INTERNATIONAL STANDARD

# ISO 10303-507

First edition
2001-02-01

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

## Part 507:
## Application interpreted construct: Geometrically bounded surface

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

*Partie 507: Établissement interprété d'application: Surface limitée géométriquement*



Reference number
ISO 10303-507:2001(E)

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.ch
Web www.iso.ch

Printed 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 3.

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 International Standard may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 10303-507 was prepared by Technical Committee ISO/TC 184 *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

This International Standard is organized as a series of parts, each published separately. The structure of this International Standard is described in ISO 10303-1.

Each part of this International Standard 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 application interpreted construct series.

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

<http://www.nist.gov/sc4/editing/step/titles/>

Annexes A and B form a normative part of this part of ISO 10303. Annexes C and D are for information only.

# Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving.

This International Standard is organized as a series of parts, each published separately. The parts of ISO 10303 fall into one of the following series: description methods, integrated resources, application interpreted constructs, application protocols, abstract test suites, implementation methods, and conformance testing. The series are described in ISO 10303–1. This part of ISO 10303 is a member of the application interpreted constructs series.

An application interpreted construct (AIC) provides a logical grouping of interpreted constructs that supports a specific functionality for the usage of product data across multiple application contexts. An interpreted construct is a common interpretation of the integrated resources that supports shared information requirements among application protocols.

This document specifies the application interpreted construct for the description of geometric shapes by means of geometrically bounded surface models. It includes the geometric resources to define purely geometrically bounded models that consist of elementary and sculptured curves and surfaces.

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

# Part 507: Application interpreted construct: Geometrically bounded surface

## 1 Scope

This part of ISO 10303 specifies the interpretation of the integrated resources in order to satisfy requirements for the representation of geometric shapes by means of geometrically bounded surface models.

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

— 3D points;

— points defined in the parameter space of curves or surfaces;

— 3D curves;

— curves defined in the parameter space of surfaces;

NOTE - Such curves are also known as pcurves or cons, which are acronyms for parametrised curve and curve on surface.

— the elementary curves line, circle, ellipse, parabola, and hyperbola;

— intersection curves;

— polylines that consist of at least three points;

— surfaces;

— the elementary surfaces plane, cylinder, cone, torus, and sphere;

— swept surfaces created by rotation or linear extrusion of a curve;

— sculptured curves and surfaces;

— trimming of curves and surfaces;

— composition of curves and surfaces;

— replication of curves, surfaces, and surface models;

— 3D offsets of curves and surfaces.

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

— unbounded geometry;

— self-intersecting geometry;

— geometry in a 2D cartesian coordinate space;

— replication of points;

— topological entities.

## 2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 8824–1:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

ISO 10303–1:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 1 : Overview and fundamental principles.*

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

ISO 10303–41:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 41 : Integrated generic resources: Fundamentals of product description and support.*

ISO 10303–42:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 42 : Integrated generic resources: Geometric and topological representation.*

ISO 10303–43:1994, *Industrial automation systems and integration – Product data representation and exchange – Part 43 : Integrated generic resources: Representation structures.*

ISO 10303–202:1996, *Industrial automation systems and integration – Product data representation and exchange: – Part 202: Application protocol: Associative draughting.*

NOTE - ISO 10303-202 is referenced normatively solely for the definition of the term AIC.



 

# 3 Terms, definitions, and abbreviations

## 3.1 Terms defined in ISO 10303–1

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-1 apply:

— abstract test suite (ATS);

— application;

— application context;

— application protocol (AP);

— data;

— data exchange;

— generic resource;

— implementation method;

— information;

— integrated resource;

— interpretation;

— model;

— product;

— product data;

— structure.

## 3.2 Terms defined in ISO 10303–42

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-42 apply:

— boundary;

— coordinate space;

— curve;

— dimensionality;

— domain;

— parameter space;

— self-intersect;

— surface.

