is greater than or equal to zero.

EXPRESS specification:

```
*)
TYPE non_negative_length_measure = length_measure;
WHERE
  WR1: SELF >= 0;
END_TYPE;
(*
```

NOTE This type of **length_measure** is appropriate for constraining distances between components in assemblies, which may be in actual contact with each other.

7.4 Explicit geometric constraint entity definitions

7.4.1 explicit_geometric_constraint

The entity data type **explicit_geometric_constraint** is a type of **explicit_constraint** that asserts relationships between elements of a geometric model in descriptive terms. Such constraints in general have two forms:

directed: in which all members of a set or list of constrained geometric elements are constrained with respect to one or more reference elements;

undirected: in which there is no reference element and the constraint is required to hold between all possible pairs of a set of constrained geometric elements.

NOTE 1 Constraints of these two kinds are sometimes alternatively referred to respectively as *asymmetrical* and *symmetrical*.

EXPRESS specification:

```
*)
ENTITY explicit_geometric_constraint
  ABSTRACT SUPERTYPE OF (ONEOF
    (fixed_element_geometric_constraint,
     parallel_geometric_constraint,
     point_distance_geometric_constraint,
     skew_line_distance_geometric_constraint,
     curve_distance_geometric_constraint,
     surface_distance_geometric_constraint,
     radius_geometric_constraint,
     curve_length_geometric_constraint,
     parallel_offset_geometric_constraint,
     angle_geometric_constraint,
     perpendicular_geometric_constraint,
     incidence_geometric_constraint,
     coaxial_geometric_constraint,
     tangent_geometric_constraint,
     symmetry_geometric_constraint,
     swept_point_curve_geometric_constraint,
     swept_curve_surface_geometric_constraint,
     curve_smoothness_geometric_constraint,
     surface_smoothness_geometric_constraint))
  SUBTYPE OF (defined_constraint, geometric_representation_item);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF geometric_representation_item;
  SELF\explicit_constraint.reference_elements   :
    SET[0:?] OF geometric_representation_item;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of constrained **geometric_representation_item** instances.

SELF\explicit_constraint.reference_elements: If not empty, the set of **geometric_representation_item** instances upon which the constrained elements are dependent in a directed constraint.

NOTE 2 ISO 10303-43 ensures that a representation using an instance of **explicit_geometric_constraint** also uses the members of the sets **constrained_elements** and **reference_elements** pertaining to that instance. Consequently, these elements all share the same instances of **representation_context**. Because **explicit_geometric_constraint** is a subtype of **geometric_representation_item**, these representation context instances will be of the ISO 10303-42 type **geometric_representation_context**, which has a dimensionality attribute **coordinate_space_dimension**. The ISO 10303-42 global rule **compatible_dimension** requires that all items in a **geometric_representation_context** have the same dimensionality. This ensures that all the constrained and reference elements have the same dimensionality as the instance of **explicit_geometric_constraint** that references them.

7.4.2 fixed_element_geometric_constraint

The **fixed_element_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that a set of **geometric_constraint_element** instances are invariant in the sense that the values of all their attributes are fixed.

NOTE An important use of this constraint is to anchor constrained configurations in space, to avoid constraint equations having infinitely many solutions due to the existence of translational or rotational degrees of freedom.

EXPRESS specification:

```
*)
ENTITY fixed_element_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF geometric_constraint_element;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.reference_elements) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of **geometric_constraint_element** instances whose attributes are frozen by the constraint.

Formal propositions:

WR1: No reference elements participate in the **fixed_element_geometric_constraint**.

7.4.3 parallel_geometric_constraint

The **parallel_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that the members of a set of two or more **linear_geometry_constraint_element** instances (lines, planes, directions or vectors) are mutually parallel. A reference element may be provided; the constraint is directed if this is the case, and undirected if not.

Lines, directions and vectors are regarded as parallel if their directions are either equal or opposite; similarly, planes are parallel if their normal directions are either equal or opposite. Parallelism of a line, direction or vector instance with a plane requires perpendicularity of its direction with respect to the plane normal. If both planes and other permissible elements both occur in the constraint, all lines, directions and vectors shall be mutually parallel, all the planes shall be mutually parallel and the lines, directions and vectors shall be parallel to the planes.

EXPRESS specification:

```
*)
ENTITY parallel_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF linear_geometry_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET[0:1] OF linear_geometry_constraint_element;
END_ENTITY;
(*
```


Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of constrained elements.

SELF\explicit_constraint.reference_elements: If present, the **linear_geometry_constraint_element** instance with which the members of the set of constrained elements are constrained to be parallel.

NOTE 1 Axial lines of **axial_geometric_constraint** elements may be constrained in terms of their direction attributes to be parallel to other **linear_geometry_constraint_element** instances.

NOTE 2 This constraint can also be used to constrain the direction attributes of axis placements. A related use is the constraining of lines to be horizontal or vertical in two-dimensional sketches as defined in clause 8.

7.4.4 pgc_with_dimension

The **pgc_with_dimension** entity data type is a type of **parallel_geometric_constraint** (see clause 7.4.3) asserting a value for the distance between two parallel line or plane instances. If the elements concerned are a line and a plane, the distance between them shall be measured in the direction of their common normal.

Distance from a reference plane shall be measured in the direction of the normal to the plane unless the BOOLEAN attribute **negative_direction** has the value TRUE. For all other cases this attribute shall have the value FALSE.

NOTE The distance between constrained elements is defined as non-negative to allow assembly constituents to be constrained to be in mutual contact.

EXPRESS specification:

```
*)
ENTITY pgc_with_dimension
  SUBTYPE OF (parallel_geometric_constraint);
  distance_value : non_negative_length_measure;
  negative_direction : BOOLEAN;
WHERE
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) = 1)
    OR (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2);
  WR2: SIZEOF(QUERY(q <* (SELF\explicit_constraint.reference_elements +
    SELF\explicit_constraint.constrained_elements) | SIZEOF(TYPEOF(q) *
    ['GEOMETRY_SCHEMA.DIRECTION', 'GEOMETRY_SCHEMA.VECTOR']) > 0)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

distance_value: The current value of the constrained distance between parallel line or plane elements.

negative_direction: A BOOLEAN attribute whose value shall be TRUE if distance from a reference plane is being measured in the direction opposite to that of the normal to the plane. Otherwise the value of this attribute shall be **false**.

Formal propositions:

WR1: If no reference element is specified the number of constrained elements shall be two.

WR2: No instances of **direction** or **vector** shall occur in an instance of **pgc_with_dimension**.

7.4.5 point_distance_geometric_constraint

The **point_distance_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting a constraint on distances of points from each other, or from one or more reference elements that may be points, curves or surfaces. Depending on the presence or absence of reference elements, the possibilities are as follows:

— No reference element is given. In this undirected case the constraint cannot be instanced unless the dimensional subtype **pdgc_with_dimension** is also instanced, the number of constrained points then being limited to two. This allows the distance between two points to be constrained;

— One or more reference elements are specified, up to a maximum of four. The constrained points are all required to lie at equal distances from the reference elements.

EXAMPLE 1 An example of the use of four reference elements in a three-dimensional context is provided by the case of a point equidistant from the four planar faces of a tetrahedron; the point will lie at the centre of a sphere inscribed in the tetrahedron.

NOTE 1 The specification of more than three reference elements in two dimensions, or more than four in three dimensions, will generally give rise to an overconstrained situation.

Distances are measured between nearest points on the elements concerned. For a smooth unbounded curve in two dimensions, or a smooth unbounded curve or surface in three dimensions, distance will therefore be be measured along an appropriate normal to that element. For bounded curves or surfaces the nearest point may lie on a boundary of the element, in which case the distance will in general not be measured in a normal direction. These facts have implications for geometric elements defined in a piecewise manner, as shown by the following example.

EXAMPLE 2 If one element involved in the constraint is a smooth two-dimensional composite curve (i.e., with tangent continuity between segments), distances will be measured along the common normal at all points of the curve except at the end points of an open curve. If the segments do not join smoothly the nearest point may be a junction point where a tangent discontinuity occurs and the normal direction to the curve is undefined.

NOTE 2 The effect of the mid-point constraint commonly implemented in CAD systems may be achieved by using an instance of **incidence_geometric_constraint** to constrain three points to lie on a line, together with an instance of **point_distance_geometric_constraint** to constrain one of them to be equidistant from the other two.

EXPRESS specification:

```
*)
ENTITY point_distance_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements : SET[1:?] OF point;
  SELF\explicit_constraint.reference_elements   :
    SET[0:4] OF point_curve_or_surface_constraint_element;
WHERE
```



```
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) > 0) OR
    (('EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA.PDGC_WITH_DIMENSION'
    IN TYPEOF(SELF)) AND
    (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2));
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of points that are constrained.

SELF\explicit_constraint.reference_elements: If present, the set of points, curves or surfaces from which the constrained points are required to be equidistant.

Formal propositions:

WR1: If no reference element is present the dimensional subtype **pdgc_with_dimension** shall be instanced, and the set of constrained points shall have two members.

NOTE 3 The undirected form of this constraint, if used without an instance of its dimensional subtype, would only permit the constraining of points to lie at the vertices of equilateral triangles in two dimensions and equilateral triangles and tetrahedra in three dimensions. These cases have been ruled out as having little application in CAD.

7.4.6 pdgc_with_dimension

The **pdgc_with_dimension** entity data type is a type of **point_distance_geometric_constraint** as defined in clause 7.4.5. It asserts a value for the distance between two points, or between multiple points and a set of one or more reference elements.

EXPRESS specification:

```
*)
ENTITY pdgc_with_dimension
  SUBTYPE OF (point_distance_geometric_constraint);
  distance_value : non_negative_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

distance_value: The current value of the specified distance.

7.4.7 skew_line_distance_geometric_constraint

The **skew_line_distance_geometric_constraint** is a type of **explicit_geometric_constraint** asserting a value for the distance between two skew (non-parallel) lines, measured along their common normal. In the directed case one line is a reference element and the other a constrained element; in the undirected case both are constrained elements.

EXAMPLE Consider a rectangular block with a vertical cylindrical hole in its upper face. The axis of the hole may be partly located by reference to one outer edge of the upper face. The line of the referenced edge and the axis of the hole form a pair of skew lines, whose common normal lies in the plane of the upper face. The length of their common normal prescribes the distance of the hole axis from the referenced edge. It may be controlled by an instance of **skew_line_distance_geometric_constraint**.

NOTE 1 The **skew_line_distance_geometric_constraint** is valid if applied between two parallel lines, but it then only constrains the distance between them and not the parallelism relationship. If it is desired to constrain the angle between two skew lines (as projected onto the plane perpendicular to their common normal) the **angle_geometric_constraint** as defined in clause 7.4.19, or its dimensional subtype, should be used.

EXPRESS specification:

```
*)
ENTITY skew_line_distance_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements : SET[1:2] OF line;
  SELF\explicit_constraint.reference_elements   : SET[0:1] OF line;
  distance_value : non_negative_length_measure;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.constrained_elements +
    SELF\explicit_constraint.reference_elements) = 2;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of two constrained lines in the undirected case, or one in the directed case.

SELF\explicit_constraint.reference_elements: A set of zero or one reference lines, in the undirected and directed cases respectively. If a reference element is present, the single constrained element is constrained with respect to it.

distance_value: The current value of the specified distance.

Formal propositions:

WR1: Two line instances shall be involved in an instance of this constraint; either they shall both be constrained elements (the undirected case) or one shall be a constrained element and the other a reference element (the directed case).

NOTE 2 The possibility of a zero value for the distance between the lines has been allowed, as a means of constraining two non-parallel lines to intersect.

7.4.8 near_point_relationship

The entity data type **near_point_relationship** is a type of **representation_item_relationship** relating a curve or surface element to a point that lies on or close to it. This allows the specification of an approximate location where a constraint condition is satisfied on that curve or surface.

EXAMPLE An instance of **tangent_geometric_constraint**, as defined in clause 7.4.24, may be used to constrain a line to be tangent to two given non-concentric circles. The two circles are reference elements, and the line is a constrained element. There are four possible lines that satisfy this constraint. The specification of a point in the neighbourhood of each intended point of tangency allows one of those four lines to be selected. Note that a point must be specified for each of the two reference circles. The entity data type **near_point_relationship** provides the means for making the necessary association.

EXPRESS specification:

```
*)
ENTITY near_point_relationship
  SUBTYPE OF (representation_item_relationship);
  SELF\representation_item_relationship.relating_representation_item :
    curve_or_surface_constraint_element;
  SELF\representation_item_relationship.related_representation_item :
    point;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_item_relationship.relating_representation_item: A curve or surface used as a reference element in an explicit geometric constraint.

SELF\representation_item_relationship.related_representation_item: A point lying on or near the reference curve or surface, indicating the approximate location where a constraint condition applies.

7.4.9 curve_distance_geometric_constraint

The **curve_distance_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** that asserts a constraint on the distance between two curves in the undirected case, or between one curve and up to four reference elements in the directed case. Reference elements, if present, are instances of type **point_curve_or_surface_constraint_element**, which includes general points, curves and surfaces. Depending on the presence or absence of reference elements, the possibilities are as follows:

— No reference element is given. In this undirected case the constraint cannot be instanced unless the dimensional subtype **cdgc_with_dimension** is also instanced, the number of constrained curves then being limited to two. This allows the minimum distance between two curves to be constrained;

— One or more reference elements are specified, up to a maximum of four. In this case a single constrained curve is required to have equal minimum distances from all the reference elements.

NOTE 1 The specification of more than three reference elements in two dimensions, or more than four in three dimensions, will generally give rise to an overconstrained situation.

EXAMPLE 1 This type of constraint may be used to constrain a line to lie at equal minimum distances from three spherical surfaces whose centres are not collinear. A necessary condition for satisfaction of this constraint is for the line to be perpendicular to the plane containing the centres of the three spheres

NOTE 2 In a two-dimensional application of this constraint all elements concerned must have dimension two. However, if it is desired to constrain the distance between coplanar curves in a three-dimensional context, it will be appropriate first to create an instance of **incidence_geometric_constraint** to ensure that all curves lie in the same plane.

Distances are measured between nearest points on the elements concerned. The distance between a smooth unbounded curve and a smooth unbounded curve or surface will therefore be measured along an appropriate normal to both elements. For bounded curves or surfaces the nearest point may lie on a boundary of the element, in which case the distance will in general not be measured in a normal direction. These facts have implications for geometric elements defined in a piecewise manner, as shown by the following example.

EXAMPLE 2 If one element involved in the constraint is a smooth two-dimensional composite curve (i.e., with tangent continuity between segments), distances will be measured along the common normal at all points of the curve except at the end points of an open curve. On the other hand, if the segments do *not* join smoothly the nearest point may be a junction point where a tangent discontinuity occurs and therefore no curve normal exists.

To resolve ambiguity in cases of multiple minimum distances between a constrained curve and a reference element, the position of a near-point may be specified for each reference element that is not itself a point. This gives an approximate location of the nearest point on a curve or surface reference element, as computed in the sending system.

EXPRESS specification:

```
*)
ENTITY curve_distance_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements : SET[1:2] OF curve;
  SELF\explicit_constraint.reference_elements   :
    SET[0:4] OF point_curve_or_surface_constraint_element;
  near_points : SET[0:4] OF near_point_relationship;
WHERE
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) > 0) OR
    (('EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA.CDGC_WITH_DIMENSION'
    IN TYPEOF(SELF))
    AND (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2));
  WR2: SIZEOF(near_points) <=
    SIZEOF(SELF\explicit_constraint.reference_elements);
  WR3: SIZEOF(QUERY(q <* near_points | NOT
    (q\representation_item_relationship.relating_representation_item
    IN SELF\explicit_constraint.reference_elements))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of one or two curves that are constrained.

SELF\explicit_constraint.reference_elements: If not empty, the set of point, curve or surface instances from which a single constrained curve is required to be equidistant.

near_points: A set of **near_point_relationship** instances each giving the approximate location on one of the reference elements of the nearest point to the constrained element.

Formal propositions:

WR1: If no reference element is present the dimensional subtype **cdgc_with_dimension** shall be instanced, and the set of constrained curves shall have precisely two members.

WR2: The number of near-points specified shall be less than or equal to the number of reference elements.

WR3: The associated curve or surface element for each of the specified near-points shall correspond to one of the reference elements of the constraint instance.

NOTE 3 The number of members of **near_points** may be less than the number of reference elements because

— The sending system may not specify a near-point for every reference element;

— If the reference element is itself a point, no near-point is needed, and none can be defined because the attribute **relating_representation_item** of an instance of **near_point_relationship** must be a curve or surface.

7.4.10 cdgc_with_dimension

The entity data type **cdgc_with_dimension** is a type of **curve_distance_geometric_constraint** as defined in clause 7.4.9, asserting a value for the minimum distance between two curves or between a curve and a set of up to three reference elements. A **distance_value** attribute is provided for this purpose.

EXPRESS specification:

```
*)
ENTITY cdgc_with_dimension
  SUBTYPE OF (curve_distance_geometric_constraint);
  distance_value : non_negative_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

distance_value: The current value of the specified distance.

NOTE The possibility of a zero value for the distance between the elements has been allowed, to permit the case of contact between assembly constituents.

7.4.11 surface_distance_geometric_constraint

The **surface_distance_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** that asserts a constraint on the distance between two surfaces in the undirected case, or between one surface and up to three reference elements in the directed case. Reference elements, if present, are instances of type **point_curve_or_surface_constraint_element**, which includes general points, curves and surfaces. Depending on the presence or absence of reference elements, the possibilities are as follows:

— No reference element is given. In this undirected case the constraint cannot be instanced unless the dimensional subtype **sdgc_with_dimension** is also instanced, the number of constrained surfaces then being limited to two. This allows the minimum distance between two surfaces to be constrained;

— One or more reference elements are specified, up to a maximum of four. In this case a single constrained surface is required to have equal minimum distances from all the reference elements.

NOTE 1 This constraint is necessarily three-dimensional.

NOTE 2 The specification of more than four reference elements would generally give rise to an overconstrained situation.

EXAMPLE 1 A spherical surface may be constrained to be equidistant from four planes.

Distances are measured between nearest points on the elements concerned. The distance between a smooth unbounded surface and a smooth unbounded curve or surface will therefore be be measured along an appropriate normal to both elements. For bounded curves or surfaces the nearest point may lie on a boundary of the element, in which case the distance will in general not be measured in a normal direction. These facts have implications for geometric elements defined in a piecewise manner, as shown by the following example.

EXAMPLE 2 If a constrained surface is a bounded composite surface with smoothly joining (i.e., tangent-continuous) components, its distance from another element will be measured along the common normal at all points of the surface except at points on its boundary. On the other hand, if the surface components do *not* join smoothly the nearest point may occur at a point where a tangent discontinuity occurs and therefore no surface normal exists.

To resolve ambiguity in cases of multiple minimum distances between a constrained surface and a reference element, the position of a near-point may be specified for each reference element that is not itself a point. This gives an approximate location of the nearest point on a curve or surface reference element as computed in the sending system.

EXPRESS specification:

```
*)
ENTITY surface_distance_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements : SET[1:2] OF surface;
  SELF\explicit_constraint.reference_elements   :
    SET[0:4] OF point_curve_or_surface_constraint_element;
  near_points : SET[0:4] OF near_point_relationship;
WHERE
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) > 0) OR
    (('EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA.SDGC_WITH_DIMENSION'
    IN TYPEOF(SELF))
    AND (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2));
  WR2: SIZEOF(near_points) <=
    SIZEOF(SELF\explicit_constraint.reference_elements);
  WR3: SIZEOF(QUERY(q <* near_points | NOT
    (q\representation_item_relationship.relating_representation_item
    IN SELF\explicit_constraint.reference_elements))) = 0;
END_ENTITY;
(*
```


Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of one or two surfaces that are constrained.

SELF\explicit_constraint.reference_elements: If not empty, the set of **geometric_constraint_element** entities from which a single constrained surface is required to be equidistant.

near_points: A set of **near_point_relationship** instances each giving the approximate location on one of the reference elements of the nearest point to the constrained element.

Formal propositions:

WR1: If no reference element is present the dimensional subtype **sdgc_with_dimension** shall be instanced, and the set of constrained surfaces shall have precisely two members.

WR2: The number of near-points specified shall be less than or equal to the number of reference elements.

WR3: The associated curve or surface element for each of the specified near-points shall correspond to one of the reference elements of the constraint instance.

NOTE 3 The number of members of **near_points** may be less than the number of reference elements because

— The sending system may not specify a near-point for every reference element;

— If the reference element is itself a point, no near-point is needed, and none can be defined because the attribute **relating_representation_item** of an instance of **near_point_relationship** must be a curve or surface.

7.4.12 sdgc_with_dimension

The entity data type **sdgc_with_dimension** is a type of **surface_distance_geometric_constraint** (see clause 7.4.11) asserting a value for the minimum distance between two surfaces, or between a surface and a set of up to three reference elements. It provides a **distance_value** attribute for this purpose.

EXPRESS specification:

```
*)
ENTITY sdgc_with_dimension
  SUBTYPE OF (surface_distance_geometric_constraint);
  distance_value : non_negative_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

distance_value: The current value of the specified distance.

NOTE The possibility of a zero value for the distance between the elements has been allowed, to permit the case of contact between assembly constituents.

7.4.13 radius_geometric_constraint

The **radius_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that the radii of all members of a set of **radial_geometry_constraint_element** instances have the same value. It is an undirected constraint, having no reference element.

EXPRESS specification:

```
*)
ENTITY radius_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF radial_geometry_constraint_element;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.reference_elements) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of constrained **radial_geometry_constraint_element** instances.

Formal propositions:

WR1: The number of reference elements shall be zero.

NOTE 1 The **toroidal_surface**, another elementary surface defined in ISO 10303-42, is characterized by two radius attributes, and hence is excluded from this constraint. The major or minor radii of **toroidal_surface** instances may be constrained by binding parameters to them and using the facilities provided in clause 5.

NOTE 2 If it is desired to constrain a set of **radial_geometry_constraint_element** instances all to have the same radius value as the radius attribute of some other instance, this can be achieved by binding parameters to all the attributes concerned and using **equal_parameter_group** as defined in clause 5.4.3.

7.4.14 rgc_with_dimension

The **rgc_with_dimension** entity data type is a type of **radius_geometric_constraint** asserting that the members of a set of **radial_geometry_constraint_element** instances all have the same specified radius.

EXPRESS specification:

```
*)
ENTITY rgc_with_dimension
  SUBTYPE OF (radius_geometric_constraint);
  radius_value : positive_length_measure;
END_ENTITY;
(*
```


Attribute definitions:

radius_value: The current value of the specified radius.

7.4.15 curve_length_geometric_constraint

The **curve_length_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that the lengths of all members of a set of **bounded_curve** instances have the same value. It is an undirected constraint, having no reference element.

NOTE 1 While most constraints in this schema apply to unbounded geometry elements, this one initially appears anomalous because it applies to bounded elements. However, it is in effect a constraint on the distances between pairs of points, measured along a curve. It is expressed in terms of the ISO 10303-42 **bounded_curve** entity data type to avoid the need for definition of a similar entity data type in this schema.

The semantics of the **curve_length_geometric_constraint** are as follows:

a) for each constrained **bounded_curve**, depending on the specific subtype, the end points are either implicit in the curve definition or are implicitly constrained to lie on an unbounded basis curve;

b) the distance between the two end points for each **bounded_curve** is measured along the curve;

c) The distances between end points for each member of the set of **bounded_curve** elements shall be constrained to be equal.

EXPRESS specification:

```
*)
ENTITY curve_length_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF bounded_curve;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.reference_elements) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of **bounded_curve** instances whose lengths are constrained to be equal.

Formal propositions:

WR1: The number of reference elements shall be zero;

NOTE 2 Conceptually, this constraint concerns distances between pairs of points, measured along a curve. However, the ISO 10303-42 **trimmed_curve** entity data type, a subtype of **bounded_curve**, permits the trimming points to be defined in terms of either **cartesian_point** or **parameter_value** entities. In the first case the **curve_length_geometric_constraint** will operate as described above, by constraining the trimming points. In the second case the trimming points may not exist explicitly in the current result model, and the constraint will require the determination of end-points in the parameter space of the curve that achieve the desired equality of length.

7.4.16 clgc_with_dimension

The **clgc_with_dimension** entity data type is a type of **curve_length_geometric_constraint** asserting that the members of a set of **bounded_curve** instances all have the same specified length.

EXPRESS specification:

```
*)
ENTITY clgc_with_dimension
  SUBTYPE OF (curve_length_geometric_constraint);
  length_value : positive_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

length_value: The current value of the specified length.

7.4.17 parallel_offset_geometric_constraint

The **parallel_offset_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that the members of a set of curves or a set of surfaces are parallel offsets of each other. It exists in an undirected form, which requires each pair of an arbitrarily large set of constrained elements to possess the specified parallel offset relationship, and a directed form in which all constrained elements are required to be parallel offsets of a reference element. This constraint is intended to ensure that explicit computed representations of offset elements present in an exchanged model can be constrained to require their recomputation in the receiving system if the details of their procedural descriptions are edited there.

NOTE 1 The ISO 10303-42 entities **offset_curve_2d** and **offset_curve_3d** are procedurally defined entities, specified in terms of a basis curve and an offset distance. Similarly, the ISO 10303-42 entity data type **offset_surface** is defined in terms of a basis surface and an offset distance.

NOTE 2 This constraint is appropriate for use in cases where the offset geometry cannot be represented in terms of simple geometric elements, but must be approximated in terms of (for example) Bézier or B-spline curves or surfaces. These cases include offsets of the following types of geometric entities defined in ISO 10303-42:

curves: conic curves other than the circle, Bézier and B-spline curves in either two or three dimensions;

surfaces: Bézier and B-spline surfaces.

The **parallel_offset_geometric_constraint** can constrain a set of two-dimensional curves in a plane, a set of three-dimensional curves or a set of surfaces, but no combinations of these three classes of elements are permitted.

NOTE 3 As stated above, this constraint is intended primarily for use with computed approximations to offset entities. Its use with elementary geometric elements whose offsets are of the same geometric type as themselves is not appropriate. Specialized constraints are provided in this schema for handling these cases, for example the parallelism constraints for lines and planes (see clauses 7.4.3 and 7.4.4). It was decided not to subsume these constraints under the present more general one because parallelism of lines and planes, in particular, is a fundamentally important logical condition. This is evident from the work of ISO TC213 on the definition of symmetry classes with application to dimensioning and tolerancing [2].

EXPRESS specification:

```
*)
ENTITY parallel_offset_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF curve_or_surface_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET[0:1] OF curve_or_surface_constraint_element;
  offset_type : parallel_offset_type;
WHERE
  WR1: NOT(((offset_type = curve_2d_offset)
    OR (offset_type = curve_3d_offset)) AND
    (SIZEOF(QUERY( q <* (SELF\explicit_constraint.constrained_elements
    + SELF\explicit_constraint.reference_elements) |
    'GEOMETRY_SCHEMA.SURFACE' IN TYPEOF(q))) > 0));
  WR2: NOT((offset_type = surface_offset) AND (SIZEOF(QUERY( q <*
    (SELF\explicit_constraint.constrained_elements +
    SELF\explicit_constraint.reference_elements) |
    'GEOMETRY_SCHEMA.CURVE' IN TYPEOF(q))) > 0));
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of constrained **curve_or_surface_constraint_element** instances.

SELF\explicit_constraint.reference_elements: If present, the basis curve or surface with which the constrained elements are required to have the parallel offset relationship.

offset_type: The nature of the offset relationship defined by the constraint.

Formal propositions:

WR1: If the **offset_type** attribute has the value **curve_2d_offset** or **curve_3d_offset**, then no surfaces shall participate in the constraint.

WR2: If the **offset_type** attribute has the value **surface_offset**, then no curves shall participate in the constraint.

NOTE 4 In the case when the dimensional subtype **pogc_with_dimension** is also instanced, the number of constrained elements has been limited to two because (i) the constraint has no significance if the number is more than 2 when there is no reference element, and (ii) when a reference element is specified there are two offsets with a specified displacement, one on either side of it.

NOTE 5 A parallel offset of a three-dimensional curve is defined procedurally in ISO 10303-42 in terms of a reference direction. A point on the offset curve is offset from the corresponding point on the base curve by a specified distance in the direction of $\mathbf{V} \times \mathbf{T}$, where $\mathbf{V}$ is the reference direction and $\mathbf{T}$ is the tangent vector at the point on the base curve. If the offset elements are three-dimensional curves then the direction of offset is not constrained by the **parallel_offset_geometric_constraint**. However, that direction may be constrained in the dimensional form of this constraint, **pogc_with_dimension**, defined in clause 7.4.18.

NOTE 6 If it is desired to constrain a curve to be a parallel offset from a surface then a curve should first be defined on the surface and used as the reference element in this constraint.

7.4.18 pogc_with_dimension

The **pogc_with_dimension** entity data type is a type of **parallel_offset_geometric_constraint** asserting that two elements are parallel offsets of each other at a specified distance. The offset distance is measured in the curve normal direction for two-dimensional curves, in a specified reference direction for three-dimensional curves and in the surface normal direction for surfaces. In all cases, either the positive or the negative sense of the direction is allowed.

NOTE 1 The above implies that every curve has two possible offsets, one in the positive and one in the negative direction. The choice that was made in the sending system is apparent from the current model.

This constraint also allows the offset direction to be constrained in the case of three-dimensional curves. It may be used in either a directed form (with a reference element) or an undirected form (with no reference element).

EXPRESS specification:

```
*)
ENTITY pogc_with_dimension
  SUBTYPE OF (parallel_offset_geometric_constraint);
  offset_value : positive_length_measure;
  offset_direction_constrained : BOOLEAN;
WHERE
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) = 1)
    OR (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2);
  WR2: (NOT (offset_direction_constrained = TRUE)
    AND ((offset_type = curve_2d_offset)
    OR (offset_type = surface_offset)));
END_ENTITY;
(*
```

Attribute definitions:

offset_value: The current value of the constrained offset distance between parallel offset elements.

offset_direction_constrained: If TRUE, requires the offset direction for a three-dimensional curve offset to be constrained, as well as the offset distance. If the attribute is FALSE, only the offset distance is constrained.

Formal propositions:

WR1: If no reference element is specified then the number of constrained elements shall be two.

WR2: The **offset_direction_constrained** attribute shall only have the value TRUE when the elements related by the constraint are three-dimensional curves.

NOTE 2 The offset direction for the case of three-dimensional curve offsets is not explicitly given in this constraint. Its value may be determined from the current result shape model or from the corresponding instance of the procedural definition of the offset curve.

7.4.19 angle_geometric_constraint

The **angle_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting constraints on angles between instances of **linear_geometry_constraint_element** (lines, planes, directions and vectors) as described below. It shall not be instantiated in undirected form, with no reference element, except in the form of its dimensional subtype **agc_with_dimension**.

NOTE 1 The undirected form with no reference element would require a set of **linear_geometry_element** instances all to make the same angle with each other. This problem only has trivial solutions, and these are covered by the **perpendicular_geometric_constraint** defined in 7.4.21.

If a reference element is specified, then arbitrarily many constrained elements are required to make the same angle with it.

The constraint is purely geometric, taking no account of the senses of directions involved in the definition of the constrained elements. Angles between constrained elements lie in the range $0° - 90°$, and are defined as described below. In the description, the treatment of lines should be regarded as also applying to directions and vectors:

two planes: The angle is that between the lines of intersection of the planes concerned with another plane perpendicular to both of them. Equivalently, the angle is that between the directions of the normals to the planes;

a line and a plane: The angle is measured in the plane that contains the line and the normal to the plane;

two lines: The angle is that between the directions of the lines.

NOTE 2 Angular relationships between lines are usually specified when the lines are coplanar, but the **angle_geometric_constraint** nevertheless permits a unique interpretation when they are not. In this case it applies to the angle between the normal projections of the lines concerned onto a plane normal to their common perpendicular.

NOTE 3 The information conveyed by this constraint, together with sense information associated with the current result representations of the elements concerned, will allow the receiving system to reformulate the constraint, if required, to take those senses into account. In that system, the angular range can therefore be extended to $0° - 180°$.

EXPRESS specification:

```
*)
ENTITY angle_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF linear_geometry_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET[0:1] OF linear_geometry_constraint_element;
WHERE
```

```
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) = 1) OR
    (('EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA.AGC_WITH_DIMENSION'
    IN TYPEOF(SELF)) AND
    (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2));
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of constrained **linear_geometry_constraint_element** instances.

SELF\explicit_constraint.reference_elements: If present, a **linear_geometry_constraint_element** instance with respect to which the constrained elements have their angles constrained.

Formal propositions:

WR1: If no reference element is present then the dimensional subtype **agc_with_dimension** shall also be instanced and the set of constrained elements shall have precisely two members.

NOTE This constraint is not designed to represent sequences of equal angular displacements about a point or line. For this purpose it must be applied repeatedly to constrain successive pairs of elements in the sequence.

7.4.20 agc_with_dimension

The **agc_with_dimension** entity data type is a type of **angle_geometric_constraint** asserting a dimensional value for an angular constraint relationship. If no reference element is present in the supertype, the number of constrained elements is limited to two, and the attribute **angle_value** specifies the angle between them. This is the undirected form of the constraint. In the directed form, when a reference element is given, there may be arbitrarily many constrained elements, all required to make the same specified angle with the reference element.

EXPRESS specification:

```
*)
ENTITY agc_with_dimension
  SUBTYPE OF (angle_geometric_constraint);
  angle_value : plane_angle_measure;
END_ENTITY;
(*
```

Attribute definitions:

angle_value: The current value of the angle.

Informal propositions:

IP1: The specified angle shall lie in the range $0° - 90°$, or the equivalent if some system of angular units other than degrees is used in a specialization of this schema.

7.4.21 perpendicular_geometric_constraint

The **perpendicular_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that instances of **linear_geometry_constraint_element** are perpendicular to each other. The constraint may be directed or undirected. In the directed case, the constrained elements are required to be perpendicular to one or two reference elements of the same type. In the undirected case there may be either two or three constrained elements; if there are three they must all have either underlying line geometry or underlying plane geometry, but not a mixture of the two.

EXPRESS specification:

```
*)
ENTITY perpendicular_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF linear_geometry_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET[0:2] OF linear_geometry_constraint_element;
WHERE
  WR1: NOT ((SIZEOF(SELF\explicit_constraint.reference_elements) = 2) AND
    NOT ((SIZEOF(QUERY(q <* SELF\explicit_constraint.constrained_elements +
    SELF\explicit_constraint.reference_elements |
    'GEOMETRY_SCHEMA.LINE' IN TYPEOF(q))) =
    SIZEOF(SELF\explicit_constraint.reference_elements +
    SELF\explicit_constraint.constrained_elements)) XOR
    (SIZEOF(QUERY(q <* SELF\explicit_constraint.constrained_elements +
    SELF\explicit_constraint.reference_elements |
    'GEOMETRY_SCHEMA.PLANE' IN TYPEOF(q))) =
    SIZEOF(SELF\explicit_constraint.reference_elements +
    SELF\explicit_constraint.constrained_elements))));
  WR2: (SIZEOF(SELF\explicit_constraint.reference_elements) > 0) OR
    (SIZEOF(SELF\explicit_constraint.constrained_elements) IN [2,3]);
  WR3: NOT ((SIZEOF(SELF\explicit_constraint.reference_elements) = 0) AND
    (SIZEOF(SELF\explicit_constraint.constrained_elements) = 3)) AND NOT
    ((SIZEOF(QUERY(q <* SELF\explicit_constraint.constrained_elements |
    'GEOMETRY_SCHEMA.LINE' IN TYPEOF(q))) = 3) XOR
    (SIZEOF(QUERY(q <* SELF\explicit_constraint.constrained_elements |
    'GEOMETRY_SCHEMA.PLANE' IN TYPEOF(q))) = 3));
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of constrained instances of **linear_geometry_constraint_element**.

SELF\explicit_constraint.reference_elements: If defined, the set of one or two **linear_geometry_constraint_element** instances to which the constrained elements are required to be perpendicular.

Formal propositions:

WR1: If the set of reference elements contains two members, then the underlying geometry of all elements involved in the constraint shall be either line geometry or plane geometry, but not a mixture of both.

WR2: If no reference element is defined then the set of constrained elements shall have either two or three members.

WR3: If no reference element is defined and the set of constrained elements has three members, then the underlying geometry of all three elements shall be either line geometry or plane geometry, but not a mixture of both.

NOTE 1 Perpendicular lines are not required to meet in a common point. The constraint will be satisfied if the directions of the two or three lines are pairwise orthogonal, regardless of whether or not the lines intersect.

NOTE 2 When there is a single reference element there is no limit to the size of the set of constrained elements, whose membership may include both lines and planes.

NOTE 3 If there are two reference elements but they are parallel to each other the effect is that of a single reference element, except that the provision of **WR2** regarding the type of constrained elements will apply.

7.4.22 incidence_geometric_constraint

The **incidence_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that one or more **geometric_constraint_element** instances lie on (or are *incident on*) one or more reference **geometric_constraint_element** instances. The inverse case, where the reference elements are included in one or more constrained elements, is also covered by the constraint. In the undirected case, where no reference element is present, the number of constrained elements is restricted to two, one of which is required to be incident on the other.

The term ‘incidence’, as defined for the purpose of this constraint, requires the inclusion of the entire incident constrained element in the reference element, or in the inverse case the containment by the incident constrained element of the entire reference element.

Incidence may be defined between points, curves and surfaces. Such special cases as collinearity of lines are included.

EXAMPLE 1 Two intersecting lines are not incident on each other, though their point of intersection is incident on both of them, and both lines are incident on the point.

EXAMPLE 2 The constrained elements might be a set of points, all required to lie on a reference element that is a curve. Conversely, the set of points might be reference elements, and the curve constrained to pass through or interpolate them.

EXAMPLE 3 A further use of the **incidence_geometric_constraint** is to represent the constraints associated with a computed intersection curve. The intersecting surfaces may be used as reference elements and the curve constrained to be incident on both of them. If either surface is subsequently edited the curve should be recomputed appropriately.

NOTE 1 The number of constrained elements in the undirected form is restricted to two because complex rules would have to be defined to identify valid cases of the incidence of three or more elements when no reference element is present.

EXPRESS specification:

```
*)
ENTITY incidence_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF geometric_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET [0:?] OF geometric_constraint_element;
  near_points : SET[0:?] OF near_point_relationship;
WHERE
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) > 0)
    OR (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2);
  WR2: SIZEOF(near_points) <=
    SIZEOF(SELF\explicit_constraint.reference_elements);
  WR3: SIZEOF(QUERY(q <* near_points | NOT
    (q\representation_item_relationship.relating_representation_item
    IN SELF\explicit_constraint.reference_elements))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of constrained **geometric_constraint_element** instances.

SELF\explicit_constraint.reference_elements: If not empty, the set of **geometric_constraint_element** instances with which the constrained elements are required to be incident.

near_points: A set of **near_point_relationship** instances, each giving the approximate location on one of the reference elements of a point of incidence on that element.

Formal propositions:

WR1: If there are no reference elements, the number of constrained elements shall be precisely two.

WR2: The number of members of **near_points** shall not exceed the number of reference elements.

WR3: The associated curve or surface element for each of the specified near-points shall correspond to one of the reference elements of the constraint instance.

NOTE 2 The number of members of **near_points** may be less than the number of reference elements because

— The sending system may not specify a near-point for every reference element;

— If the reference element is itself a point, no near-point is needed, and none can be defined because the attribute **relating_representation_item** of an instance of **near_point_relationship** must be a curve or surface.

NOTE 3 The use of this type of constraint is not necessary or appropriate when the incidence of points or curves on parametrically defined curves or surfaces results from their being defined in the parameter space of those curves or surfaces. Thus, for example, a **point_on_curve**, specified in terms of a parameter value of that curve, is automatically incident on the curve. Hence the imposition of the corresponding **incidence_geometric_constraint** would be redundant in this case.

7.4.23 coaxial_geometric_constraint

The **coaxial_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that a set of **axial_geometry_constraint_element** instances share the same axis. The constrained set may contain a mixture of points, lines, circles and axially symmetric surfaces and solids. Points are constrained to lie on the axis, lines to be coincident with it and planes perpendicular to it. For a circular element, the axis is taken to be the line through its centre and perpendicular to its plane of definition; constrained circles are not required to be coplanar with each other. Any line through the centre of a sphere or a spherical surface may be regarded as its axis.

The constraint may be either directed or undirected. In the directed case a single specified reference element defines the axis of symmetry. The element types **point**, **plane**, **spherical_surface** and **sphere** are not permitted for use as reference elements because none of them defines a unique axis.

EXPRESS specification:

```
*)
ENTITY coaxial_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF axial_geometry_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET[0:1] OF axial_geometry_constraint_element;
WHERE
  WR1: SIZEOF(QUERY(q <* SELF\explicit_constraint.reference_elements |
    SIZEOF(TYPEOF(q) * ['GEOMETRY_SCHEMA.POINT','GEOMETRY_SCHEMA.PLANE',
    'GEOMETRY_SCHEMA.SPHERICAL_SURFACE','GEOMETRY_SCHEMA.SPHERE'])
    > 0)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A set of **axial_geometry_constraint_element** instances constrained to share the same axis.

SELF\explicit_constraint.reference_elements: If present, the element defining the common axis of all the constrained elements.

Formal propositions:

WR1: The reference element shall not be a point, plane, spherical surface or sphere, because none of those elements defines a unique axis of rotational symmetry.

7.4.24 tangent_geometric_constraint

The **tangent_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting a tangency relationship between two or more instances of **curve_or_surface_constraint_element**. In its undirected form it requires two elements to be tangential to each other, and in its directed form it requires one or more elements to be tangential to one or more reference elements.

EXAMPLE 1 Two distinct lines may be constrained both to be tangential to a pair of circles.

Tangency of two elements at a point requires both elements to include that point, and to satisfy further conditions there as follows:

tangency between two curves: The directions of the tangent vectors of the two curves must exist and be identical or opposite in direction;

tangency between a curve and a surface: The tangent vector of the curve must exist and lie in the tangent plane of the surface; equivalently, it must be perpendicular to the surface normal direction (it is assumed that the surface actually possesses a tangent plane at the point in question);

tangency between two surfaces: The tangent planes of the two surfaces must exist and coincide at the point. Equivalently, their normal directions must be identical or opposite there.

The **tangent_geometric_constraint** does not specify a point of tangency, but it requires one to exist for the constraint to be satisfied. However, *near-points* may be specified, each of which indicates the approximate location of an intended tangency in a model. These may be used in determining chosen solutions where multiple possibilities arise.

NOTE 1 Near-points play no part in the mathematical relationship defining the tangency. Their primary function is to distinguish between multiple solutions of that relationship, when they arise.

NOTE 2 Cases arise in which a constrained element may have multiple tangencies with a reference element. An example is provided by a constrained circle of radius R and a reference ellipse with the same centre and minor radius R. These curves are tangential at two distinct points. Such special cases may be handled by the association of multiple near-points with the same reference element.

An attribute **tangent_contact** specifies the type of tangential contact:

a) point contact (for example, between a plane and a sphere);

b) contact along a line or other curve (for example, between a cone and a plane);

c) contact over a region of a surface (for example, between two bounded planes).

NOTE 3 This constraint does not capture the sense comparisons (aligned/opposed) of the tangent elements at their points of tangency. However, this information is readily available from the current result shape model to whose elements the constraint applies.

EXPRESS specification:

```
*)
ENTITY tangent_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:?] OF curve_or_surface_constraint_element;
  SELF\explicit_constraint.reference_elements   :
    SET[0:?] OF curve_or_surface_constraint_element;
  near_points         : SET[0:?] OF near_point_relationship;
  tangent_contact     : tangent_contact_type;
WHERE
```

```
  WR1: (SIZEOF(SELF\explicit_constraint.reference_elements) > 0) OR
    (SIZEOF(SELF\explicit_constraint.constrained_elements) = 2);
  WR2: NOT ((SELF\geometric_representation_item.dim = 2)
    AND (tangent_contact = surface_contact));
  WR3: SIZEOF(QUERY(q <* near_points | NOT
    (q\representation_item_relationship.relating_representation_item
    IN SELF\explicit_constraint.reference_elements))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of **curve_or_surface_constraint_element** instances that are constrained.

SELF\explicit_constraint.reference_elements: If not empty, the set of **curve_or_surface_constraint_element** instances to which the each member of the set of constrained elements is required to be tangential.

near_points: A set of **near_point_relationship** instances, each giving the approximate location on one of the reference elements of one or more points of incidence on that element.

tangent_contact: An indicator of whether the tangencies are point contact, contact along a curve or contact over a surface region.

Formal propositions:

WR1: If no reference elements are defined then the number of distinct constrained elements shall be two.

WR2: If the dimensionality of the constraint is 2 then the **tangent_contact** attribute shall not have the value **surface_contact**.

WR3: The associated curve or surface element for each of the specified near-points shall correspond to one of the reference elements of the constraint instance.

NOTE 4 For this entity, the number of members of **near_points** may exceed the number of reference elements because there may be multiple tangencies with a single reference element.

7.4.25 symmetry_geometric_constraint

The **symmetry_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting that two geometric elements are symmetrically disposed with respect to a specified mirror element. The geometric elements are specified as instances of **geometric_representation_item**. In two dimensions the mirror element shall be a line. In three dimensions the mirror element shall be either a line or a plane, and the semantics of the constraint shall be as follows:

plane mirror element: The two **geometric_representation_item** instances are constrained to be mirror images of each other with respect to the plane;

line mirror element: Any plane through the mirror line cuts the constrained **geometric_representation_item** instances in two-dimensional point sets (curves together with isolated points) that are constrained to be mirror images of each other with respect to the line.

The intention of the constraint is that, following a model transfer, modification of either of the two constrained **geometric_representation_item** instances shall cause a corresponding modification in the other to maintain their mutual symmetry with respect to the reference element.

NOTE 1 To allow the specified semantics, this constraint has been defined as an undirected constraint with no reference element. If the mirror element had been used as a reference element the implication would have been that the constrained configuration could only be modified by editing that reference element.

NOTE 2 No requirement has been imposed that the constrained elements should be of the same subtype of **geometric_representation_item**. This is because those elements may represent geometrical mirror images using different representation techniques. Examples include

— symmetry of a **curve** instance and an instance of **curve_replica** defined in terms of a transformation;

— symmetry of a bounded line segment with a single-segment B-spline curve of degree 1.

Thus what is required is specifically that the point-sets represented by the two constrained elements shall be symmetric with respect to the mirror element. Even in cases where the constrained elements share the same subtype of **geometric_representation_item** there is no requirement that the parameterizations associated with the two elements have any relationship to each other.

NOTE 3 Mathematically, it is also possible to define symmetry about a *point*, but there seems to be little use for this in product modelling. Symmetry of three-dimensional elements about a line is analogous to that of symmetry of two-dimensional elements about a point, and it is anticipated that the use of the line form of reference element will normally be restricted to constraining planar two-dimensional configurations.

EXPRESS specification:

```
*)
ENTITY symmetry_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[2:2] OF geometric_representation_item;
  mirror_element : linear_geometry_constraint_element;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.reference_elements) = 0;
  WR2: SIZEOF(TYPEOF(mirror_element) *
    ['GEOMETRY_SCHEMA.DIRECTION','GEOMETRY_SCHEMA.VECTOR']) = 0;
  WR3: NOT ((SELF\geometric_representation_item.dim = 2) AND
    ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(mirror_element)));
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: A pair of **geometric_representation_item** instances that are constrained to be symmetrical about the reference element.

mirror_element: The line or plane about which symmetry between the constrained elements shall be maintained.

Formal propositions:

WR1: The constraint shall contain no reference element.

WR2: The mirror element shall not be a direction or a vector.

WR3: If the dimensionality of the constrained **geometric_representation_item** instances is two, the mirror element shall be a line.

NOTE 4 There is a frequent requirement for constraining a point to lie midway between two other points, which then lie symmetrically with respect to it. This inverse case can be handled using the **point_distance_geometric_constraint** specified in clause 7.4.5.

7.4.26 swept_point_curve_geometric_constraint

The **swept_point_curve_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting constraints on the swept edge curves of a computed explicit configuration corresponding to any subtype of the ISO 10303-42 entity data type **swept_face_solid**. The constraint asserts that these edge curves are constrained

a) to pass through the **vertex_point** instances of the swept **face_surface** instance;

b) to have the correct relation to the geometry of the directrix curve of the sweep motion, as specified below.

If this constraint is applied, and if either the swept **face_surface** or the geometry of the sweep motion is edited following a model transfer, the intention is that the receiving system shall recompute the geometry of the edge curves to maintain satisfaction of these relationships.

The directrix curves of the three subtypes of **swept_face_solid** defined in ISO 10303-42 are as follows:

— **extruded_face_solid:** An implicit line, parallel to the **direction** specified by the **extruded_direction** attribute;

— **revolved_face_solid:** An implicit circle, coaxial with the **axis1_placement** specified by the **axis** attribute;

— **surface_curve_swept_face_solid:** An explicit **curve** specified by the **directrix** attribute.

Thus the swept curves shall be lines with the same direction in the first case, circles with the same axis in the second case and parallel offsets of the directrix in the third case.