## 3.3 Terms defined in ISO 10303–202

For the purposes of this part of ISO 10303, the following terms defined in ISO 10303-202 apply:

**3.3.1**
**application interpreted construct (AIC)**
a logical grouping of interpreted constructs that supports a specific function for the usage of product data across multiple application contexts.

## 3.4 Other terms and definitions

For the purposes of this part of ISO 10303, the following terms and definitions apply:

**3.4.1**
**geometrically bounded**
a description for a geometric shape that uses only values in coordinate space to define its boundaries and connectivity, and no topological constructs.

## 3.5 Abbreviations

For the purposes of this part of ISO 10303, the following abbreviations apply:

| | |
|---|---|
| AIC | application interpreted construct |
| AP | application protocol |
| ATS | abstract test suite |

# 4 EXPRESS short listing

This clause specifies the EXPRESS schema that uses elements from the integrated resources and contains the types, entity specializations, and functions that are specific to this part of ISO 10303.

NOTE 1 - There may be subtypes and items of select lists that appear in the integrated resources that are not imported into the AIC. Constructs are eliminated from the subtype tree or select list through the use of the implicit interface rules of ISO 10303-11. References to eliminated constructs are outside the scope of the AIC. In some cases, all items of the select list are eliminated. Because AICs are intended to be implemented in the context of an application protocol, the items of the select list will be defined by the scope of the application protocol.

This application interpreted construct provides a consistent set of geometric entities for the definition of surface models that consist of points, elementary or sculptured curves, and elementary or sculptured surfaces. Geometry shall be bounded; no topological entities are used for bounding.

EXPRESS specification:

```
*)
SCHEMA aic_geometrically_bounded_surface;

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

  USE FROM geometric_model_schema (           -- ISO 10303-42
      geometric_set);

  USE FROM geometry_schema (                  -- ISO 10303-42
      axis1_placement,
      axis2_placement_2d,
      axis2_placement_3d,
      b_spline_curve,
      b_spline_curve_with_knots,
      b_spline_surface,
      b_spline_surface_with_knots,
      bezier_curve,
      bezier_surface,
      boundary_curve,
      bounded_pcurve,
      bounded_surface_curve,
      cartesian_point,
      cartesian_transformation_operator_3d,
      circle,
      composite_curve,
      composite_curve_on_surface,
      composite_curve_segment,
      conical_surface,
      curve,
      curve_bounded_surface,
      curve_replica,
      cylindrical_surface,
      degenerate_pcurve,
      degenerate_toroidal_surface,
      direction,
      ellipse,
      evaluated_degenerate_pcurve,
```

```
      geometric_representation_context,
      hyperbola,
      intersection_curve,
      line,
      offset_curve_3d,
      offset_surface,
      outer_boundary_curve,
      parabola,
      pcurve,
      plane,
      point,
      point_on_curve,
      point_on_surface,
      polyline,
      quasi_uniform_curve,
      quasi_uniform_surface,
      rational_b_spline_curve,
      rational_b_spline_surface,
      rectangular_composite_surface,
      rectangular_trimmed_surface,
      reparametrised_composite_curve_segment,
      seam_curve,
      spherical_surface,
      surface,
      surface_curve,
      surface_of_linear_extrusion,
      surface_of_revolution,
      surface_patch,
      surface_replica,
      swept_surface,
      toroidal_surface,
      trimmed_curve,
      uniform_curve,
      uniform_surface,
      vector);

  USE FROM product_property_representation_schema (
      shape_representation);                    -- ISO 10303-41

  USE FROM representation_schema (              -- ISO 10303-43
      definitional_representation,
      mapped_item,
      parametric_representation_context,
      representation,
      representation_item,
      representation_map);
(*
```



 

NOTE 2 - The schemas referenced above can be found in the following parts of ISO 10303:

| | |
|---|---|
| geometric_model_schema | ISO 10303-42 |
| geometry_schema | ISO 10303-42 |
| product_property_representation_schema | ISO 10303-41 |
| representation_schema | ISO 10303-43 |
| support_resource_schema | ISO 10303-41 |

## 4.1 Fundamental concepts and assumptions

The following entities are intended to be independently instantiated in the application protocol schemas that use this AIC:

– geometrically_bounded_surface_shape_representation.