NOTE 1 Of the ISO 10303-42 entities that may be swept to generate surfaces or volumes, only the **face_surface** case gives rise to the sweeping of individual points. In all other cases it is only curves that are swept.

NOTE 2 In this constraint, as in the following one, the **constrained_elements** are explicit elements that result from the evaluation of a procedural definition. The curves generated are constrained in terms of attributes of that procedural definition, i.e. the **swept_face_solid** instance, which plays the role of reference element in the constraint. The constraint cannot be applied unless the **swept_face_solid** and the set of explicit curves it generates both exist in the model at the time of transfer.

EXPRESS specification:

```
*)
ENTITY swept_point_curve_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements : SET[1:?] OF curve;
  SELF\explicit_constraint.reference_elements   :
    SET[1:1] OF swept_face_solid;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of curve instances swept out by the points associated with the vertices of the swept face.

SELF\explicit_constraint.reference_elements: A single **swept_face_solid** instance that provides reference data for the swept curves.

Informal propositions:

IP1: The curves constrained are restricted to those swept out by the points associated with the vertices of the swept face.

7.4.27 swept_curve_surface_geometric_constraint

The **swept_curve_surface_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting constraints on the swept surfaces of a computed explicit configuration corresponding to any of the ISO 10303-42 entity data types **swept_surface**, **swept_face_solid** or **swept_area_solid**. The constraint asserts that these surfaces are constrained

a) to include the one or more curves involved in the definition of the entity data type instance that is swept;

b) to have the correct relation to the geometry of those curves and to the directrix curve of the sweep motion, as specified below.

If this constraint is applied, and if either the swept instance or the geometry of the sweep motion is edited following a model transfer, the intention is that the receiving system shall recompute the geometry of the swept surfaces to maintain satisfaction of these relationships.

The directrix curves of the swept surface and swept volume subtypes defined in ISO 10303-42 are as follows:

— **surface_of_linear_extrusion**, **extruded_face_solid**, **extruded_area_solid:** An implicit line, parallel to the **direction** specified by the **extruded_direction** attribute;

— **surface_of_revolution**, **revolved_face_solid**, **revolved_area_solid:** An implicit circle, coaxial with the **axis1_placement** specified by the **axis** attribute;

— **fixed_reference_swept_surface**, **surface_curve_swept_surface**, **surface_curve_swept_face_solid**, **surface_curve_swept_area_solid:** An explicit **curve** specified by the **directrix** attribute.

For linear extrusions, the swept surfaces shall be ruled surfaces with generators parallel to the specified direction. For rotational sweeps, the swept surfaces shall be surfaces of revolution with the specified axis. For the remaining cases the surface shall be explicit representations corresponding to the appropriate subtype of **swept_surface** as defined in ISO 10303-42.

NOTE In this constraint, as in the preceding one, the **constrained_elements** are explicit elements that result from the evaluation of a procedural definition. The surfaces generated are constrained in terms of attributes of that procedural definition, i.e. the **swept_surface_or_volume** instance, which plays the role of reference element in the constraint. The constraint cannot be applied unless the **swept_surface_or_volume** and the set of explicit surfaces it generates both exist in the model at the time of transfer.

EXPRESS specification:

```
*)
ENTITY swept_curve_surface_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements : SET[1:?] OF surface;
  SELF\explicit_constraint.reference_elements   :
    SET[1:1] OF swept_surface_or_solid;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of surface instances swept out by the curves of the swept element.

SELF\explicit_constraint.reference_elements: A single ISO 10303-42 swept surface or solid instance that provides reference data for the swept surfaces.

Informal propositions:

IP1: The surfaces constrained are restricted to those swept out by the swept curve, or the curves defining the swept face or area.

7.4.28 curve_segment_set

The entity data type **curve_segment_set** is a type of **geometric_representation_item** that defines a set of **composite_curve_segment** elements, for use in the constraint **curve_smoothness_geometric_constraint** as specified in clause 7.4.29.

NOTE Subtyping this entity data type from **geometric_representation_item** allows its use as a constrained element in a subtype of **explicit_geometric_constraint**. It is not possible to constrain elements of type **composite_curve_segment** directly, because this type is not a subtype of **geometric_representation_item**.

EXPRESS specification:

```
*)
ENTITY curve_segment_set
  SUBTYPE OF (geometric_representation_item);
  segments : SET[1:?] OF composite_curve_segment;
END_ENTITY;
(*
```

Attribute definitions:

segments: A set of **composite_curve_segment** instances.

7.4.29 curve_smoothness_geometric_constraint

The **curve_smoothness_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting specified degrees of smoothness at junctions between the individual **composite_curve_segment** instances involved in a **composite_curve** instance.

ISO 10303-42 defines an enumerated type **transition_code** for specifying this smoothness. The **transition_code** values in an exchanged model reflect the geometric continuity between curve segments at the time of transfer, but do not require that continuity to be preserved in the receiving system if the model is subsequently edited. The **curve_smoothness_geometric_constraint** provides the constraint needed for this purpose. It simply collects together all the **composite_curve_segment** instances having a particular transition code value that is intended to be preserved under modification following a model transfer.

NOTE There is no requirement for all the constrained curve segments to belong to the same composite curve. This constraint may be applied to any or all composite curve segments in a **representation** that have the same value of **transition_code**.

EXPRESS specification:

```
*)
ENTITY curve_smoothness_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET[1:1] OF curve_segment_set;
  smoothness : transition_code;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.reference_elements) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of instances of **composite_curve_segment** whose continuity properties are constrained.

smoothness: The transition code value associated with the constrained **composite_curve_segment** instances.

Formal propositions:

WR1: The number of reference elements shall be zero.

7.4.30 surface_patch_set

The entity data type **surface_patch_set** is a type of **geometric_representation_item** that defines a set of **surface_patch** elements, for use in the constraint **surface_smoothness_geometric_constraint** as specified in clause 7.4.31.

NOTE Subtyping this entity data type from **geometric_representation_item** allows its use as a constrained element in a subtype of **explicit_geometric_constraint**. It is not possible to constrain elements of type **surface_patch** directly, because this type is not a subtype of **geometric_representation_item**.

EXPRESS specification:

```
*)
ENTITY surface_patch_set
  SUBTYPE OF (geometric_representation_item);
  patches : SET[1:?] OF surface_patch;
END_ENTITY;
(*
```

Attribute definitions:

patches: A set of **surface_patch** instances.

7.4.31 surface_smoothness_geometric_constraint

The **surface_smoothness_geometric_constraint** entity data type is a type of **explicit_geometric_constraint** asserting specified degrees of smoothness at boundaries between the individual **surface_patch** instances involved in a **rectangular_composite_surface** instance.

This constraint is the surface analogue of the **curve_smoothness_geometric_constraint**, which applies to composite curves (see clause 7.4.29). The primary difference is that for composite surfaces there are two transition codes, one for each parametric direction on the surface. As for composite curves, the transmitted **transition_code** values reflect the state of geometric continuity of a composite surface at the time of model transfer, but do not constrain the prescribed conditions to be maintained in the receiving system if the model is subsequently edited. The **surface_smoothness_geometric_constraint** provides the constraint needed for this purpose. It collects together all the **surface_patch** instances that have particular transition code values in their u- and v-parameter directions that are to be preserved under modification following a model transfer.

NOTE There is no requirement for all the constrained surface patches to belong to the same composite surface. This constraint may be applied to any or all surface patches in a **representation** that have the same value of **transition_code**.

EXPRESS specification:

```
*)
ENTITY surface_smoothness_geometric_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  SELF\explicit_constraint.constrained_elements :
    SET [1:1] OF surface_patch_set;
  u_smoothness : transition_code;
  v_smoothness : transition_code;
WHERE
  WR1: SIZEOF(SELF\explicit_constraint.reference_elements) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\explicit_constraint.constrained_elements: The set of instances of **surface_patch** whose continuity properties are constrained.

u_smoothness: The transition code value associated with the u parametric direction of the constrained **surface_patch** instances.

v_smoothness: The transition code value associated with the v parametric direction of the constrained **surface_patch** instances.

Formal propositions:

WR1: The number of reference elements shall be zero.

EXPRESS specification:

```
*)
END_SCHEMA; -- explicit_geometric_constraint_schema
(*
```

8 Sketch

8.1 Introduction

The following EXPRESS declaration begins the sketch schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA sketch_schema;

REFERENCE FROM product_property_representation_schema -- ISO 10303-41
  (shape_representation);

REFERENCE FROM geometry_schema;                       -- ISO 10303-42

REFERENCE FROM topology_schema                        -- ISO 10303-42
  (face_surface);

REFERENCE FROM geometric_model_schema                 -- ISO 10303-42
  (solid_model,
   surface_model);

REFERENCE FROM representation_schema                  -- ISO 10303-43
  (item_in_context,
   mapped_item,
   representation,
   representation_item_relationship,
   representation_map,
   using_items,
   using_representations);

REFERENCE FROM variational_representation_schema      -- ISO 10303-108
  (auxiliary_geometric_representation_item,
   variational_representation,
   variational_representation_item);
(*
```

NOTE 1 The schemas referenced above, unless otherwise stated, can be found in the following Parts of ISO 10303:

product_property_representation_schema	ISO 10303-41
geometry_schema	ISO 10303-42
topology_schema	ISO 10303-42
geometric_model_schema	ISO 10303-42
representation_schema	ISO 10303-43
variational_representation_schema	clause 6 of this part of ISO 10303

NOTE 2 See annex D, Figures D.15 – D.18, for a graphical presentation of this schema.

8.2 Fundamental concepts and assumptions

This schema provides for the representation of sketches, which are planar constructions used as the basis of many high-level geometric operations in CAD systems.

NOTE 1 A sketch is sometimes alternatively referred to as a *profile* in the CAD modelling context.

NOTE 2 It is important to note that, for the purposes of this part of ISO 10303, a sketch is regarded as an actual or potential constructional element of a three-dimensional model, not an informal drawing.

A distinction is made between *neutral* sketches, defined in general two-dimensional coordinate systems, and *positioned* sketches, defined explicitly in terms of three-dimensional geometric elements in the space of the model they participate in. A means is provided for linking the definition of a positioned sketch with that of a neutral sketch to which it may be related by a mapping or transformation.

Because a positioned sketch participates directly in a CAD model, the entity data type **positioned_sketch** defined in this schema is specified as a subtype of **geometric_representation_item** (see ISO 10303-42). A neutral sketch, on the other hand, is not part of the CAD model, because it is not defined in the same space. It may however play an indirect role in a CAD model if a mapped replica of it is used to generate a positioned sketch. A neutral sketch has an independent existence, may be re-used multiple times and may be stored in a library. Neutral sketches are therefore modelled as instances of **neutral_sketch_representation**, defined as a subtype of the ISO 10303-43 entity data type **representation**.

In general, constraint relationships may apply to the elements of a sketch, and in the positioned sketch case further constraints may also be specified between those elements and reference elements in the form of *imported points* or *imported curves*. These are referred to collectively as *imported geometry*, because they are implicitly defined in terms of 'external' elements that do not lie in the plane of the sketch. Representations are provided in this schema for several types of imported points and curves.

EXAMPLE One form of imported curve is defined by projecting an external curve in a specified direction onto the plane of a positioned sketch.

NOTE 3 Two-dimensional sketches may be used in the early stages of assembly design, constituent parts being represented by their outline shapes. Three-dimensional assembly constraints may then later be derived from the two-dimensional sketch constraints.

Provision is also made for the representation of partial sketches ('subsketches'), as portions of complete sketches that may have some special significance in a design.

8.3 Sketch type definitions

8.3.1 surface_or_solid_model

The **surface_or_solid_model** type allows a selection between the ISO 10303-42 entity data types **surface_model** and **solid_model**.

NOTE This type is used in the definition of the **implicit_model_intersection_curve** and **implicit_silhouette_curve** entity data types defined in clauses 8.4.7 and 8.4.8.

EXPRESS specification:

```
*)
TYPE surface_or_solid_model = SELECT
  (surface_model,
   solid_model);
END_TYPE;
(*
```

8.3.2 planar_curve_select

The **planar_curve_select** type allows a selection between various ISO 10303-42 possibilities for explicit representations of planar three-dimensional curves.

NOTE 1 Such a curve may be used as the basis for an instance of **positioned_sketch** as defined in clause 8.4.10.

NOTE 2 Lines and conics are planar by definition. In CAD practice, any more complex type of planar three-dimensional curve will normally be defined on a pre-existing plane, either directly or by transformation of a neutral sketch onto that plane. For that reason the following ISO 10303-42 curve types, not in general associated with planes, are omitted from the SELECT list:

— **polyline**;

— **b_spline_curve**;

— **composite_curve**.

If it is desired to use a three-dimensional planar curve of one of these types as the basis of a positioned sketch it will be appropriate to represent it by an instance of **surface_curve** with a planar basis surface, whose attribute **curve_3d** references the required instance of **polyline**, **b_spline_curve** or **composite_curve**.

NOTE 3 The ISO 10303-42 entity data type **curve_replica** has also been excluded from the SELECT list because it is an implicit representation in terms of a base curve and a transformation.

NOTE 4 The ISO 10303-42 entity data type **offset_curve_3d** has additionally been excluded from the SELECT list because it is also implicitly defined, and the definition is such that no general test for planarity is possible.

EXPRESS specification:

```
*)
TYPE planar_curve_select = SELECT
  (line,
   conic,
   trimmed_curve,
   pcurve,
   surface_curve);
WHERE
  WR1: SELF\geometric_representation_item.dim = 3;
  WR2: check_curve_planarity(SELF);
END_TYPE;
(*
```

Formal propositions:

WR1: The curve shall be three-dimensional.

WR2: The curve shall be planar.

NOTE 5 If the SELECT value is **trimmed_curve** its validity can only be checked by WR2 if the underlying basis curve is of one of the other four SELECT types. It is difficult or impossible to provide a formal check for other cases, but these are not likely to be important in practice.

8.3.3 sketch_element_select

The **sketch_element_select** type allows a selection between those types of element that may be used to define the geometry of a sketch. The two basic element types provided here are **point** and **curve** as defined in ISO 10303-42, but the SELECT list may be extended in other parts of ISO 10303 to include additional types of geometric sketch element that cannot be defined as subtypes of **point** or **curve**.

EXAMPLE ISO 10303-42 requires any instance of **curve** or its subtypes to have an associated parameterization. It may be convenient to define sketch elements that have the geometric properties of curves but have no associated parameterization.

EXPRESS specification:

```
*)
TYPE sketch_element_select = EXTENSIBLE SELECT
  (point,
   curve);
WHERE
  WR1: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN TYPEOF(SELF);
END_TYPE;
(*
```

Formal propositions:

WR1: Membership of the SELECT list defined in any extension of this type shall be restricted to subtypes of **geometric_representation_item** as defined in ISO 10303-42.

8.3.4 sketch_basis_select

The **sketch_basis_select** type allows a selection between those subtypes of **geometric_representation_item** that may serve as the basis for the definition of an instance of **positioned_sketch** (see clause 8.4.10). They are all entities that may be used in the creation of the swept surface and solid entities defined in ISO 10303-42.

EXPRESS specification:

```
*)
TYPE sketch_basis_select = SELECT
  (planar_curve_select,
   curve_bounded_surface,
   face_surface);
END_TYPE;
(*
```

8.3.5 sketch_type_select

The **sketch_type_select** type allows a selection between the three primary forms of sketch or partial sketch definition provided in this schema, the **neutral_sketch_representation**, the **positioned_sketch** and the **subsketch**.

EXPRESS specification:

```
*)
TYPE sketch_type_select = SELECT
  (neutral_sketch_representation,
   positioned_sketch,
   subsketch);
END_TYPE;
(*
```

8.3.6 curves_or_area

The type **curves_or_area** has two enumerated values, which determine whether a neutral sketch is to be interpreted as a set of general curves or as the area enclosed by a set of closed curves. This type is used in the definition of **neutral_sketch_representation** (see clause 8.4.9).

EXPRESS specification:

```
*)
TYPE curves_or_area = ENUMERATION OF
  (curves, area);
END_TYPE;
(*
```

Enumerated item definitions:

curves: The neutral sketch shall be interpreted as a set of general two-dimensional curves;

area: The neutral sketch shall be interpreted as the area bounded by a set of closed two-dimensional curves.

8.4 Sketch entity definitions

8.4.1 implicit_point_on_plane

The entity data type **implicit_point_on_plane** is a type of **point**, and also of **auxiliary_geometric_representation_item**. It is the abstract supertype of a class of implicitly defined points lying in the plane of a **positioned_sketch** (see clause 8.4.10) for use as reference elements in constraints. It has an attribute **computed_representation**, of type **cartesian_point**, that provides an explicit representation of the point as computed from the implicit representation in the originating system.

NOTE 1 This entity data type has not been defined as a subtype of **point_on_surface** as defined in ISO 10303-42, because **point_on_surface** requires the point to be defined explicitly in the parameter space of the surface concerned.

NOTE 2 The attribute **computed_representation** may be used to distinguish between multiple possibilities arising from the implicit definition, for example to identify a specific point of intersection with the sketch plane of a curve that has multiple intersections with it.

EXPRESS specification:

```
*)
ENTITY implicit_point_on_plane
  ABSTRACT SUPERTYPE OF (ONEOF
    (implicit_planar_intersection_point,
     implicit_planar_projection_point))
  SUBTYPE OF (point, auxiliary_geometric_representation_item);
  using_sketch   : positioned_sketch;
  computed_representation
                 : cartesian_point;
DERIVE
  plane_of_point :
    plane := get_plane_of_implicit_geometry(using_sketch);
WHERE
  WR1: (plane_of_point\elementary_surface.position.location =
    computed_representation) XOR
    (dot_product(plane_of_point\elementary_surface.position.p[3],
    get_relative_direction_2points
    (plane_of_point\elementary_surface.position.location,
    computed_representation)) = 0);
  WR2: SIZEOF(TYPEOF(computed_representation) *
    ['SKETCH_SCHEMA.IMPLICIT_POINT_ON_PLANE',
     'GEOMETRY_SCHEMA.POINT_REPLICA']) = 0;
END_ENTITY;
(*
```

Attribute definitions:

using_sketch: The **positioned_sketch** instance whose plane contains the implicitly defined point.

computed_representation: An explicit representation of the point, computed from the implicit representation.

plane_of_point: The plane of the **positioned_sketch** instance that contains the **implicit_point_on_plane**.

Formal propositions:

WR1: Either the **computed_point** calculated from the implicit representation shall coincide with the defining point of the specified plane, or it shall lie in that plane.

WR2: The computed representation of the implicitly defined point shall not be of type **implicit_point_on_plane** or **point_replica**.

NOTE 3 WR1 tests whether the relative position vector of one point with respect to the other lies in the sketch plane by establishing whether it is perpendicular to the plane normal. This must be the case if the computed point lies in the plane but does not coincide with the point defining that plane.

NOTE 4 WR2 is intended to ensure that the computed representation of the point specifies its position explicitly, either in terms of its coordinates in model space or in an appropriate parametric coordinate system.

8.4.2 implicit_planar_intersection_point

The entity data type **implicit_planar_intersection_point** is a type of **implicit_point_on_plane**. It provides an implicit representation for a point generated by the intersection of a three-dimensional curve with the plane of a **positioned_sketch** instance.

EXPRESS specification:

```
*)
ENTITY implicit_planar_intersection_point
  SUBTYPE OF (implicit_point_on_plane);
  external_curve : curve;
END_ENTITY;
(*
```

Attribute definitions:

external_curve: The external curve whose intersection with the sketch plane defines the point.

Informal propositions:

IP1: The curve intersects the plane of the sketch.

IP2: The curve does not lie in the plane of the sketch.

8.4.3 implicit_planar_projection_point

The entity data type **implicit_planar_projection_point** is a type of **implicit_point_on_plane**. It provides an implicit representation for a point generated by the parallel projection of an external point in a specified direction onto the plane of a **positioned_sketch** instance.

EXPRESS specification:

```
*)
ENTITY implicit_planar_projection_point
  SUBTYPE OF (implicit_point_on_plane);
  external_point       : point;
  projection_direction : direction;
END_ENTITY;
(*
```

Attribute definitions:

external_point: The external point whose projection onto the sketch plane defines the implicit point.

projection_direction: The direction of projection of the external point onto the sketch plane.

Informal propositions:

IP1: If the external point lies in the sketch plane, the imported projected point shall be geometrically identical with it.

IP2: If the external point does not lie in the sketch plane, the direction of projection shall be such that the external point is projected onto the sketch plane.

8.4.4 implicit_planar_curve

The entity data type **implicit_planar_curve** is a type of **curve**, and also of **auxiliary_geometric_representation_item**. It is the abstract supertype of a class of implicitly defined curves lying in the plane of a **positioned_sketch** (see clause 8.4.10) for use as reference elements in constraints. It has an attribute **computed_representation**, also of type **curve**, that specifies an explicit approximation to the curve, computed in the originating system from the primary implicit definition.

NOTE 1 None of the various ISO 10303-42 entities for defining curves on surfaces was appropriate as a supertype for this entity data type, which may represent a curve that is either bounded or unbounded, and either simple or composite.

NOTE 2 The attribute **computed_representation** may be used to distinguish between multiple possibilities arising from the implicit definition, for example to identify one of several curves of intersection of a solid model with the sketch plane.

EXPRESS specification:

```
*)
ENTITY implicit_planar_curve
  ABSTRACT SUPERTYPE OF (ONEOF
    (implicit_intersection_curve,
     implicit_projected_curve,
     implicit_model_intersection_curve,
     implicit_silhouette_curve))
  SUBTYPE OF (curve, auxiliary_geometric_representation_item);
  using_sketch                : positioned_sketch;
  computed_representation     : curve;
DERIVE
  curve_plane : plane
    := get_plane_of_implicit_geometry(using_sketch);
WHERE
  WR1: SIZEOF(TYPEOF(computed_representation) *
    ['SKETCH_SCHEMA.IMPLICIT_PLANAR_CURVE',
     'GEOMETRY_SCHEMA.CURVE_REPLICA']) = 0;
END_ENTITY;
(*
```

Attribute definitions:

using_sketch: The **positioned_sketch** instance in which the imported implicit curve is used.

computed_representation: A computed explicit representation of the imported curve.

curve_plane: The plane of the sketch with which the imported curve is associated.

Formal propositions:

WR1: The computed representation shall not be of type **implicit_planar_curve** or **curve_replica**.

NOTE 3 WR1 ensures that the computed curve has an explicit representation.

8.4.5 implicit_intersection_curve

The entity data type **implicit_intersection_curve** is a type of **implicit_planar_curve**. It provides an implicit representation for an imported intersection curve in the plane of a **positioned_sketch** instance. The curve is defined by intersection of a specified surface with the plane of the sketch.

EXPRESS specification:

```
*)
ENTITY implicit_intersection_curve
  SUBTYPE of (implicit_planar_curve);
  external_surface : surface;
END_ENTITY;
(*
```

Attribute definitions:

external_surface: The **surface** element used to define the intersection curve.

Informal propositions:

IP1: The intersection of the sketch plane with the specified model is non-null.

IP2: The external surface is not a plane parallel to or identical with the sketch plane.

NOTE The **intersection_curve** entity data type from ISO 10303-42 was not used for this purpose because not all the attributes of its supertypes are relevant in the context of a sketch.

8.4.6 implicit_projected_curve

The entity data type **implicit_projected_curve** is a type of **implicit_planar_curve**. It provides an implicit representation for an imported curve generated by parallel projection of an external curve onto the plane of a **positioned_sketch** instance.

EXPRESS specification:

```
*)
ENTITY implicit_projected_curve
  SUBTYPE of (implicit_planar_curve);
  external_curve       : curve;
  projection_direction : direction;
END_ENTITY;
(*
```


Attribute definitions:

external_curve: The external **curve** element used to define the projected curve.

projection_direction: The direction of projection of the external curve onto the sketch plane.

Informal propositions:

IP1: If the external curve lies in the sketch plane the imported projected curve shall be identical to it.

IP2: If the external curve does not lie in the sketch plane then the direction of projection shall be such that the curve is projected onto the sketch plane.

8.4.7 implicit_model_intersection_curve

The entity data type **implicit_model_intersection_curve** is a type of **implicit_planar_curve**. It provides an implicit representation for an imported curve defined by the intersection of the plane of a**positioned_sketch** instance with a surface model or a solid model.

NOTE The explicit computed version of the intersection curve (corresponding to the attribute **computed_representation** of the **implicit_planar_curve** supertype) will generally be a composite curve.

EXPRESS specification:

```
*)
ENTITY implicit_model_intersection_curve
  SUBTYPE of (implicit_planar_curve);
  intersected_model : surface_or_solid_model;
END_ENTITY;
(*
```

Attribute definitions:

intersected_model: The **surface_or_solid_model** instance used as the basis for generating the intersection curve.

Informal propositions:

IP1: The intersection of the sketch plane with the specified model is non-null.

8.4.8 implicit_silhouette_curve

The entity data type **implicit_silhouette_curve** is a type of **implicit_planar_curve**. It provides an implicit representation for an imported silhouette curve in the plane of a**positioned_sketch** instance. The curve is generated by parallel projection of the visual profile of a solid or surface model, viewed in a specified direction, onto the plane of the sketch.

NOTE The explicit computed version of the silhouette curve (corresponding to the attribute **computed_representation** of the **implicit_planar_curve** supertype) will generally be a composite curve.

EXPRESS specification:

```
*)
ENTITY implicit_silhouette_curve
  SUBTYPE of (implicit_planar_curve);
  silhouetted_model : surface_or_solid_model;
  view_direction    : direction;
END_ENTITY;
(*
```

Attribute definitions:

silhouetted_model: The **surface_or_solid_model** used as the basis for generating the silhouette curve.

view_direction: The direction of projection of the surface or solid model profile onto the sketch plane.

Informal propositions:

IP1: The position of the specified model and the view direction are consistent with the existence of a non-null silhouette curve on the sketch plane.

8.4.9 neutral_sketch_representation

The **neutral_sketch_representation** entity data type is a type of **shape_representation**. It represents a planar configuration of geometry defined in a neutral 2D coordinate space. Such sketches are often stored in a libraries for reuse in a variety of circumstances. A neutral sketch may be interpreted either as a set of general curves or as the area bounded by a set of non-intersecting closed curves. The two cases are distinguished by the value of the attribute **neutral_sketch_semantics**. In the general curve case, no specific structural relationships are defined between the elements of the sketch.

NOTE A neutral sketch often represents an initial element in the generation of a new model or a new feature of an existing model. It can be repositioned in 3D model space by means of a transformation. Such a repositioned sketch may be used in the generation of a swept surface or solid of extrusion or revolution, or as a cross-sectional curve in the generation of an interpolated 'skinned' or lofted free-form surface.

An instance of **neutral_sketch_representation** may participate as the current result in an instance of **variational_representation** (see clause 6.3.3). This permits the association of parameters and constraints with the geometric elements of the sketch.

EXPRESS specification:

```
*)
ENTITY neutral_sketch_representation
  SUBTYPE OF (shape_representation);
  neutral_sketch_semantics  : curves_or_area;
  SELF\representation.items : SET[1:?] OF sketch_element_select;
WHERE
  WR1: SIZEOF(QUERY(q <* SELF\representation.items | q.dim <> 2)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

neutral_sketch_semantics: A two-valued attribute determining whether the sketch is to be interpreted as a set of general curves or as the area bounded by a set of closed curves.