## 4.2 aic_geometrically_bounded_surface schema entity definition: geometrically_bounded_surface_shape_representation

A **geometrically_bounded_surface_shape_representation** describes the shape or portions of the shape of a **product** using a surface model without topology.

NOTE 1 - Entity **product** is not included in this part of ISO 10303.

The highest level entity of this part of ISO 10303 is **geometrically_bounded_surface_shape_representation**. It is a **shape_representation** as defined in ISO 10303-41 consisting of **geometric_set**s. **Point**s, **curve**s, and **surface**s may be contained in a **geometric_set** provided they are of the same dimensionality as defined in ISO 10303-42, rule **compatible_dimension**. A **geometrically_bounded_surface_shape_representation** shall include at least one **surface**. The geometric entities that are exclusively used to support the definition of other geometric entities shall not themselves exist in the sets of **elements** of a **geometric_set**. All geometric entities shall be of dimensionality three except for two-dimensional geometry that is used for the purpose of defining **pcurve**s. The use of one-dimensional **cartesian_point**s is excluded.

All unbounded curves and surfaces shall be explicitly trimmed. The boundaries of curves shall be defined either by points on curves and explicit associations between these points and curves, or by parameter values. The boundaries of surfaces shall be defined either by curves on surfaces and explicit associations between these curves and surfaces, or by parameter values.

The **items** of a **geometrically_bounded_surface_shape_representation** may also be of type **mapped_item**, which is defined in ISO 10303-43, or **axis2_placement_3d**. These are used to assemble one or several **geometrically_bounded_surface_shape_representation**s into one other **geometrically_bounded_surface_shape_representation**.

The WHERE-rules of this entity restrict the use of the entity data types that are imported from ISO 10303-42 and ISO 10303-43 according to the statements above. Most of these validations of entity type and constraints are specified in the following three functions:

– **gbsf_check_point**;

– **gbsf_check_curve**;

– **gbsf_check_surface**.

The three functions shall be applied to all **elements** of all **geometric_set**s in a **geometrically_bounded_surface_shape_representation**. The functions automatically assess all **point**s, **curve**s, and **surface**s that are referenced by these **elements**. For this the functions are called recursively.

EXAMPLE A **pcurve** references both a **curve** and a **surface**. Function **gbsf_check_curve** validates not only the **pcurve**, but also its underlying geometry. It will, therefore, not only call itself, but also **gbsf_check_surface**.

NOTE 2 - An application protocol that uses this part of ISO 10303 should explicitly permit that the **shape_-representation** entity may be instantiated as a **geometrically_bounded_surface_shape_representation**.

<u>EXPRESS specification:</u>

```
*)

ENTITY geometrically_bounded_surface_shape_representation
   SUBTYPE OF (shape_representation);
WHERE
  WR1 : SIZEOF (QUERY (it <* SELF.items |
        NOT (SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET',
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.MAPPED_ITEM',
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.AXIS2_PLACEMENT_3D'] * TYPEOF
        (it)) = 1))) = 0;
  WR2 : SIZEOF (QUERY (it <* SELF.items |
        SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET',
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.MAPPED_ITEM'] * TYPEOF
        (it)) = 1)) > 0;
  WR3 : SIZEOF (QUERY (mi <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.MAPPED_ITEM' IN TYPEOF (it)) |
        NOT (('AIC_GEOMETRICALLY_BOUNDED_SURFACE.' +
        'GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION'
        IN TYPEOF (mi\mapped_item.mapping_source.mapped_representation))
        AND
        (SIZEOF(QUERY (mr_it <*
        mi\mapped_item.mapping_source.mapped_representation.items |
        ('AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET'
        IN TYPEOF (mr_it)))) > 0 )))) = 0;
  WR4 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (pnt <* QUERY (gsel <*
        gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.POINT' IN TYPEOF (gsel)) |
        NOT (gbsf_check_point(pnt)))) = 0))) = 0;
```