SELF\representation.items: A set of geometric elements that compose the geometry of the sketch.

Formal propositions:

WR1: The dimensionality of every item involved in the geometry of the neutral sketch shall be 2.

Informal propositions:

IP1: If the sketch defines an area, then its geometry shall consist entirely of closed curves that have no intersections with each other.

IP2: If the sketch defines an area and the geometry of that area is defined by more than one curve then one of the defining curves shall be an outer boundary curve enclosing all the others.

8.4.10 positioned_sketch

The entity data type **positioned_sketch** is a type of **geometric_representation_item**. It provides a representation for a planar geometric configuration that may be subjected to a sweep operation or used as a section curve in the construction of a skinned or lofted surface. In a CAD system, such a configuration may be created directly in three dimensional model space or may be derived from a neutral sketch by the application of a transformation. Explicit geometric representations have therefore been provided for both neutral and positioned sketches.

NOTE 1 Conversely, it is also possible for a positioned sketch to be created in three dimensions and then for a neutral two-dimensional sketch representation to be derived from it. The same sketch may then be re-used in multiple locations and orientations in a model.

ISO 10303-42 defines the following entities resulting from the performance of sweep operations:

swept_face_solid: the result of sweeping an instance of **face_surface**;

swept_area_solid: the result of sweeping an instance of **curve_bounded_surface**;

swept_surface: the result of sweeping an instance of **curve**.

For this reason, the three entity data types **curve**, **curve_bounded_surface** and **face_surface** are all permitted as the basis of a positioned sketch.

NOTE 2 In the first and second cases ISO 10303-42 requires the swept entities to be planar. For the purposes of this schema a curve forming the basis of a positioned sketch is also required to be planar — a 3D curve would be inconsistent with the notion of a linear mapping between a 2D neutral sketch and a 3D positioned sketch.

ISO 10303-42 also defines the entity data type **sectioned_spine**, a representation of the shape of a three dimensional object defined in terms of a *spine curve* and a number of planar cross-sections positioned and oriented at points on that curve. A surface or a solid may be defined, depending on whether the

cross-sectional curves are open or closed. Instances of **positioned_sketch** based on planar curves are suitable for the representation of the cross-sectional curves of an instance of **sectioned_spine**.

If a mapping relationship exists between a **positioned_sketch** instance and a **neutral_sketch_representation** instance it may be captured by the use of **repositioned_neutral_sketch** (see clause 8.4.11) and **implicit_explicit_neutral_sketch_relationship** (see clause 8.4.12).

Auxiliary geometric elements may be defined in the plane of a positioned sketch, to act as reference elements for constraints on geometric elements of the sketch itself. Such auxiliary elements may include instances not only of the implicitly defined types of points and curves defined in this schema, but also more general types of points and curves used as reference elements.

The variational capabilities defined in earlier schemas of this part of ISO 10303 (including, in particular, parameterization and constraints) may be applied to the elements of a positioned sketch. In that case any representation in which the positioned sketch participates shall necessarily have **variational_representation** in its type list.

EXPRESS specification:

```
*)
ENTITY positioned_sketch
  SUBTYPE OF (geometric_representation_item);
  sketch_basis          : sketch_basis_select;
  auxiliary_elements    :
    SET[0:?] OF auxiliary_geometric_representation_item;
WHERE
  WR1: NOT (('GEOMETRY_SCHEMA.CURVE_BOUNDED_SURFACE' IN
    TYPEOF(sketch_basis)) AND NOT ('GEOMETRY_SCHEMA.PLANE' IN
    TYPEOF(sketch_basis.basis_surface)));
  WR2: NOT (('TOPOLOGY_SCHEMA.FACE_SURFACE' IN TYPEOF(sketch_basis)) AND
    NOT ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(sketch_basis.face_geometry)));
  WR3: SIZEOF(QUERY(q <* auxiliary_elements | (SIZEOF(TYPEOF(q) *
    ['GEOMETRY_SCHEMA.POINT','GEOMETRY_SCHEMA.CURVE']) = 0))) = 0;
  WR4: SIZEOF(QUERY(q <* auxiliary_elements |
    q\geometric_representation_item.dim <> 3)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

sketch_basis: The instance of **planar_curve_select**, **curve_bounded_surface** or **face_surface** that defines the geometry of the positioned sketch.

auxiliary_elements: Imported or other auxiliary geometric elements with respect to which geometric elements of the sketch may be constrained.

Formal propositions:

WR1: If the basis of the sketch is of type **curve_bounded_surface** then the underlying surface shall be planar.

WR2: If the basis of the sketch is of type **face_surface** then the face geometry shall be planar.

WR3: All members of the set **auxiliary_elements** shall be curves or points.

WR4: All members of the set **auxiliary_elements** shall have dimension 3.

Informal propositions:

IP1: All points and curves in the set **auxiliary_elements** that are not instances of subtypes of **implicit_point_on_plane** or **implicit_planar_curve** shall lie in the plane of the positioned sketch.

8.4.11 repositioned_neutral_sketch

The entity data type **repositioned_neutral_sketch** is a type of **mapped_item**, and also of **geometric_representation_item**. It provides (through the attributes of its **mapped_item** supertype) details of the transformation applied to a two-dimensional **neutral_sketch_representation** instance to reposition it in three-dimensional model space. This entity data type only defines the sketch geometry implicitly, in terms of the neutral sketch geometry and the transformation applied; it is provided primarily to capture details of the transformation.

NOTE 1 The transformation has the effect of raising the dimensionality of the geometric elements of the original neutral sketch from two to three.

NOTE 2 The manner in which the transformation is specified also allows determination of the inverse transformation giving the mapping from the positioned form of the sketch to the neutral form, though in this case the dimensionality of the geometric elements will be decreased from three to two.

NOTE 3 Normally the geometry of a positioned sketch will also be captured explicitly by an instance of **positioned_sketch**. The entity data type **implicit_explicit_positioned_sketch_relationship** (see clause 8.4.12) has been provided to establish a link between the implicit and explicit formulations of the same positioned sketch.

EXPRESS specification:

```
*)
ENTITY repositioned_neutral_sketch
  SUBTYPE OF (mapped_item, geometric_representation_item);
DERIVE
  map : representation_map := SELF\mapped_item.mapping_source;
WHERE
  WR1 : 'SKETCH_SCHEMA.NEUTRAL_SKETCH_REPRESENTATION'
    IN TYPEOF(map.mapped_representation);
  WR2 : 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_2D' IN
    TYPEOF(map.mapping_origin);
  WR3 : 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_3D' IN
    TYPEOF(SELF\mapped_item.mapping_target);
END_ENTITY;
(*
```

Attribute definitions:

map: The source information for the repositioning (see below).

SELF\mapped_item.mapping_source: An ISO 10303-43 **representation_map** whose attributes are

mapped_representation: the **representation** whose geometry is being mapped, in this case a **neutral_sketch_representation**;

mapping_origin: a **representation_item** from that **representation**, in this case an **axis2_placement_2d**.

SELF\mapped_item.mapping_target: This **axis2_placement_3d** defines the location and orientation in three dimensional space of the **repositioned_neutral_sketch**. The two-dimensional x- and y-axes of the neutral sketch are aligned with the corresponding axes of the three-dimensional coordinate system of the positioned sketch.

Informal propositions:

IP1: The **mapping_origin** shall be the two-dimensional axis system of the **mapped_representation**.

IP2: If the instance of **neutral_sketch_representation** that is repositioned participates as the current result in an instance of **variational_representation** because it has associated parameters and constraints (see clause 6), those parameters and constraints shall be considered to be reformulated so as to apply to the geometric elements of the transformed sketch in a manner consistent with their application to the corresponding elements in the original neutral sketch.

IP3: Conversely, if the instance of **neutral_sketch_representation** has been derived from an original instance of **positioned_sketch** having variational data associated with its elements then it shall play the role of current result in an instance of **variational_representation** and the relationships between the transformed and untransformed variational data shall be similarly consistent.

8.4.12 implicit_explicit_positioned_sketch_relationship

The entity data type **implicit_explicit_positioned_sketch_relationship** is a type of **representation_item_relationship**, defining the relationship between implicit and explicit representations of a positioned sketch. These are, respectively, instances of types **repositioned_neutral_sketch** and **positioned_sketch**; both are subtypes of **representation_item**.

EXPRESS specification:

```
*)
ENTITY implicit_explicit_positioned_sketch_relationship
  SUBTYPE OF (representation_item_relationship);
  SELF\representation_item_relationship.relating_representation_item
    : repositioned_neutral_sketch;
  SELF\representation_item_relationship.related_representation_item
    : positioned_sketch;
WHERE
  WR1: SIZEOF(QUERY(q <* using_representations(
    SELF\representation_item_relationship.related_representation_item) |
    item_in_context(
    SELF\representation_item_relationship.relating_representation_item,
    q.context_of_items))) > 0;
END_ENTITY;
(*
```


Attribute definitions:

SELF\representation_item_relationship.relating_representation_item: An implicit, transformation-based, definition of a positioned sketch.

SELF\representation_item_relationship.related_representation_item: The explicit definition of the same positioned sketch.

Formal propositions:

WR1: The implicit and explicit representations of the positioned sketch shall share at least one common representation context.

8.4.13 subsketch

The entity data type **subsketch** is a type of **geometric_representation_item**, defining a partial sketch in terms of a subset of the geometric elements composing a full sketch.

NOTE 1 The geometric elements of a subsketch may be partial geometric elements of the owning sketch, as in the case when the subsketch references just one base curve of an instance of **composite_curve** occurring in the owning sketch.

NOTE 2 The ISO 10303-42 concept of **geometric_representation_context**, with its associated dimensionality attribute, ensures that all elements of a subsketch necessarily share the dimensionality of their owning sketch. This will be 2 in the case of an instance of **neutral_sketch_representation** and 3 in the case of an instance of **positioned_sketch**. If the referenced owning sketch is itself an instance of **subsketch** then this principle applies recursively.

NOTE 3 Any variational elements associated with the definition of the full sketch shall apply also in the context of the subsketch insofar as they specify relationships between elements that occur in the subsketch.

EXPRESS specification:

```
*)
ENTITY subsketch
  SUBTYPE OF (geometric_representation_item);
  subsketch_elements : SET[1:?] OF sketch_element_select;
  owning_sketch      : sketch_type_select;
WHERE
  WR1: (('SKETCH_SCHEMA.NEUTRAL_SKETCH_REPRESENTATION'
    IN TYPEOF(owning_sketch)) AND (SIZEOF(QUERY(q <*
    subsketch_elements | NOT (owning_sketch IN
    using_representations(q)))) = 0))
    XOR
    ((SIZEOF(TYPEOF(owning_sketch) *
     ['SKETCH_SCHEMA.POSITIONED_SKETCH','SKETCH_SCHEMA.SUBSKETCH']) = 1)
    AND
    (SIZEOF(QUERY(q <* subsketch_elements |
     NOT (owning_sketch IN using_items(q,[])))) = 0));
END_ENTITY;
(*
```

Attribute definitions:

subsketch_elements: The set of geometric elements composing the subsketch.

owning_sketch: The sketch of which the subsketch forms a part.

Formal propositions:

WR1: All geometric elements belonging to the subsketch shall be used, directly or indirectly, by the owning sketch.

8.4.14 rigid_subsketch

The entity data type **rigid_subsketch** is a type of **subsketch** with specific semantics. The elements of an instance of **rigid_subsketch** are required to remain invariant in shape and in their relationships with respect to each other when the sketch to which they belong is edited, while the location and orientation of the subsketch as a whole may change subject to constraint relationships with other elements not belonging to the subsketch. The intended effect is that the group of elements behaves as a rigid body.

EXPRESS specification:

```
*)
ENTITY rigid_subsketch
  SUBTYPE OF (subsketch);
END_ENTITY;
(*
```

Informal propositions:

IP1: The elements composing an instance of **rigid_subsketch** shall transform as a rigid ensemble when the sketch of which they form a part is modified.

8.5 Sketch function definitions

8.5.1 get_relative_direction_2points

The **get_relative_direction_2points** function calculates the relative direction of one cartesian point with respect to another. The output direction is not normalized.

NOTE This function does not provide geometric founding for the **direction** returned. The caller of the function is responsible for ensuring that, where appropriate, it is used in a **representation** with a **geometric_representation_context**.

EXPRESS specification:

```
*)
FUNCTION get_relative_direction_2points
           (cp1, cp2 : cartesian_point) : direction;
```

```
  LOCAL
    d1, d2, d3, magnitude : REAL;
    result                : direction := ?;
  END_LOCAL;

  -- check that input points are three-dimensional

  IF ((cp1.dim <> 3) OR (cp2.dim <> 3)) THEN
    RETURN(result);
  ELSE

    -- construct components of vector and compute its magnitude

    BEGIN
      d1 := cp2.coordinates[1] - cp1.coordinates[1];
      d2 := cp2.coordinates[2] - cp1.coordinates[2];
      d3 := cp2.coordinates[3] - cp1.coordinates[3];
      magnitude := sqrt(d1*d1 + d2*d2 + d3*d3);
      IF (magnitude = 0) THEN
        return(result); -- direction is indeterminate in this case
      END_IF;
      result := dummy_gri || direction([d1, d2, d3]);
    END;
  END_IF;
  RETURN(result);

END_FUNCTION;
(*
```

Argument definitions:

cp1: The first **cartesian_point**.

cp2: The second **cartesian_point**, whose relative direction from the first is to be determined.

8.5.2 check_curve_planarity

The **check_curve_planarity** function checks whether a given three-dimensional curve is planar. Those types of curve that are provided as SELECT types by **planar_curve_select** (see clause 8.3.2) but are not examined by this function are automatically planar by virtue of their mode of definition.

EXPRESS specification:

```
*)
FUNCTION check_curve_planarity (checked_curve: curve) : BOOLEAN;

  LOCAL
    crv    : curve   := checked_curve;
    i,j    : INTEGER;
    result : LOGICAL := UNKNOWN;
  END_LOCAL;
```

```
-- Determine whether the curve lies on a plane, according to its type

  IF (SIZEOF(['GEOMETRY_SCHEMA.CONIC','GEOMETRY_SCHEMA.LINE'] *
    TYPEOF(crv)) > 0)
  THEN result := TRUE;
  ELSE
    IF (('GEOMETRY_SCHEMA.TRIMMED_CURVE' IN TYPEOF(crv))
      AND check_curve_planarity(crv.basis_curve))
    THEN result := TRUE;
    ELSE
      IF (('GEOMETRY_SCHEMA.PCURVE' IN TYPEOF(crv))
        AND ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(crv.basis_surface)))
      THEN result := TRUE;
      ELSE
        IF ('GEOMETRY_SCHEMA.SURFACE_CURVE' IN TYPEOF(crv))
        THEN
        BEGIN
          REPEAT j := 1 TO HIINDEX(crv.basis_surface);
            IF ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(crv.basis_surface[j]))
            THEN result := TRUE;
            END_IF;
          END_REPEAT;
        END;
        END_IF;
      END_IF;
    END_IF;
  END_IF;

  RETURN(result);

END_FUNCTION;
(*
```

Argument definitions:

checked_curve: The curve whose planarity is to be established.

8.5.3 get_plane_of_implicit_geometry

The **get_plane_of_implicit_geometry** function determines the plane associated with an instance of **implicit_point_on_plane** or **implicit_planar_curve**.

NOTE 1 Both these entity data types define the geometric element concerned in terms of a specified interaction between the plane of the positioned sketch in which it lies and some external element. The identity of the plane is therefore obtained from the definition of the owning positioned sketch, and more specifically from the relevant attribute of the entity instance on which the sketch is based. This may be of type **face_surface**, **curve_bounded_surface** or **planar_curve_select**. In the first two cases the underlying surface is constrained to be planar by a WHERE rule in the definition of **positioned_sketch** (see clause 8.4.10).

NOTE 2 If the geometry of a sketch consists of a single line segment then this function returns an indeterminate result because the geometry does not define a unique plane.

EXPRESS specification:

```
*)
FUNCTION get_plane_of_implicit_geometry
         (ps : positioned_sketch) : plane;

  LOCAL
    sb     : sketch_basis_select := ps.sketch_basis;
    result : plane := ?;
  END_LOCAL;

  -- determine plane of implicit geometry from the underlying entity data
  -- type of its owning instance of positioned_sketch_representation.  If
  -- the sketch basis is of type planar_curve_select then the planarity
  -- of the curve is guaranteed by a WHERE rule on the SELECT type of that
  -- name.

  IF ('TOPOLOGY_SCHEMA.FACE_SURFACE' IN TYPEOF(sb)) THEN
    result := sb.face_geometry;
  ELSE
    IF ('GEOMETRY_SCHEMA.CURVE_BOUNDED_SURFACE' IN TYPEOF(sb)) THEN
      result := sb.basis_surface;
    ELSE
      IF ('SKETCH_SCHEMA.PLANAR_CURVE_SELECT' IN TYPEOF(sb)) THEN
      BEGIN

        IF ('GEOMETRY_SCHEMA.CONIC' IN TYPEOF(sb))
          THEN result := dummy_gri || surface() ||
            elementary_surface(sb.position) || plane();
        END_IF;

        IF ('GEOMETRY_SCHEMA.TRIMMED_CURVE' IN TYPEOF(sb))
          THEN
          BEGIN
            result := get_plane_of_implicit_geometry(
            dummy_gri || positioned_sketch(sb.basis_curve,[]));
          END;
        END_IF;

        IF ('GEOMETRY_SCHEMA.PCURVE' IN TYPEOF(sb))
          THEN result := sb.basis_surface;
        END_IF;

        IF ('GEOMETRY_SCHEMA.SURFACE_CURVE' IN TYPEOF(sb)) THEN
        BEGIN
          IF ((SIZEOF(sb.basis_surface) = 1) -- case of one basis surface
            AND ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(sb.basis_surface[1])))
          THEN result := sb.basis_surface[1];
          ELSE -- case of two basis surfaces
            IF (('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(sb.basis_surface[1]))
              AND ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(sb.basis_surface[2])))
            THEN result := ?;
              -- both basis surfaces are planes, their intersection curve
              -- is a line, and no unique plane is defined
            ELSE -- only one of the two basis surfaces is a plane
              IF ('GEOMETRY_SCHEMA.PLANE' IN TYPEOF(sb.basis_surface[1]))
              THEN result := sb.basis_surface[1];
```

```
              ELSE result := sb.basis_surface[2];
              END_IF;
            END_IF;
          END_IF;
        END;
        END_IF;
      END;
      END_IF;
    END_IF;
  END_IF;

  RETURN(result);

END_FUNCTION;
(*
```

Argument definitions:

ps: The **positioned_sketch** whose associated plane is to be determined.

EXPRESS specification:

```
*)
END_SCHEMA; -- sketch_schema
(*
```

Annex A
(normative)

Short names of entities

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

Table A.1 – Short names of entities

Entity data type names	Short names
agc_with_dimension	AGWTDM
angle_geometric_constraint	ANGMCN
auxiliary_geometric_representation_item	AGRI
bound_model_parameter	BNMDPR
bound_parameter_environment	BNPREN
cdgc_with_dimension	CDWTDM
clgc_with_dimension	CLWTDM
coaxial_geometric_constraint	CXGMCN
curve_distance_geometric_constraint	CDGC
curve_length_geometric_constraint	CLGC
curve_segment_set	CRSGST
curve_smoothness_geometric_constraint	CSGC
defined_constraint	DFNCNS
equal_parameter_constraint	EQPRCN
explicit_constraint	EXPCNS
explicit_geometric_constraint	EXGMCN
fixed_element_geometric_constraint	FEGC
fixed_instance_attribute_set	FIAS
free_form_assignment	FRFRAS
free_form_constraint	FRFRCN
free_form_relation	FRFRRL
generated_finite_numeric_space	GFNS
implicit_explicit_positioned_sketch_relationship	IEPSR
implicit_intersection_curve	IMINCR
implicit_model_intersection_curve	IMIC
implicit_planar_curve	IMPLCR
implicit_planar_intersection_point	IPIP
implicit_planar_projection_point	IPPP
implicit_point_on_plane	IPOP
implicit_projected_curve	IMPRCR
implicit_silhouette_curve	IMSLCR
incidence_geometric_constraint	INGMCN

Table A.1 – (continued)

Entity data type names	Short names
instance_attribute_reference	INATRF
model_parameter	MDLPRM
near_point_relationship	NRPNRL
neutral_sketch_representation	NTSKRP
parallel_geometric_constraint	PRGMCN
parallel_offset_geometric_constraint	POGC
pdgc_with_dimension	PDWTDM
perpendicular_geometric_constraint	PRG2
pgc_with_dimension	PGWTDM
pogc_with_dimension	PGW0
point_distance_geometric_constraint	PDGC
positioned_sketch	PSTSKT
radius_geometric_constraint	RDGMCN
repositioned_neutral_sketch	RPNTSK
rgc_with_dimension	RGWTDM
rigid_subsketch	RGDSBS
sdgc_with_dimension	SDWTDM
simultaneous_constraint_group	SMCNGR
skew_line_distance_geometric_constraint	SLDGC
subsketch	SBSKTC
surface_distance_geometric_constraint	SDGC
surface_patch_set	SRPTST
surface_smoothness_geometric_constraint	SSGC
swept_curve_surface_geometric_constraint	SCSGC
swept_point_curve_geometric_constraint	SPCGC
symmetry_geometric_constraint	SYGMCN
tangent_geometric_constraint	TNGMCN
unbound_model_parameter	UNMDPR
unbound_model_parameter_semantics	UMPS
unbound_parameter_environment	UNPREN
variational_current_representation_relationship	VCRR
variational_representation	VRTRPR
variational_representation_item	VRRPIT

Annex B
(normative)

Information object registration

B.1 Document identification

To provide for unambiguous identification of an information object in an open system, the object identifier

```
{ iso standard 10303 part(108) version(1) }
```

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

B.2 Schema identification

B.2.1 parameterization_schema identification

To provide for unambiguous identification of the parameterization-schema in an open information system, the object identifier

```
{ iso standard 10303 part(108) version(1) schema(1)
        parameterization-schema(1) }
```

is assigned to the **parameterization_schema** (see clause 4). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

B.2.2 explicit_constraint_schema identification

To provide for unambiguous identification of the explicit-constraint-schema in an open information system, the object identifier

```
{ iso standard 10303 part(108) version(1) schema(1)
        explicit-constraint-schema(2) }
```

is assigned to the **explicit_constraint_schema** (see clause 5). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

B.2.3 variational_representation_schema identification

To provide for unambiguous identification of the variational-representation-schema in an open information system, the object identifier

```
{ iso standard 10303 part(108) version(1) schema(1)
      variational-representation-schema(3) }
```

is assigned to the **variational_representation_schema** (see clause 6). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

B.2.4 explicit_geometric_constraint_schema identification

To provide for unambiguous identification of the explicit-geometric-constraint-schema in an open information system, the object identifier

```
{ iso standard 10303 part(108) version(1) schema(1)
    explicit-geometric-constraint-schema(4) }
```

is assigned to the **explicit_geometric_constraint_schema** (see clause 7). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

B.2.5 sketch_schema identification

To provide for unambiguous identification of the sketch-schema in an open information system, the object identifier

```
{ iso standard 10303 part(108) version(1) schema(1)
                sketch-schema(5) }
```

is assigned to the **sketch_schema** (see clause 8). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex C
(informative)

Computer interpretable listings

This annex references a listing of the EXPRESS entity data type names and corresponding short names as specified in this part of ISO 10303. It also references a listing of each EXPRESS schema specified in this part of ISO 10303, without comments or other explanatory text. These listings are available in computer-interpretable form, and can be found at the following URLs:

Short names:

`http://www.tc184-sc4.org/Short_Names/`

EXPRESS:

`http://www.tc184-sc4.org/EXPRESS/`

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC184/SC4 Secretariat directly at `sc4sec@tc184-sc4.org`.

NOTE The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex D
(informative)

EXPRESS-G diagrams

The diagrams in this annex correspond to the EXPRESS schemas specified in this part of ISO 10303. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

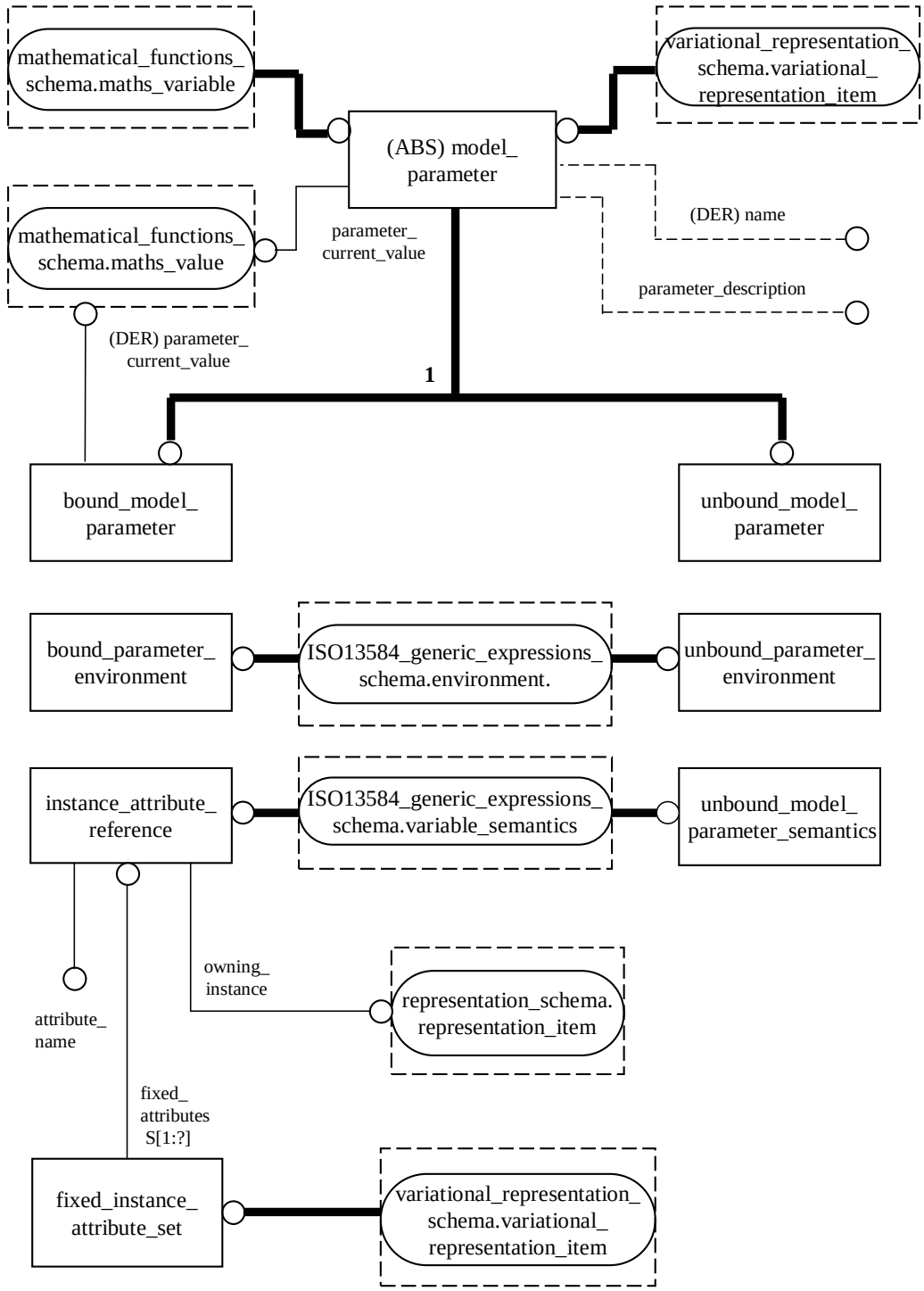

Figure D.1 – EXPRESS-G diagram of the parameterization_schema (1 of 2)

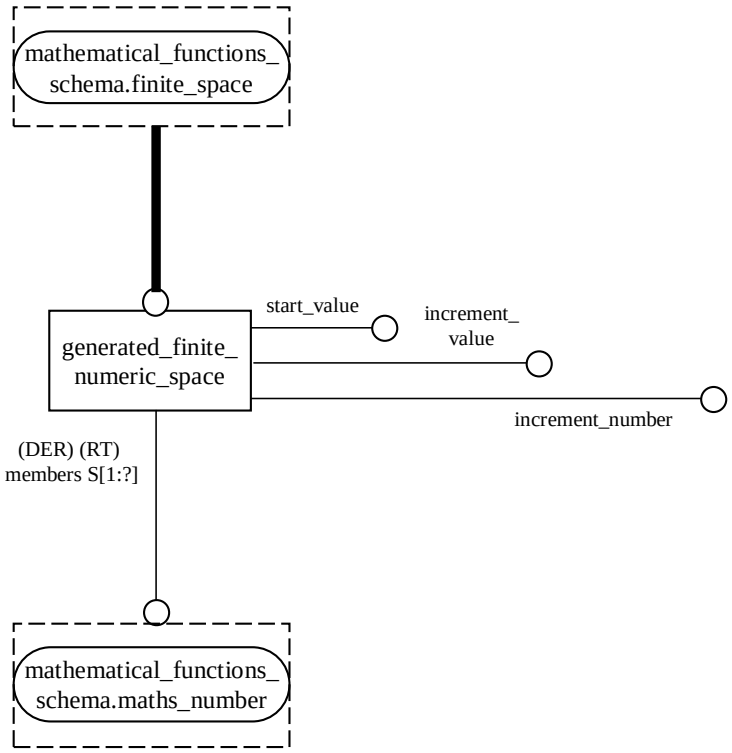

Figure D.2 – EXPRESS-G diagram of the parameterization_schema (2 of 2)

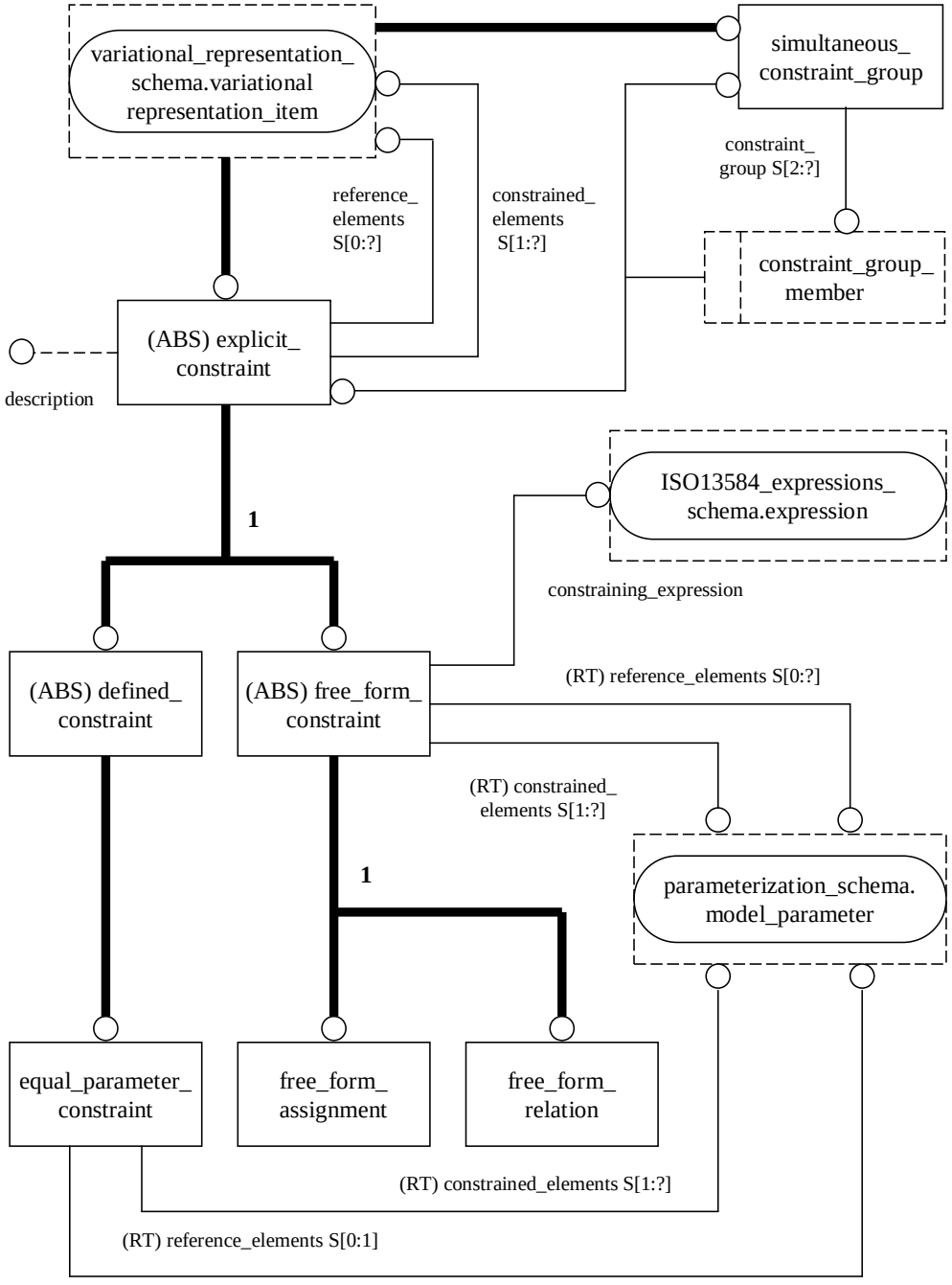

Figure D.3 – EXPRESS-G diagram of the explicit_constraint_schema (1 of 1)

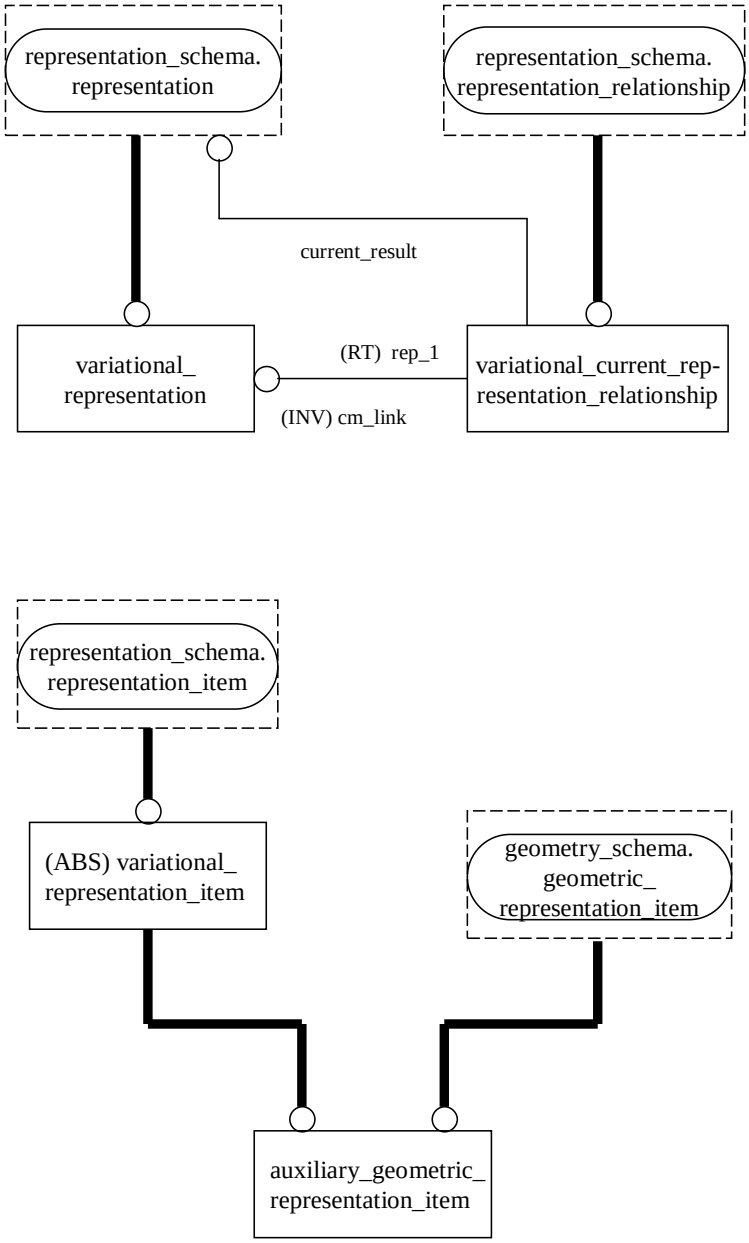

Figure D.4 – EXPRESS-G diagram of the variational_representation_schema (1 of 1)

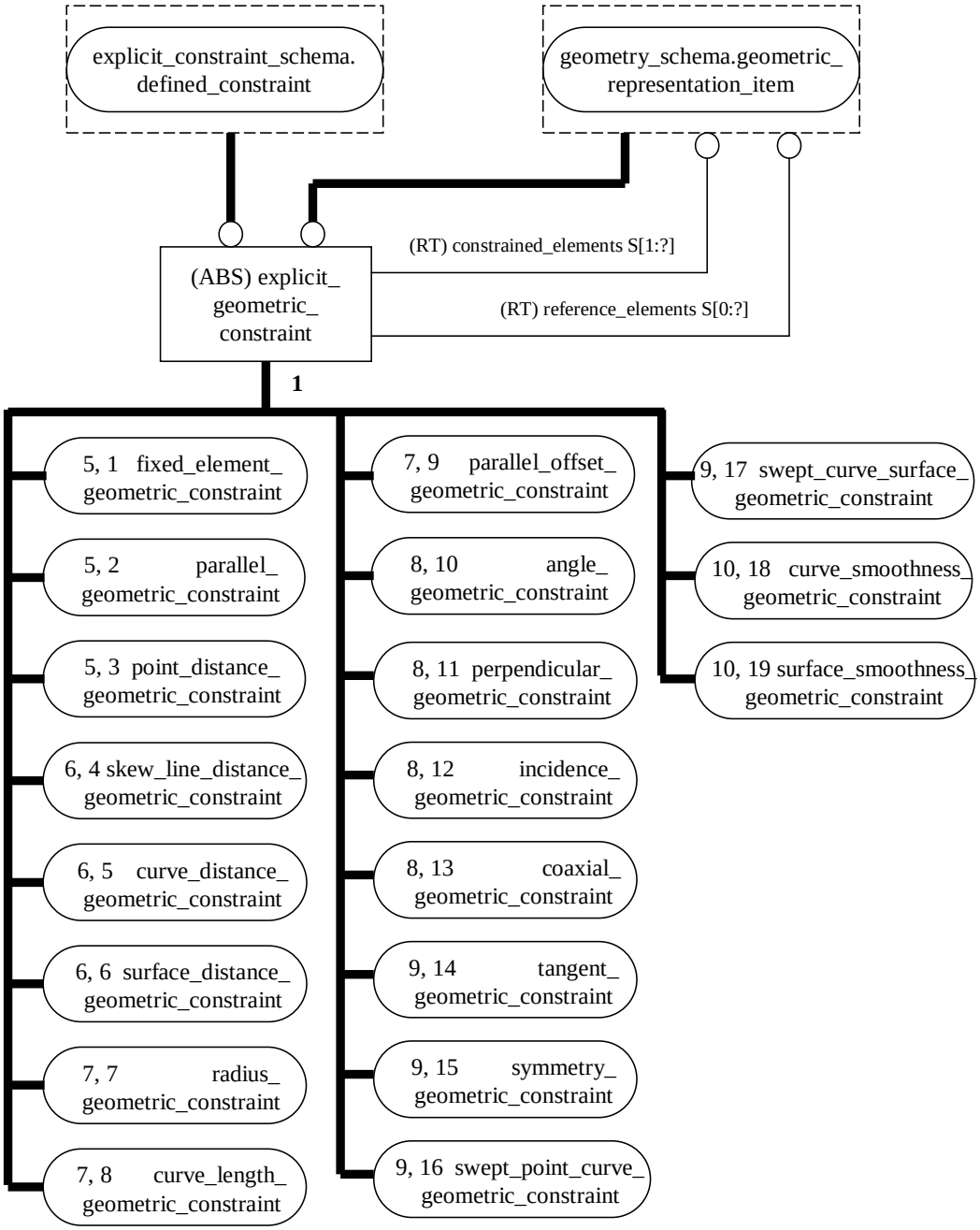

Figure D.5 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (1 of 10)

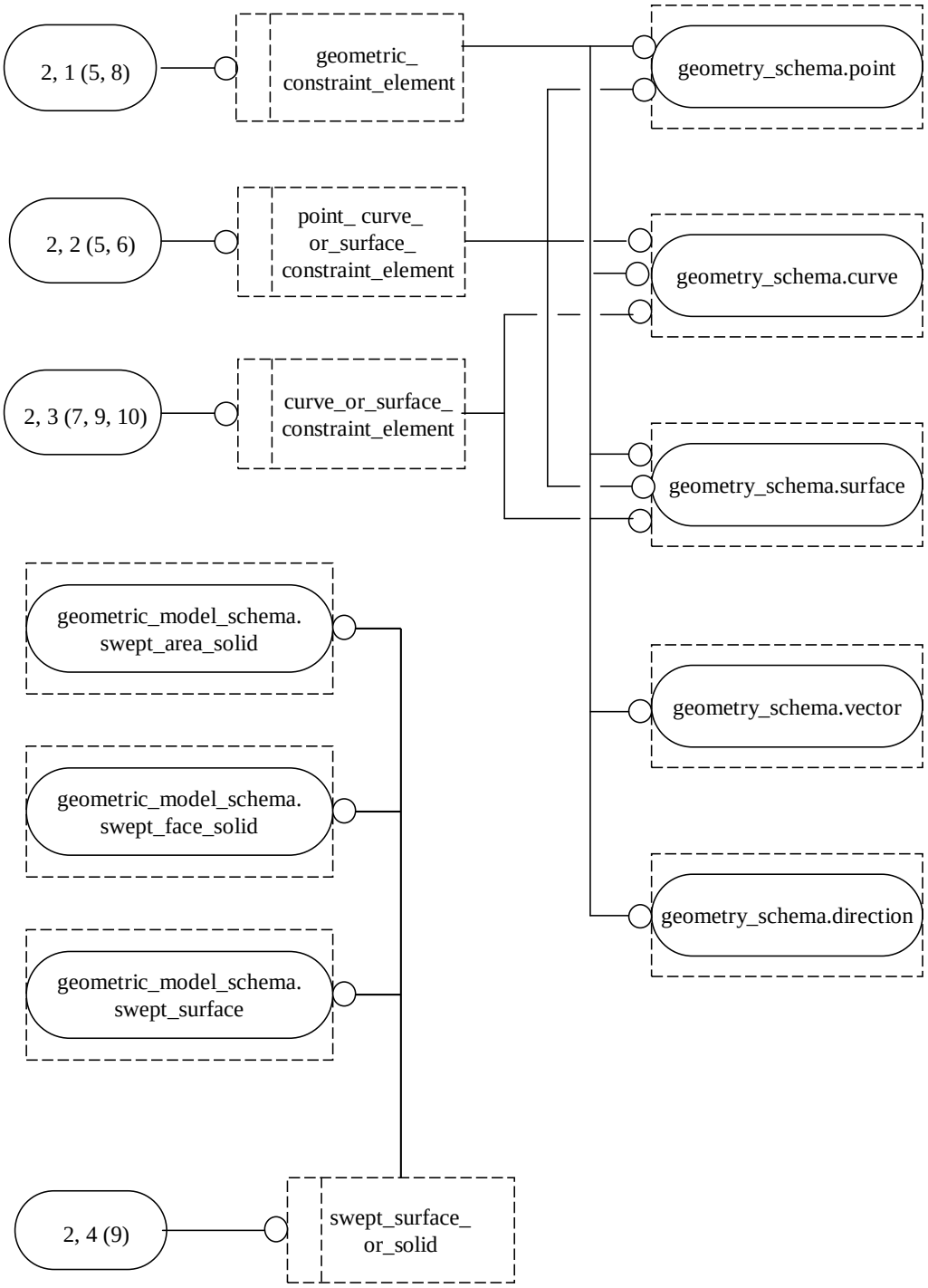

Figure D.6 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (2 of 10)

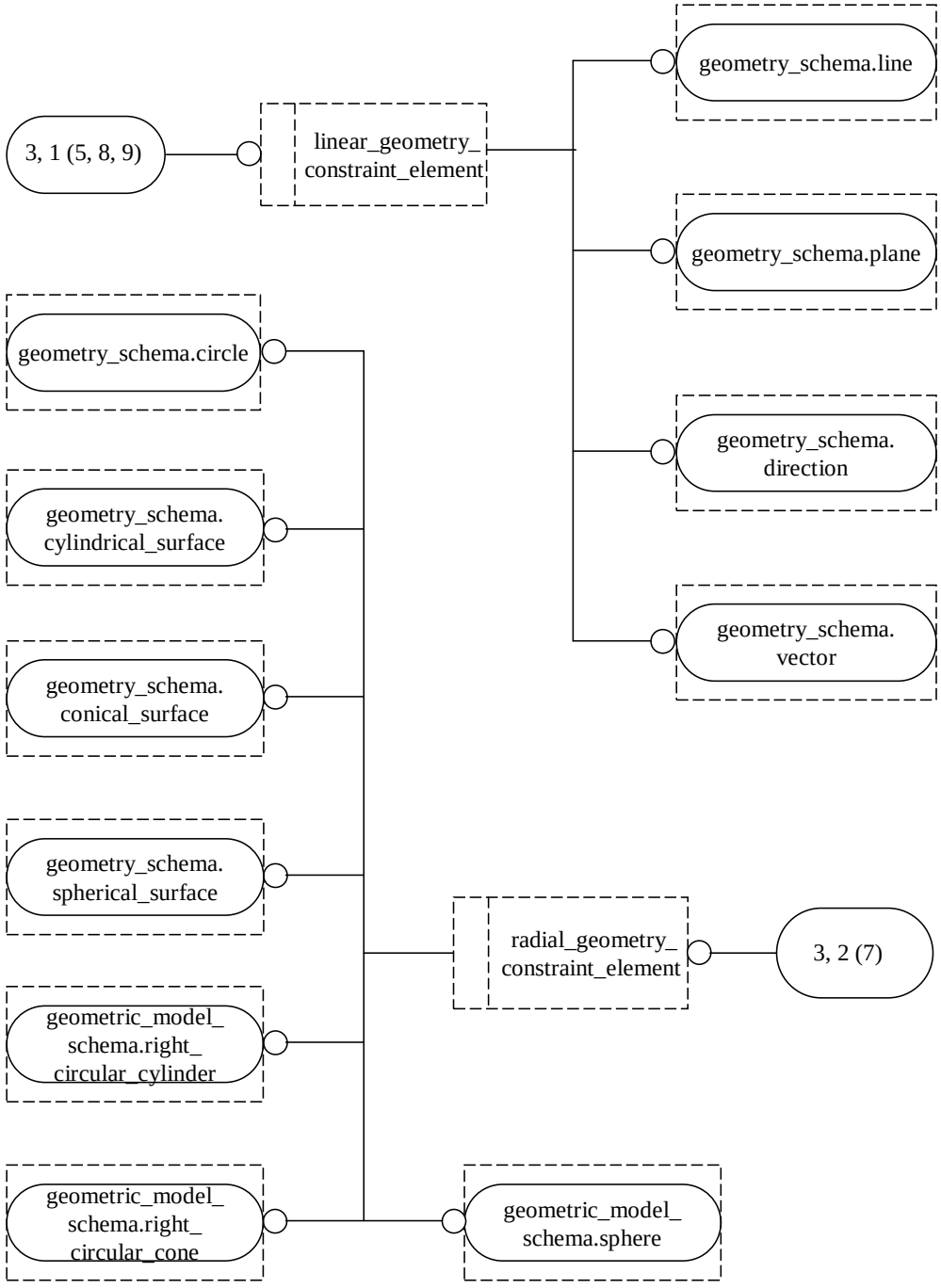

Figure D.7 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (3 of 10)

Figure D.8 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (4 of 10)

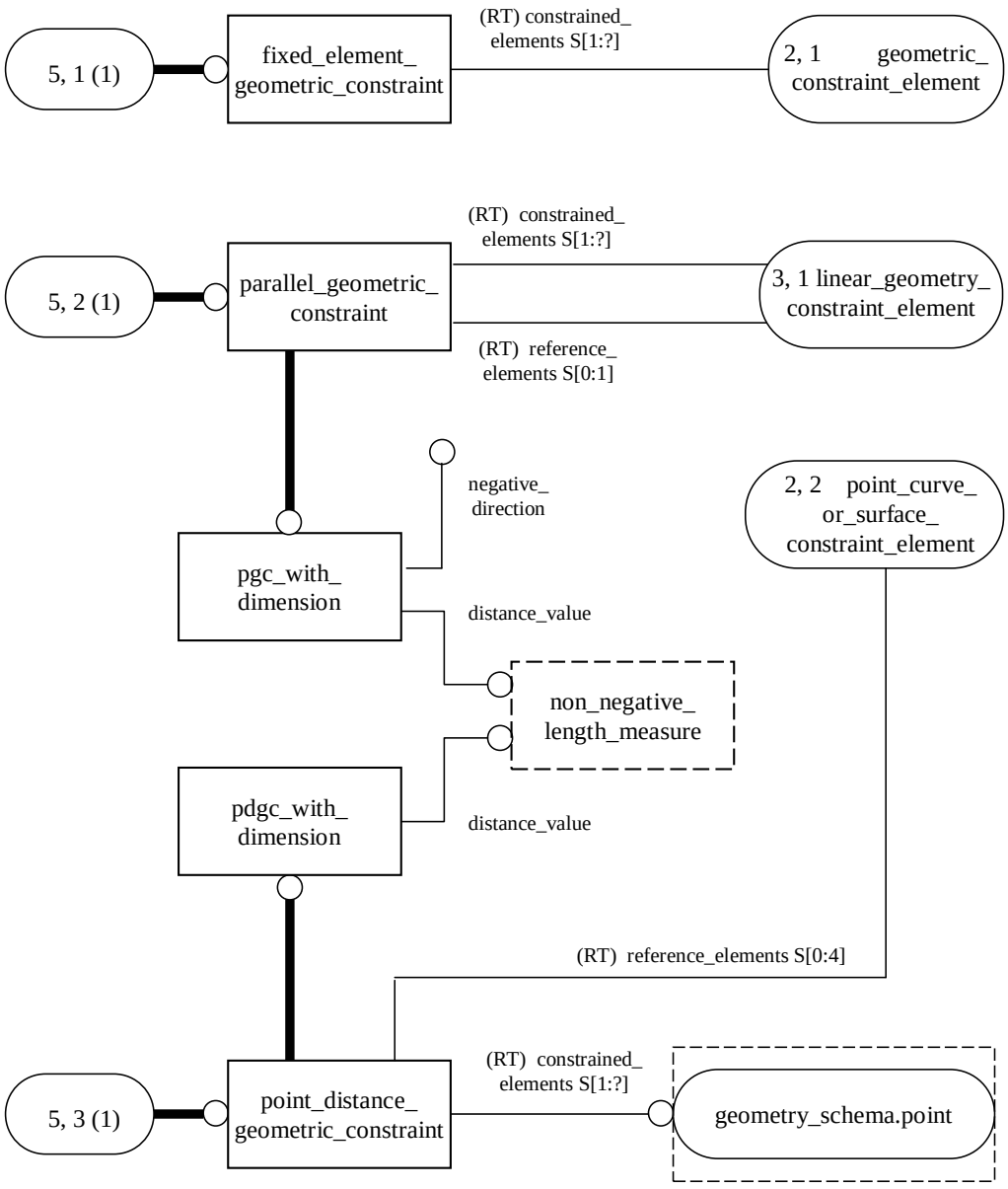

Figure D.9 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (5 of 10)

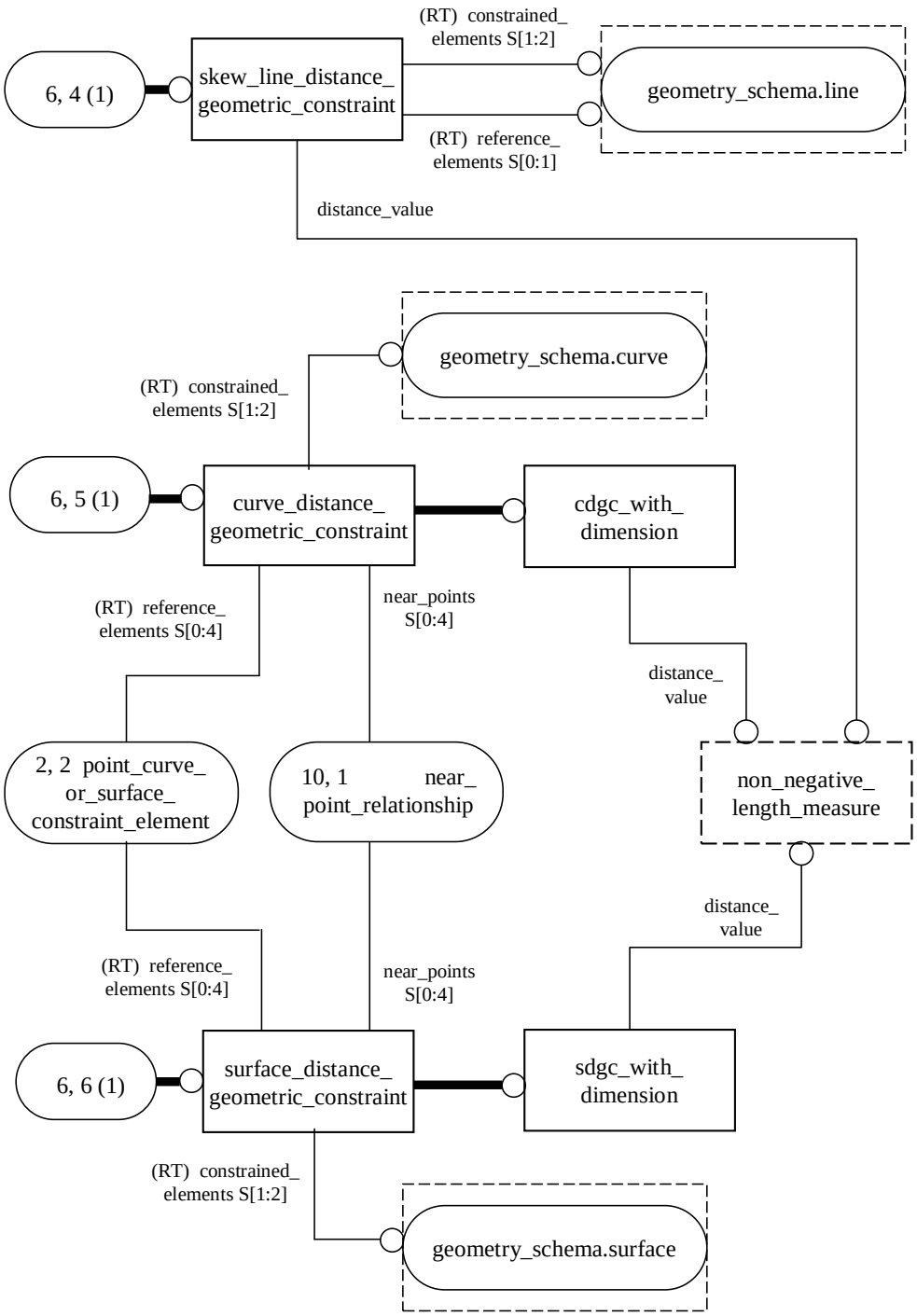

Figure D.10 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (6 of 10)

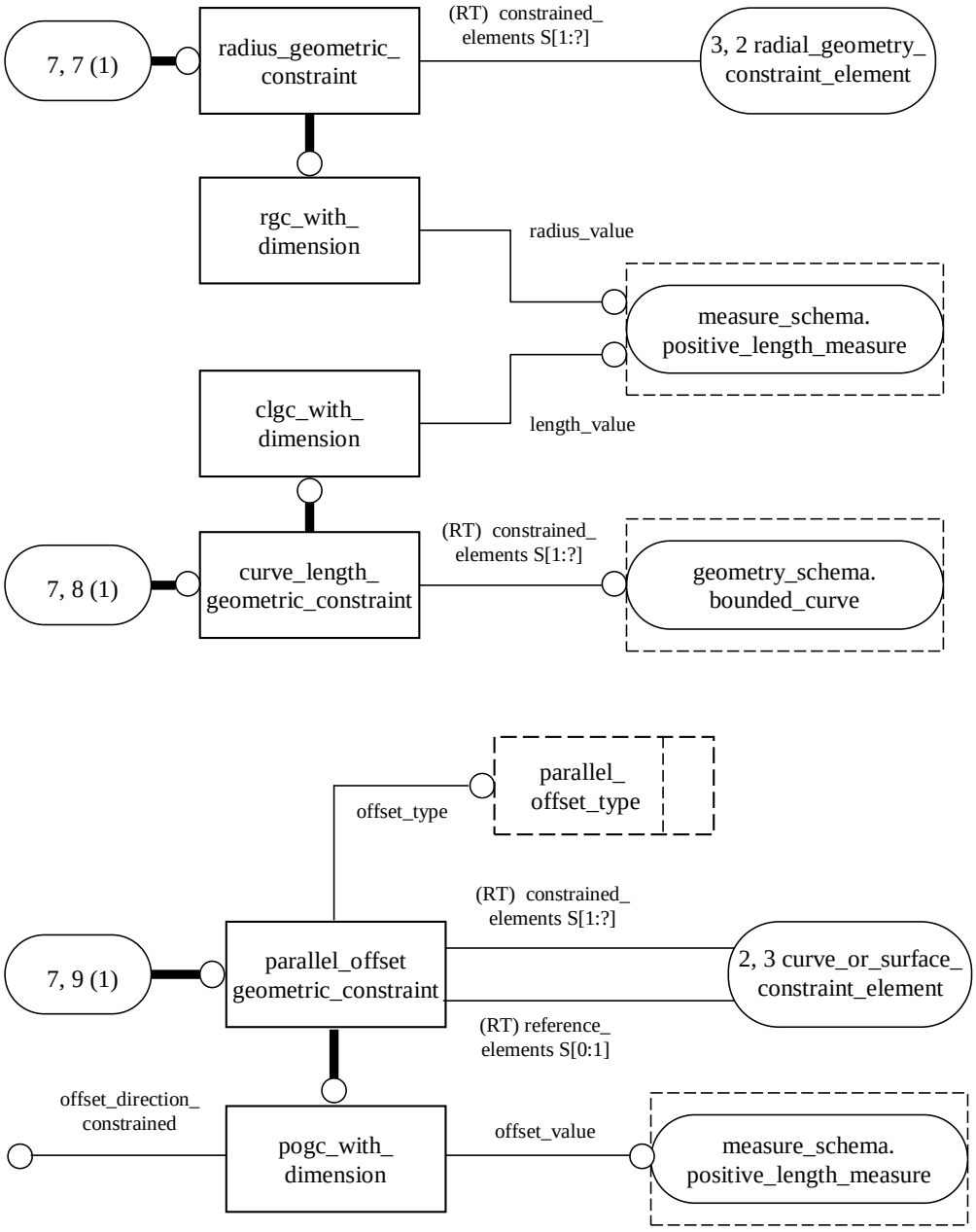

Figure D.11 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (7 of 10)

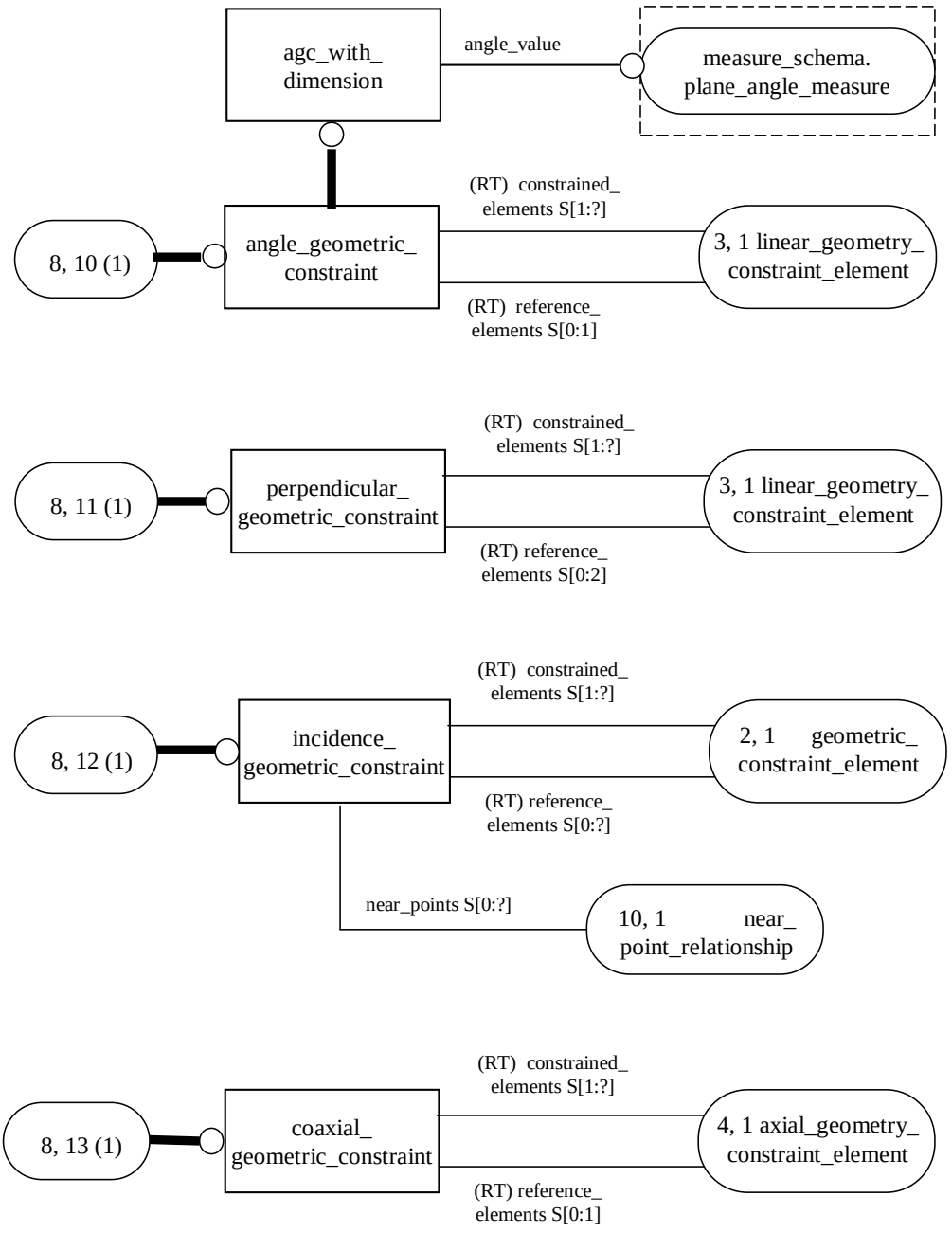

Figure D.12 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (8 of 10)

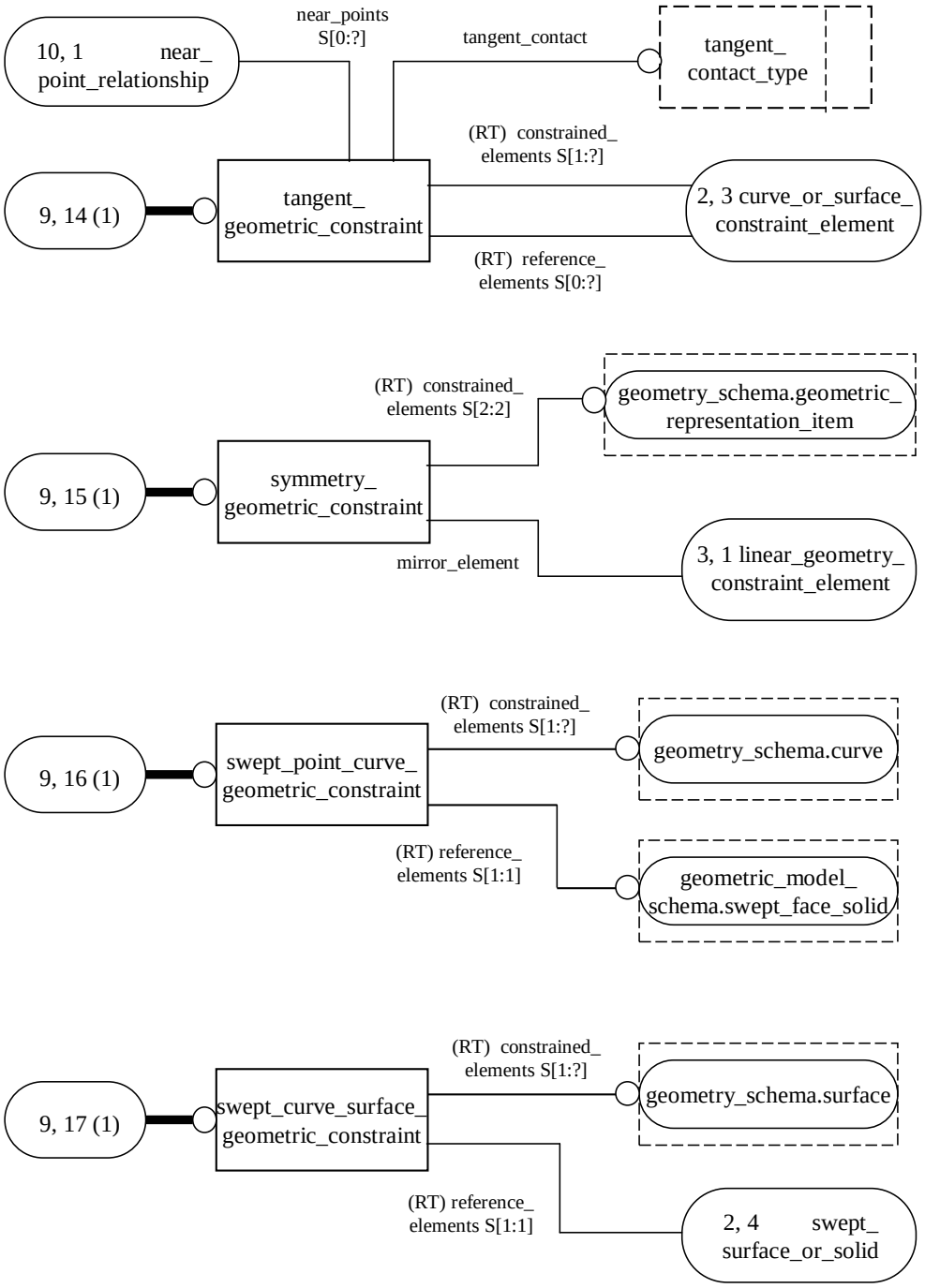

Figure D.13 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (9 of 10)

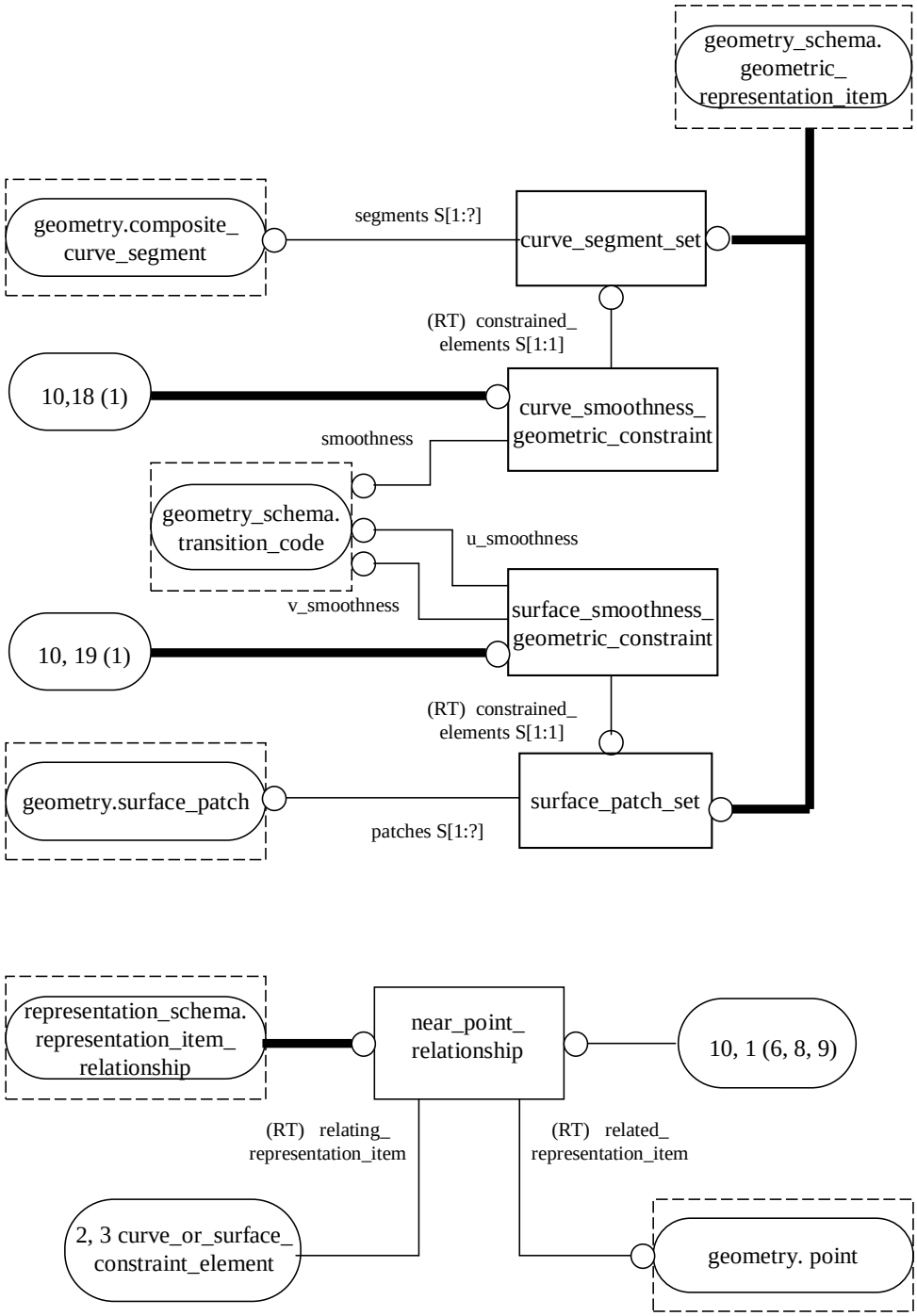

Figure D.14 – EXPRESS-G diagram of the explicit_geometric_constraint_schema (10 of 10)

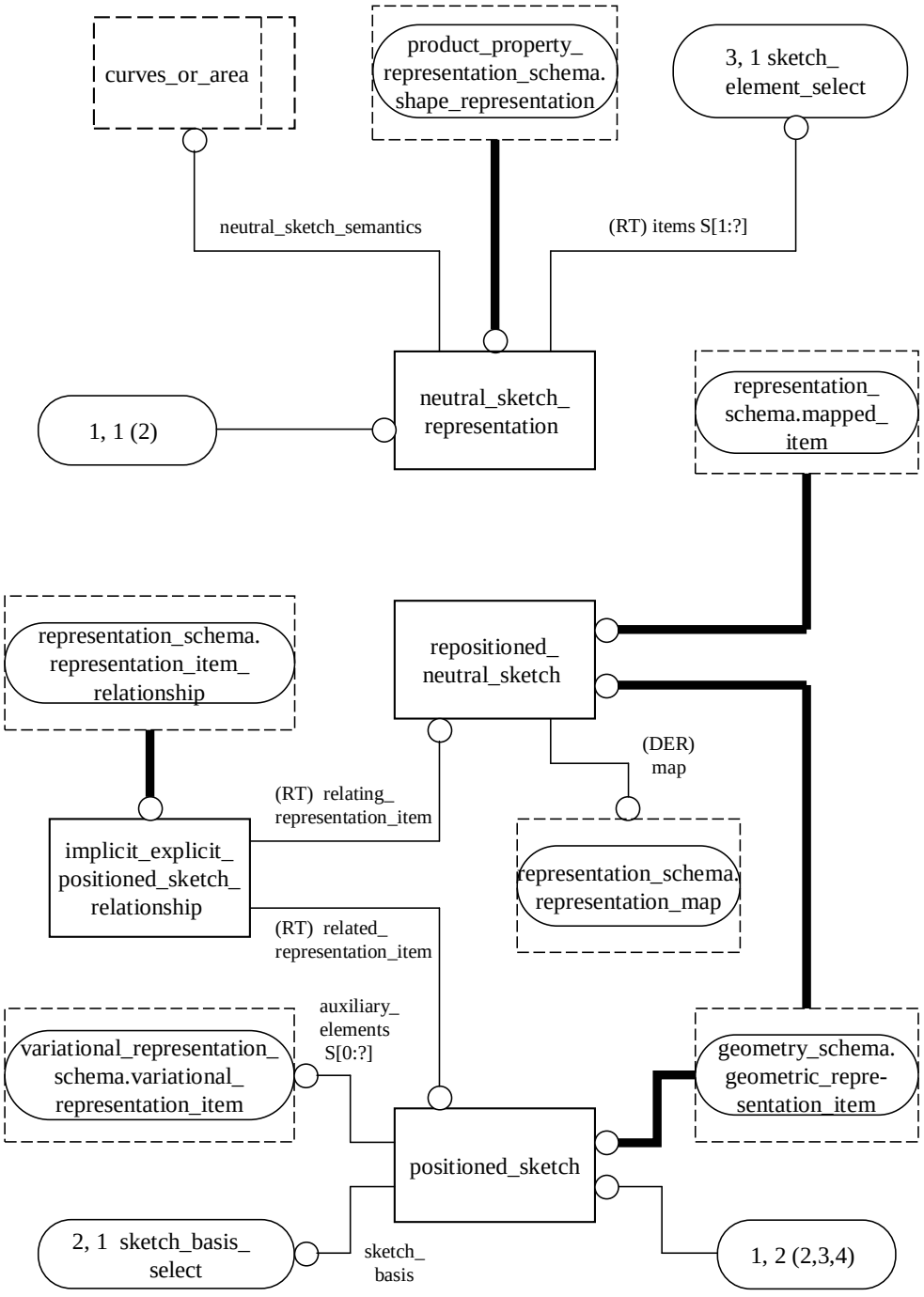

Figure D.15 – EXPRESS-G diagram of the sketch_schema (1 of 4)

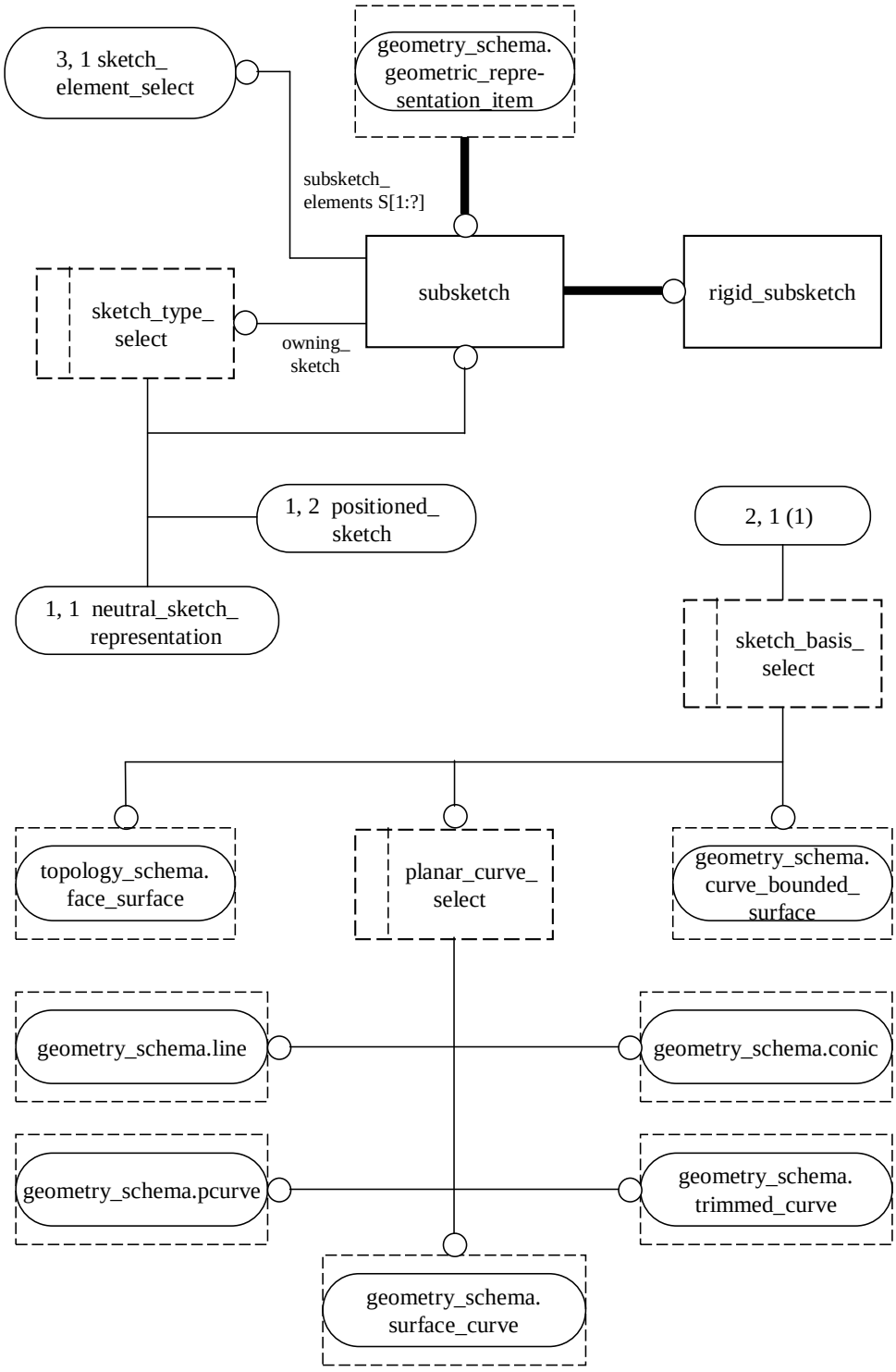

Figure D.16 – EXPRESS-G diagram of the sketch_schema (2 of 4)

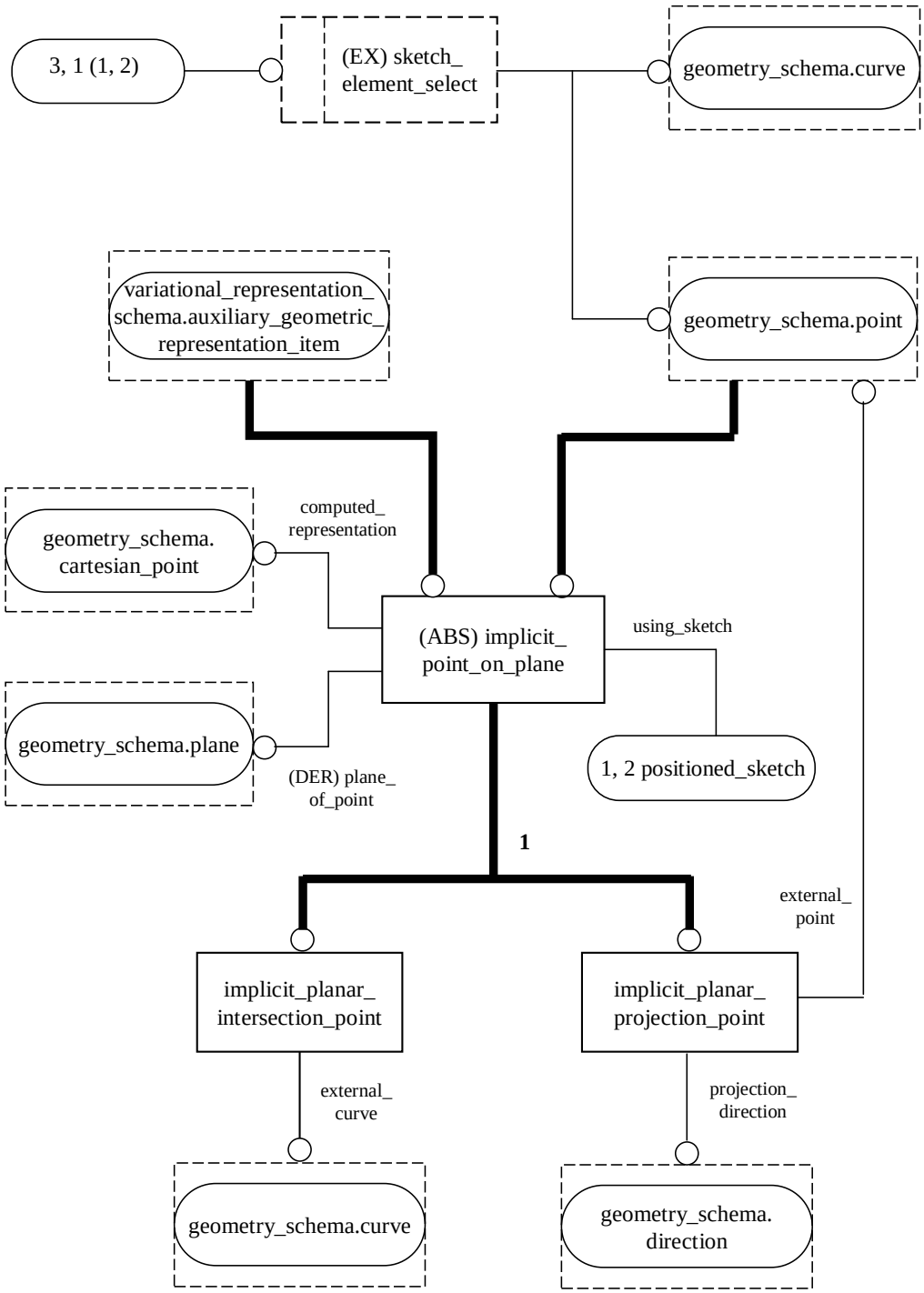

Figure D.17 – EXPRESS-G diagram of the sketch_schema (3 of 4)

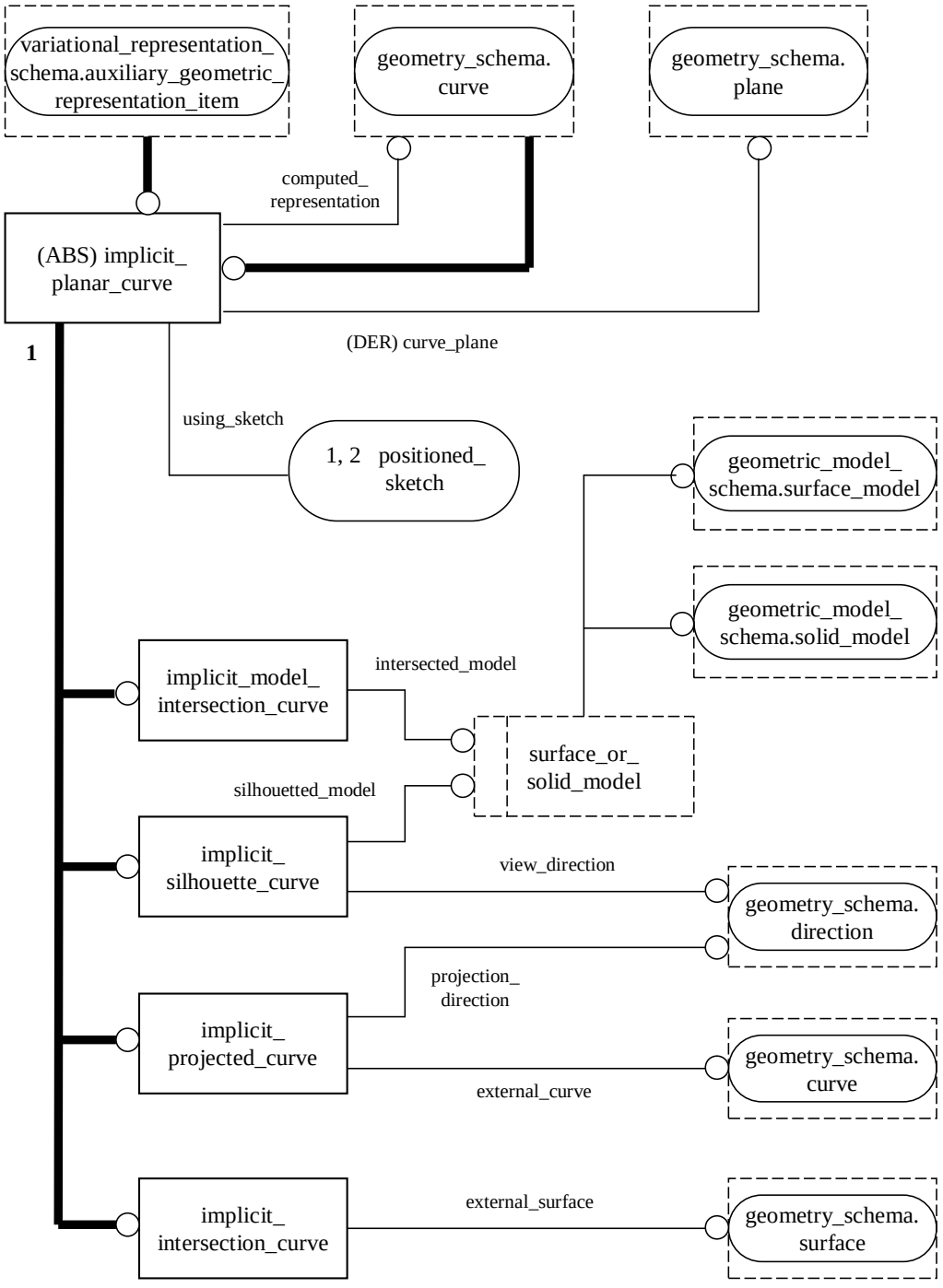

Figure D.18 – EXPRESS-G diagram of the sketch_schema (4 of 4)

Annex E
(informative)

Technical discussions

E.1 Role of parameterization and constraints in procedural and hybrid representations

Modern CAD systems generate product models containing parameterization and geometric constraints. Such models can easily be edited for a variety of purposes, for example design optimization. Parameterization, the association of variables with dimensional or other values in the model, provides an indication of what it is permissible to change. Constraints, often specified as relationships between geometric or topological elements, provide invariant characteristics in the model, usually in the interests of maintaining product functionality during modification. The combination of the two capabilities allows the designer to explore options and to respond to feedback from downstream applications, such as manufacturing or maintenance. Further, it provides flexibility for cooperative and concurrent engineering.

Parameterization and constraints may be implemented in CAD systems in either of two ways:

a) The first method is by associating explicitly represented parameters with dimensions in a model of the boundary representation or similar type, and defining constraints as relationships between sets of elements of that model. The model elements concerned will typically be instances of geometric entities such as are specified in ISO 10303-42. This part of ISO 10303 provides the appropriate capabilities for capturing explicit parameterization (see clause 4) and general explicit constraints (see clauses 5 and 7).

b) The second method is based on constructional history. The *procedural* or *history-based* modelling approach represents a shape by the sequence of operations used in generating it. In such a model, parameters arise as input variables of constructional operations. The shape model may therefore be edited by changing the values of those inputs and re-running the constructional history. Constraints may be implicit in constructional procedures; for example, a procedure may be provided for generating a plane parallel to another plane, in which case parallelism results automatically whenever that procedure is invoked. A model defined purely in terms of its constructional history does not have explicit geometric elements, since such elements are not called into existence until the specified procedures are actually performed. ISO 10303-55 (‘Procedural and hybrid representation’) provides the capability for capturing and exchanging procedural models.

Most major CAD systems use a combination of the two approaches described above. Generally, the primary model is basically of the procedural type, but a secondary explicit model is also generated, which is used during a design session to display a picture of the modelled product, allowing the user to interact with the system by picking model elements such as edges or faces from the screen for modification or for use in further constructional operations (see the paper by Rappoport [5]). The explicit model is also used internally by the system for analysis purposes (for example, in the calculation of mass properties) and in the performance of internal geometric calculations that occur during the modelling process.