 

```
  WR5 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (cv <* QUERY (gsel <*
        gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CURVE' IN TYPEOF (gsel)) |
        NOT (gbsf_check_curve(cv)))) = 0))) = 0;
  WR6 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        NOT (SIZEOF (QUERY (sf <* QUERY (gsel <*
        gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE' IN TYPEOF (gsel)) |
        NOT (gbsf_check_surface(sf)))) = 0))) = 0;
  WR7 : SIZEOF (QUERY (gs <* QUERY (it <* SELF.items |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.GEOMETRIC_SET' IN TYPEOF (it)) |
        SIZEOF (QUERY (gsel <* gs\geometric_set.elements |
        'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE'
        IN TYPEOF (gsel))) > 0)) > 0;

END_ENTITY;
(*
```

Formal propositions:

**WR1:** The **items** in a **geometrically_bounded_surface_shape_representation** shall be **geometric_sets**, **mapped_items**, or **axis2_placement_3ds**.

NOTE 3 - **Axis2_placement_3d** is a valid **mapped_item.mapping_target**. To include another **representation** into the list of **items** of a **geometrically_bounded_surface_shape_representation** (see WR3 for valid **mapped_items**), the **mapped_item.mapping_source.mapping_origin** may be any entity that is geometrically founded in the **geometric_representation_context** of the **mapped_representation**. If this entity is an **axis2_placement_3d**, the operator that maps the **mapped_representation** into the **geometrically_bounded_surface_shape_representation** corresponds to a transformation matrix with only translation and rotation enabled. If a **cartesian_transformation_operator_3d** is used as **mapping_origin**, scaling and mirroring are possible.

**WR2:** At least one of the **items** in a **geometrically_bounded_surface_shape_representation** shall be either a **geometric_set** or a **mapped_item**.

**WR3:** If there is a **mapped_item** in a **geometrically_bounded_surface_shape_representation**, the **mapped_representation** of its **mapping_source** shall be a **geometrically_bounded_surface_shape_representation**. This **shape_representation** shall include at least one **geometric_set**.

**WR4:** Each **point** that is among the **elements** of a **geometric_set** that is one of the **items** of a **geometrically_bounded_surface_shape_representation** shall be a valid **point**.

**WR5:** Each **curve** that is among the **elements** of a **geometric_set** that is one of the **items** of a **geometrically_bounded_surface_shape_representation** shall be a valid **curve**.

**WR6:** Each **surface** that is among the **elements** of a **geometric_set** that is one of the **items** of a **geometrically_bounded_surface_shape_representation** shall be a valid **surface**.

**WR7:** At least one of the **elements** of at least one **geometric_set** that is among the **items** of a **geometrically_bounded_surface_shape_representation** shall be a **surface**.

Informal propositions:

**IP1:** A **b_spline_curve** shall not self-intersect.

**IP2:** A **composite_curve** shall not self-intersect.

**IP3:** An **offset_curve_3d** shall not self-intersect.

**IP4:** A **b_spline_surface** shall not self-intersect.

**IP5:** An **offset_surface** shall not self-intersect.

**IP6:** The geometric entities that are exclusively used to support the definition of other geometric entities shall not themselves exist in the sets of **elements** of a **geometric_set**.

## 4.3 aic_geometrically_bounded_surface function definitions

This section describes functions required to formulate constraints for the aic_geometrically_bounded_surface schema. These functions are used in the specification of the entity **geometrically_bounded_surface_shape_representation**.

### 4.3.1 gbsf_check_point

The **gbsf_check_point** function checks a **point** instance for validity in the context of a **geometrically_bounded_surface_shape_representation**. All geometry that is referenced by this **point** instance, such as **curve**s and **surface**s, are also validated. When **curve**s or **surface**s are referenced, the functions **gbsf_check_curve** respectively **gbsf_check_surface** are called. The recursive process within these functions terminates at entity types that do not reference any **point**s, **curve**s, or **surface**s.