To generate an explicit model from a procedural model it is necessary to perform the specified sequence of constructional operations. Most modern CAD systems use their native procedural representations to drive a solid modeler of the boundary representation type in building the explicit model. When a model is edited in a procedural system, the explicit model is usually discarded and a new one created by re-running the modified constructional history.

This part of ISO 10303 provides for the capture and transfer of explicitly represented parameters and constraints as used in the first of the two modelling approaches described earlier in this clause. However, it also provides important capabilities in the transfer of procedural or hybrid models, as described below. Particular application areas include

Two-dimensional sketches: These are often represented in terms of explicit geometric elements and used as the input to procedures for extrusion or rotation that implicitly define three-dimensional shapes in history-based models. Explicit dimensional or other constraints may be imposed between elements of such a sketch;

Assembly models composed of explicitly represented constituent models: For this application the constituent models are treated as rigid, and constraints imposed between geometric elements of different constituents are used in the relative positioning and orientation of those constituents with respect to each other.

Inter-feature relationships: Dimensional or other types of explicit constraints may also be used in the positioning and orientation of features in a part model with respect to one another. The necessary relationships will often be defined between datum elements (reference points, lines or planes) associated with those features.

The first two of these applications are concerned with explicit models, although explicit sketches with parameterization and constraints are often used as inputs to constructional operations in hybrid models. However, the third application provides an example of how explicit inter-feature relationships may be defined in a shape model that is basically of the procedural type. This may be achieved through a different use of the use of the hybrid approach, as illustrated in the following example. For simplicity, the example is expressed in terms of a Boolean operation between CSG primitives rather than in terms of an inter-feature relationship in the more general CAD sense, but the method is equally applicable in that case. The key lies in the use of a hybrid model that is basically procedural but also contains explicit elements supporting definitions of the constructional operations used.

EXAMPLE Consider a solid defined by the Boolean union of two cylinders of finite length. It requires three operations for its procedural definition, the creation operations for the two cylinders plus the union operation. Now suppose that the ISO 10303-42 entity **right_circular_cylinder** is used for the creation of the cylinders (see ISO 10303-55 for the underlying mechanism and for further examples). Apart from a label giving it a name, the cylinder entity has three attributes. The first, of type **axis1_placement**, defines the axis of the cylinder, requiring a point and a direction for its definition. The second and third attributes are of type **positive_length_measure**; they define the height and radius of the cylinder, and their values will in effect be supplied as input arguments to the creation procedures. The associated points and lines, however, are explicitly represented supporting information that is referenced by the creation operations, and it is therefore possible to impose, for example, an explicit perpendicularity constraint between the directions of the axial lines of the two cylinders.

Parameterization and constraints, together with the construction history of a model, constitute what is known as *design intent*. The association of some or all of this information with a model allows it to be edited in a manner more or less consistent with the original mode of model creation. If a model is exchanged between systems without such information it is difficult or impossible to edit it effectively in the receiving system, because details of the original construction process are totally absent. On the other hand, if design intent information is available in the transmitted model, ease of editing can in principle be preserved after the transfer.

CAD systems generally allow the explicit model they create to be saved in a file or to a database, even though their primary native model representation may be procedural in nature. The file format used may conform to one of the early ISO 10303 application protocols such as ISO 10303-203 [1], in which case the model exchanged is the secondary explicit model which contains no design intent. The use of this

part of ISO 10303 by an application protocol allows the enhancement of such a model by the addition of explicit parameterization and constraint information, providing partial design intent. However, as mentioned above, the full transfer of design intent requires the use of this part of ISO 10303 together with ISO 10303-55, which permits the exchange of procedural and hybrid models. The example above shows that these two resources work together effectively in the context of hybrid models.

To summarize, the purpose of this part of ISO 10303 is to enable the exchange of a significant part of the design intent information referred to above, by providing the means to associate parameters and constraints with elements of explicit models, or with explicit elements of hybrid models. Whereas use of the earlier integrated generic resources of ISO 10303 (those having part numbers in the range 41 – 47) allows only the transfer of a static representation of a model, this resource provides the means to capture and transfer a subset of the information that is used to govern the*behaviour* of the model in the sending system. If this information can lead to the automatic provision of the same or similar behaviour under editing operations in a receiving system, great benefit will result for applications using the model subsequent to the transfer.

E.2 Justification of representational choices made in this part of ISO 10303

This clause briefly outlines the reasons for the use of certain specific modelling techniques in this part of ISO 10303.

E.2.1 Non-binary constraints

The supertype entity **explicit_constraint**, defined in clause 5.4.1, represents a general explicit constraint as a relationship between two sets of elements, one of them possibly empty. The sets are those of **constrained_elements** and **reference_elements**, respectively. This formulation permits the capture, by appropriate specialization, of various specific types of constraints, including the following important cases:

— a directed or undirected dimensional constraint between two geometric elements (example: assertion of the value of the angle between two planes);

— an undirected constraint applying to many elements (example: assertion that members of a set of ten lines are all mutually parallel);

— a directed constraint with a single reference element (example: assertion that members of a set of surfaces of revolution are all coaxial with a specified cylinder);

— a directed constraint with many reference elements (example: assertion that a constrained curve interpolates a set of reference points);

— a hybrid, partially directed constraint between values of two sets of **model_parameter** instances (example: see clause 5.4.6).

At an early stage in the development of this part of ISO 10303, consideration was given to the use of the ISO 10303-43 entity data type **representation_item_relationship** for modelling explicit constraints. This captures binary relationships between instances of **representation_item**. Its use for the purposes described above would require either

a) any constraint involving more than two elements to be broken down into a set of binary constraints, or

b) any constraint involving more than two elements to be defined as a binary relationship between two *sets* of elements (such sets would need to be typed as **representation_item** to permit this approach).

A problem with the first approach is that it could potentially lead to large sizes for exchanged neutral representations. A single undirected constraint requiring ten lines to be parallel to each other is logically equivalent to 45 separate binary constraints between all possible pairs of lines. It would be necessary to introduce new entity data types to capture the fact that such sets of separate constraints are logically related. Furthermore, not all constrained situations can be expressed in terms of binary relations between the elements involved.

EXAMPLE A mathematical relation between several variables, which can be captured using the methodology of clause 5 as a **free_form_relationship** constraint between instances of **model_parameter** subtypes, cannot in general be decomposed into binary relations between pairs of parameters.

The second approach has the difficulty that the set of reference elements may be empty, in which case the constraint is a unary one on a set of constrained elements rather than a binary relation between two sets of elements. Clearly **representation_item_relationship** could not be used in this case, and it would be necessary to define a separate class of unary constraints. If this were done, most constraints defined in this part of ISO 10303 would then need to be defined in both unary and binary forms, which would greatly increase the length and complexity of the document.

For the reasons given above, it was decided not to use **representation_item_relationship** for the representation of explicit constraints. The modelling approach adopted for these entities in this part of ISO 10303 combines the virtues of flexibility with comparative simplicity. It has the additional advantage of minimising the number of instances of constraint entities in neutral representations of exchanged models.

E.2.2 The modelling of variational representations

As defined in clause 6 of this part of ISO 10303, a variational representation consists of a non-variational representation together with a collection of variational items. The variational information may be used to govern the behaviour of a transferred model when it is edited in a receiving system. The only restriction on the nature of the non-variational 'current result' representation, the result of evaluating the variational model for the current values of all its parameters, is that it shall not itself have a variational nature. This is enforced by WHERE rules on the entity data type **variational_current_representation_relationship** defined in clause 6.3.4, which permit no instances of **variational_representation_item** to occur in the current result.

It is therefore possible to use an instance of *any* non-variational type of representation as the current result of a variational representation. This includes all those types of representation defined in parts of ISO 10303 issued prior to the publication of this part of the standard. To base a variational representation on a specific non-variational representation it is only necessary to create an instance of **variational_representation** having the same set of representation items as the current result, supplemented by all the instances of **variational_representation_item** that apply directly or indirectly to the elements of the current result. The two instances of representation must then be linked via an instance of **variational_current_representation_relationship**.

This mechanism was chosen as being simple, flexible, and upwardly compatible with pre-existing parts of ISO 10303.

E.3 Application-related sketches with specific geometric forms

Some applications, for example building and construction, frequently require the modelling of extrusions with specific geometric forms.

EXAMPLE Steel reinforcing rods typically have circular cross-sections, and many structural beams have cross-sections in the form of an **I** or a **T**.

In order to allow the convenient modelling of such extrusions, application protocols may define sketches that have these or other particular forms. A suitable means for achieving this is suggested in what follows.

The entity **neutral_sketch_representation** (see clause 8.4.9) has a derived attribute **sketch_geometry** that references a set of instances of **sketch_element_select**, points and curves defining the geometry of the sketch. If it is desired to define a neutral sketch representation having the form of a circle, it is only necessary to create a subtype of **neutral_sketch_representation** with a WHERE rule restricting the content of that set to a single circle. For more complex 2D shapes such as I- or T-profiles it will first be necessary to create entities defining these shapes in terms of their salient dimensions, after which the same procedure can be used.

A similar approach may be used for specialized forms of **positioned_sketch** (see clause 8.4.10). In this case appropriate subtypes of **curve_bounded_surface** can be specified to define the geometry of the planar shape to be extruded.

Annex F
(informative)

Examples

F.1 Examples of the intended usage of the ISO 10303-108 mechanism for linking parameters with attributes of entity instances

The basic principle for the referencing of attributes and the binding of **model_parameter** instances to them was illustrated in clause 4.2.2. It is recommended that the earlier example be studied before reading the following additional examples, because it provides some of the necessary preliminary understanding. The further illustrations show the use of free-form constraints for the specification of mathematical relationships between instances of **model_parameter** subtypes.

F.1.1 Example 1

In this and the following example, entity data types from the ISO 10303 integrated generic resources are treated as though they are instantiable elements of an application protocol. This first example illustrates the modelling of a reciprocal constraint requiring the radius of circle C6 to be 2.5 times that of circle C5 (and conversely, the radius of C5 to be 0.4 times that of C6). Thus both **model_parameter** instances required are constrained elements (see clause 4.2.1), since if the value of one is changed in the receiving system the other is required to adjust correspondingly.

In the example, all entity data types occurring are defined in this part of ISO 10303 unless otherwise mentioned. The ISO 10303-21 transfer file must first contain definitions of the circle instances. The necessary entity data types are specified in ISO 10303-42. The current values of the radii of the circles are 10.0 and 25.0 respectively:

```
#200 = AXIS2_PLACEMENT_3D(.....);
#210 = AXIS2_PLACEMENT_3D(.....);
.....
#500 = CIRCLE('C5', #200, 10.0);
#510 = CIRCLE('C6', #210, 25.0);
```

Next, the file must include two instances of **bound_model_parameter**, each with domain specified by an instance of **finite_real_interval** (all entities defining such domains are taken from ISO 10303-50):

```
#520 = FINITE_REAL_INTERVAL(8.0, .CLOSED., 15.0, .CLOSED.);
#530 = BOUND_MODEL_PARAMETER('P1', #520, *, 'C5 RADIUS', *);

#540 = FINITE_REAL_INTERVAL(20.0, .CLOSED., 30.0, .CLOSED.);
#550 = BOUND_MODEL_PARAMETER('P2', #550, *, 'C6 RADIUS', *);
```

Now the binding must be established between these two **bound_model_parameter** instances and the radius attributes of the two circles. The mechanism for achieving this is defined in ISO 13584-20, and the relations between the entities defined there and their subtypes as defined in this part of ISO 10303 are depicted in an EXPRESS-G diagram in clause F.1.3 following the examples. For each binding, an instance of **environment** (subtyped in this part of ISO 10303 as **bound_parameter_environment** and **unbound_parameter_environment**) must exist. The **environment** entity data type has two attributes, as follows:

syntactic_representation: a reference to a mathematical variable (in the present case an instance of **bound_model_parameter**);

semantics: a reference to an instance of the ISO 13584-20 entity data type **variable_semantics**, which is intended to indicate the significance of the variable in its modelling context.

In this part of ISO 10303 a subtype of **variable_semantics** called **instance_attribute_reference** is provided. This simply declares the specific attribute (by name) and the specific entity data type instance (by reference) that the **model_parameter** is to be associated with. The next fragment of the exchange file creates this association:

```
#560 = INSTANCE_ATTRIBUTE_REFERENCE
         ('GEOMETRY_SCHEMA.CIRCLE.RADIUS', #500);
#570 = BOUND_PARAMETER_ENVIRONMENT(#530, #560);
#580 = INSTANCE_ATTRIBUTE_REFERENCE
         ('GEOMETRY_SCHEMA.CIRCLE.RADIUS', #510);
#590 = BOUND_PARAMETER_ENVIRONMENT(#550, #580);
```

Note that the **radius** attribute is specified in fully qualified form, as required by an informal proposition applying to the **instance_attribute_reference** entity data type.

The parameter bindings having been established, an instance of **free_form_relation** is now required, to model the mathematical relationship between the values of the **bound_model_parameter** instances. The constraint instance is `#600` below, in which both parameters occur as constrained elements. The first two attribute values of this instance correspond to name and textual description attributes inherited from the supertypes **representation_item** and **explicit_constraint** respectively. These are followed by the set of (two) constrained elements, the (empty) set of reference elements and a reference to the **constraining_expression**, represented by `#610`. This and the two following instances build a relationship asserting that the value of parameter `#550` is 2.5 times that of the parameter `#530`:

```
#600 = FREE_FORM_RELATION('FFR1', 'RADIUS-RELATION',
         (#530, #550), (), #610);
#610 = COMPARISON_EQUAL((#550, #620));
#620 = MULT_EXPRESSION((#630, #530));
#630 = REAL_LITERAL(2.5);
```

Instances `#610` – `#630` are of entities defined in the **expressions_schema** of ISO 13584-20. The modelled relationship is true in the model as transmitted; the existence of the constraint is intended to ensure that the relationship can be maintained if the model is edited in the receiving system.

NOTE 1 The first attribute value in instances `#560` and `#580` corresponds to the attribute **attribute_name**, declared in the **instance_attribute_reference** entity data type defined in clause 4.4.6. In the present context this value must be interpreted as an *intelligent string*, i.e., an EXPRESS string with semantics. This has obvious implications for implementers; analysis of the string by reference to the schemas in current use is necessary to enable the appropriate references to be established in the model. However, that appears to be unavoidable for the transfer of models containing specific relationships between parameters and instance attributes, regardless of the precise details of how the relationships are represented in the underlying EXPRESS schema.

NOTE 2 It will be necessary for the label referred to above to allow references to attributes of underlying entities.

EXAMPLE The string `'GEOMETRY_SCHEMA.TRIMMED_CURVE.BASIS_CURVE.RADIUS'` could be used as the **attribute_name** value in constraining the radius of a circular arc expressed as an ISO 10303-42 **trimmed_curve** instance.

NOTE 3 It will similarly be necessary to provide for references to individual members of aggregate-valued attributes.

F.1.2 Example 2

The second example shows the use of **unbound_model_parameter**. The details are much the same as in the previous example, but it is necessary to use a different subtype of ISO 13584-20 **environment** for an unbound parameter. The example modelled here was used in clause 4.2.1 to illustrate a situation where an unbound parameter may be used. To recapitulate, it is desired to constrain the height h and radius r of an instance of ISO 10303-42 **right_circular_cylinder** to satisfy $h = t^2 + 1,\ r = 3t - 2$, where t is a parameter that does not represent any physical quantity in the model. In this case the model parameters corresponding to h and r are required to be bound to attributes of the **right_circular_cylinder** instance, but that corresponding to t is unbound.

The circular cylinder is represented by the ISO 10303-42 entity data type instances

```
#340 = AXIS1_PLACEMENT(...);
#350 = RIGHT_CIRCULAR_CYLINDER('CYL1', #340, 10.0, 7.0);
```

Here the current values of the radius and height are specified as 10.0 and 7.0 respectively. Two **bound_model_parameter** instances are next defined and bound to the relevant attributes of `#350`, as shown in the previous example:

```
#360 = FINITE_REAL_INTERVAL(2.0, .CLOSED., 20.0, .CLOSED.);
#370 = BOUND_MODEL_PARAMETER('H', #360, *, 'CYL_HT', *);
#380 = INSTANCE_ATTRIBUTE_REFERENCE
         ('GEOMETRIC_MODEL_SCHEMA.RIGHT_CIRCULAR_CYLINDER.HEIGHT',
          #350);
#390 = BOUND_PARAMETER_ENVIRONMENT(#370, #380);

#400 = FINITE_REAL_INTERVAL(1.0, .CLOSED., 12.0, .CLOSED.);
#410 = BOUND_MODEL_PARAMETER('R', #360, *, 'CYL_RAD', *);
#420 = INSTANCE_ATTRIBUTE_REFERENCE
         ('GEOMETRIC_MODEL_SCHEMA.RIGHT_CIRCULAR_CYLINDER.RADIUS',
          #350);
#430 = BOUND_PARAMETER_ENVIRONMENT(#410, #420);
```

Next the **unbound_model_parameter** corresponding to t is defined, and **unbound_parameter_semantics** is instanced rather than **instance_attribute_reference**. It may be noted that **unbound_parameter_semantics** has no attributes, and that the instance `#460` therefore conveys no information, but its presence is required by the structures defined in ISO 13584-20.

```
#440 = REAL_INTERVAL_FROM_MIN(2.0, .CLOSED.);
#450 = MODEL_PARAMETER('T', #440, *, 'FREE_PARAM', 3.0);
#460 = UNBOUND_PARAMETER_SEMANTICS;
#470 = UNBOUND_PARAMETER_ENVIRONMENT(#450, #460);
```

Note that, whereas in an instance of **bound_model_parameter** the value of the parameter is represented by an asterisk (denoting a derived value), in an instance of **unbound_model_parameter** it is given explicitly (as 3.0 in the present case).

Finally, the relationships between the parameters are established by means of two **free_form_assignment** instances, in both of which the parameter #450 corresponding to t plays the role of a reference element:

```
#480 = FREE_FORM_ASSIGNMENT('FFA1', 'HEIGHT-VALUE',
         (#370), (#450), #490);
#490 = PLUS_EXPRESSION((#500, #510));
#500 = MULT_EXPRESSION((#450, #450));
#510 = REAL_LITERAL(1.0);

#520 = FREE_FORM_ASSIGNMENT('FFA2', 'RADIUS-VALUE',
         (#410), (#450), #530);
#530 = MINUS_EXPRESSION((#540, #550));
#540 = MULT_EXPRESSION((#560, #450));
#550 = REAL_LITERAL(2.0);
#560 = REAL_LITERAL(3.0);
```

F.1.3 Relationship between ISO 10303-108 and ISO 13584-20

Figure F.1 on the following page illustrates the primary relationships between the entities defined in this part of ISO 10303 for binding parameters to attribute values and the supertypes of those entities as defined in ISO 13584-20. The presence of an intervening subtype, **maths_variable** as defined in ISO 10303-50, is also shown.

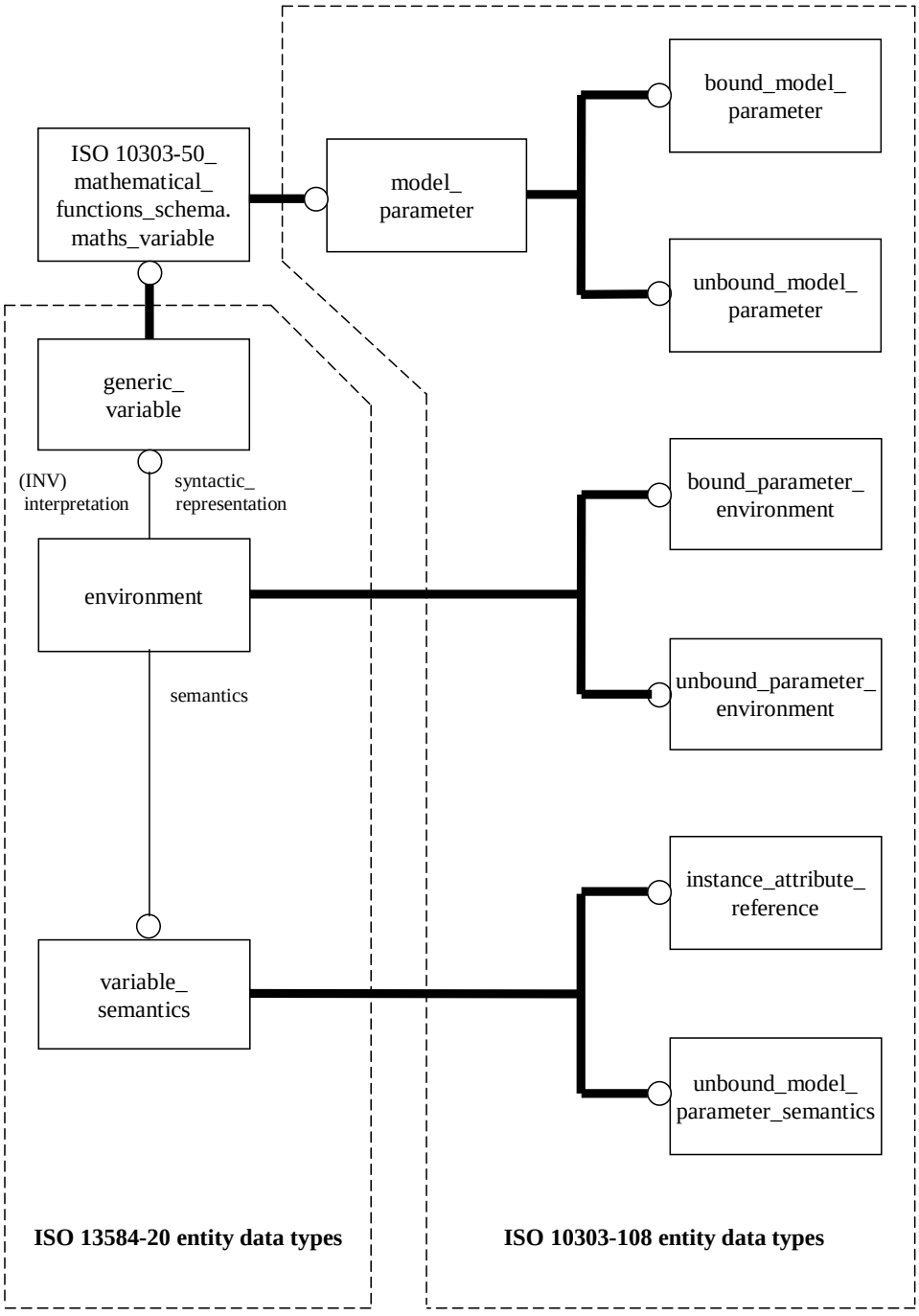

Figure F.1 – Key relationships between ISO 10303-108 parameterization_schema and ISO 13548 generic_expressions_schema

F.2 Example of a two-dimensional sketch

Clause 8 defines representations for sketches, sometimes known as profiles — two-dimensional geometric configurations often used in the generation of surfaces or volumes. Figure F.2 below illustrates a simple sketch, in this case a closed figure composed of line and circular arc segments, as listed below:

l_1: line segment from (1,0) to (4,0)
l_2: line segment from (1,4) to (8,4)
l_3: line segment from (0,1) to (0,3)
l_4: line segment from (4,0) to (8,4)
c_1: circular arc with center (1,1), start point (1,0), end point (0,1)
c_2: circular arc with center (1,3), start point (0,3), end point (1,4)

The annotations are present purely to aid in the following discussion, and should not be regarded as integral to the sketch.

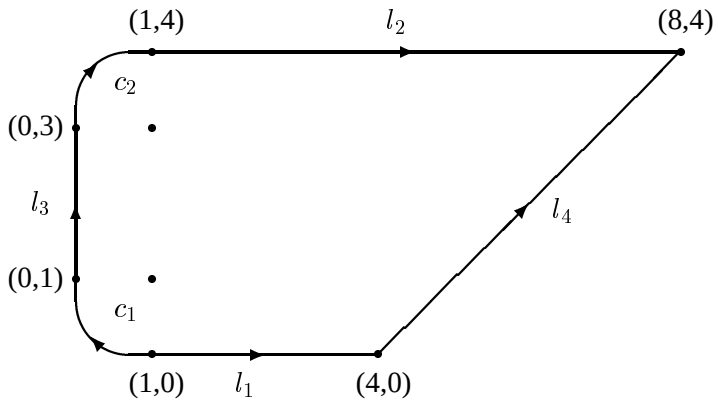

Figure F.2 – A simple sketch composed of line segments and circular arcs

Some constraints which could be applied to this profile include

a) l_1 and l_2 are parallel and 4 units apart

b) the (oriented) angle between l_1 and l_4 is $45°$

c) l_3 is perpendicular to l_1

d) the initial point of l_1 is fixed at (1,0)

The entity data types **pgc_with_dimension**, **angle_geometric_constraint**, **perpendicular_geometric_constraint** and **fixed_element_geometric_constraint**, as defined in clause 7 of this part of ISO 10303, can be used to represent these constraint relationships, the first three of which will be specified between the basis curves of the curve segments. Such constraints may only be present if the sketch belongs to an instance of **variational_representation**. The intention in associating constraints with the geometric model is to ensure that the relationships can be maintained if the sketch is edited following a transfer between systems. It may be desired, for instance, to change the distance between l_1 and l_2 or the radii of the rounded corners.

Clause 8 of this part of ISO 10303 provides several possibilities for the representation of sketches. These are listed below, with remarks applicable to the various specific cases:

neutral_sketch_representation: The geometry of the sketch is defined by a set of geometric elements. Any curves involved will generally be bounded, and constraints on them will have to be applied between the corresponding unbounded basis curves. No topological connectivity is defined between the elements in the set, and in the variational case coincidence between the end points of bounded curves will need to be enforced through the use of the **incidence_geometric_constraint** specified in clause 7.4.22 of this part of ISO 10303. In addition to the constraints on parallelism, perpendicularity etc. mentioned previously, the tangency relationships occurring between line and circular arc elements in the sketch must be captured by instances of **tangent_geometric_constraint** as defined in clause 7.4.24.

positioned_sketch based on a planar face_surface: In this case the topology of the sketch is explicitly recorded in terms of the edges and vertices of the associated face. The topological relationships will not be affected by geometric changes made to the sketch following a model exchange, and will therefore have effects analogous to those of explicit geometric constraints — elements that are connected in the sending system should remain connected under modification in the receiving system. However, a **face_surface** instance does not record continuity of the face boundary at vertices where edges join. If it is desired, for example, that two edges that are tangential at a vertex in the originating system should remain tangential if the sketch geometry is modified, than instances of **tangent_geometric_constraint** should be imposed to ensure this.

positioned_sketch based on a planar curve_bounded_surface: The geometry of the sketch consists of one or more closed curves, one of them possibly being the outer boundary of the bounded surface region defined. The closed curves may be, for example, instances of **pcurve** defined on the plane concerned, and the underlying basis curves may be of type **b_spline_curve** or **composite_curve**. A composite curve would be appropriate for the representation of the illustrated sketch. In this case the segments of the curve will be connected at their end-points — this is enforced by the ISO 10303-42 definition. Also, each curve segment will have an attribute defining the order of continuity at its end point with the following curve segment (this may be simple positional continuity, tangent continuity or curvature continuity). If the sketch is edited in a receiving system the segments must remain connected, by definition, since what results will be another composite curve. The topology is not modelled explicitly, but is implicit in the definition of this type of curve. However, there is no guarantee that the order of continuity at the junction points will be preserved; the attributes that define this in the sending system do not have the necessary semantics to ensure its preservation. The constraint **curve_smoothness_geometric_constraint** defined in clause 7 can be used to enforce preservation of order of continuity between segments if the sketch is edited following a transfer. This will ensure, for example, that the centres of the corner arcs will adjust their positions to ensure continued tangency of the arcs with their adjacent line segments.

positioned_sketch based on a planar curve: There are several possible types of curves that may be used in this case. As indicated in the previous examples, the types of explicit constraints that will need to be imposed to make the sketch editable in the receiving system will depend upon the presence or absence of implicit constraints in whatever form of curve definition is used.

F.3 Usage of ISO 10303-108 for the representation of incompletely defined models

Some purely procedural approaches to CAD modelling allow the definition of families of geometric shapes through the use of ‘shape macros’.

EXAMPLE 1 Consider a bolt with a hexagonal head and a threaded shank. At a very basic level, any bolt in the family of such bolts can be defined in terms of a few dimensional parameter values:

— shank length;

— shank diameter;

— length of threaded portion of shank;

— head thickness;

— distance between parallel flats of the hexagon.

Assignment of values to all of these quantities results in the specification of a particular size and shape of bolt. However, the underlying definition of the family specifies no detailed geometry; it merely provides a set of placeholders into which values must be inserted. A procedure for determining the geometry of a bolt defined by those values may be provided. It is also possible to define a subfamily of bolts by giving values to some of the parameters but leaving others undefined – the dimensions of the head and the diameter of the shank may be specified, for example, leaving the lengths of the shank and its threaded portion indeterminate.

One modelling methodology that lends itself to this kind of approach is constructive solid geometry (CSG) with parameterization of the dimensions of volumetric primitives.

NOTE In some applications, (e.g., in catalogues of standard parts), the conceptual model takes the form of a printed diagram, with a table of available dimensional values. In such cases there is no generative procedure.

The capabilities defined in this part of ISO 10303 do not by themselves permit the definition of pure shape macros of the kind illustrated in the example above. However, they do allow a closely related capability, based on the use of the concept of the **variational_representation** as defined in clause 6. The primary difference in the two approaches is that the one based on **variational_representation** requires the provision of an explicit model of one member of the family of bolts. This 'current result' model can be embedded in a wrapper of variational information that may, for example,

— associate instances of **bound_model_parameter** with the salient dimensions occurring in the current result;

— impose constraints on geometric elements of the current result, so that if it is modified the functionality of the bolt is retained.

Domains of validity can be associated with the dimensional parameters; one possibility would be to allow only certain discrete values of each of those dimensions, to limit the number of valid members of the family. The constraints imposed could require the shank axis to be perpendicular to the head, the flats of the hexagon to occur in parallel pairs and to be equidistant from the shank axis, the angles between adjacent flats to be 120°, and so on.

Whereas the shape macro approach generates a shape *ab initio*, based on an underlying conceptualization of the class of shapes concerned, the variational approach starts with an explicitly defined shape and defines permissible variations of it. The latter methodology allows the modelling of incompletely defined shapes to the extent that such shapes are derivable from the shape defined by the current model through variation of parameter values within their admissible domains, subject to any constraints that may be imposed between its elements.

EXAMPLE 2 The shape of a washer may be modelled in boundary representation terms by two planar faces, each having the shape of an annulus, connected by inner and outer cylindrical faces. The facilities provided in this part of ISO 10303 may be used to constrain the cylinders to be coaxial and the planar faces to be both mutually parallel and also perpendicular to the axis of the cylinders. Those facilities additionally allow a parameter to be associated with the distance between the planar faces, and two further parameters to be associated with the inner and outer radii of the washer. The current model of the washer may then be regarded as a representative example of the class of circular washers. In this sense it provides, together with its associated variational information, not only a detailed definition of the shape of one particular washer, but also an incomplete definition of all washers in the larger class. That class contains all washers having the same type of surface geometry and topological connectivity as the current result, subject to the same constraints on their geometric elements and conforming to the specified domains of parameter validity. Thus a variational representation of a washer whose current result has thickness 2mm, internal radius 2mm and external radius 5mm may be regarded as an incomplete definition of a washer with thickness 3mm, inner radius 4mm and outer radius 10mm. The second may be obtained from the first by variation of parameter values.

Bibliography

[1] ISO 10303-203:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled design.*

[2] ISO TS 17450:1999, *Geometrical product specification (GPS) — Model for geometric specification and verification.*

[3] HOFFMANN, C.M., *Geometric and Solid Modeling*, San Mateo, CA: Morgan Kaufmann, 1989.

[4] HOSCHEK, J.; DANKWORT, W. (editors). *Parametric and Variational Design*, Stuttgart: Teubner-Verlag, 1994.

[5] RAPPOPORT, A. *Breps as Displayable-Selectable Models in Interactive Design of Families of Geometric Objects*, in *Geometric Modeling: Theory and Practice* (edited by W. Strasser, R. Klein and R. Rau); Berlin: Springer-Verlag, 1997.

[6] SHAH, J. J.; MÄNTYLÄ, M. *Parametric and Feature-based CAD/CAM*, New York: Wiley, 1995.

Index

ICS 25.040.40

Price based on 154 pages