The following point types and their subtypes are within the scope of the **geometrically_bounded_surface_shape_representation** and are valid input to this function:

– **cartesian_point**;

– **degenerate_pcurve**;

– **point_on_curve**;

– **point_on_surface**.

The three latter ones in the list reference either **curve**s or **surface**s or both.



 

The **basis_surface** of a **degenerate_pcurve** may be any of the valid **surface**s in a **geometrically_-bounded_surface_shape_representation**. The **reference_to_curve** of a **degenerate_pcurve** shall be of one of the following types:

– **b_spline_curve**;

– **composite_curve** (recursive);

– **conic**;

– **curve_replica** (recursive);

– **line**;

– **polyline**;

– **trimmed_curve** (recursive).

NOTE 1 - This function applies to those entity types that are marked in the list above a recursive process to check their entity references for valid instantiations.

The **basis_curve** of a **point_on_curve** may be any of the valid **curve**s in a **geometrically_bounded_-surface_shape_representation**.

The **basis_surface** of a **point_on_surface** may be any of the valid **surface**s in a **geometrically_-bounded_surface_shape_representation**.

This function returns TRUE, if the types of all referenced geometries are within the scope of the **geometrically_bounded_surface_shape_representation**, otherwise the function returns FALSE.

NOTE 2 - This function does not check the correctness of references with respect to ISO 10303-42. Only additional requirements due to the scope of the **geometrically_bounded_surface_shape_-representation** are checked.

<u>EXPRESS specification</u>:

```
*)
FUNCTION gbsf_check_point (pnt : point) : BOOLEAN;

(* This function varifies the validity of a point in the context of a
   geometrically bounded surface model.
*)
  (* a cartesian_point is valid
  *)
  IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CARTESIAN_POINT'
    IN TYPEOF (pnt) THEN RETURN(TRUE);
  ELSE
```

```
  (* a point_on_curve shall reference a valid curve
  *)
    IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.POINT_ON_CURVE'
      IN TYPEOF (pnt) THEN RETURN
      (gbsf_check_curve(pnt\point_on_curve.basis_curve));
    ELSE

    (* a point_on_surface shall reference a valid surface
    *)
      IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.POINT_ON_SURFACE'
        IN TYPEOF (pnt) THEN RETURN (gbsf_check_surface
        (pnt\point_on_surface.basis_surface));
      ELSE

      (* a degenerate_pcurve shall reference a valid curve and
         a valid surface
      *)
        IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.DEGENERATE_PCURVE'
          IN TYPEOF (pnt) THEN RETURN
          ((gbsf_check_curve
          (pnt\degenerate_pcurve.reference_to_curve\
                                    representation.items[1]))
          AND (gbsf_check_surface (pnt\degenerate_pcurve.basis_surface)));
        END_IF;
      END_IF;
    END_IF;
  END_IF;
  RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

**pnt:** (input) the **point** that is being checked for being a valid **point** in a **geometrically_bounded_surface_shape_representation**.

**BOOLEAN:** (return) is TRUE if the **point** is a valid **point**; else FALSE.

## 4.3.2 gbsf_check_curve

The **gbsf_check_curve** function checks a **curve** instance for validity in the context of a **geometrically_bounded_surface_shape_representation**. One of the requirements for validity is that the resulting geometry of the **curve** instance is bounded. The fulfillment of this requirement is checked by this function by validating all geometry that is referenced by this **curve** instance, such as other **curve**s and **surface**s. Where appropriate an instance is investigated recursively. This means if a **curve** references another **curve** as a basis curve or parent curve, the **gbsf_check_curve** function is called again. If a **surface** is referenced, the **gbsf_check_surface** function is called. The recursive process terminates at entity types that do not reference any **curve**s or **surface**s.



 

Besides the requirement for bounding other criteria for validity apply; a **curve** shall not be defined to be self-intersecting.

EXAMPLE One of the constraints that is validated by this function is whether the self-intersection flag of a **b_spline_curve** instance is set to TRUE, FALSE, or UNKNOWN; only FALSE and UNKNOWN are valid.

The following curve types and their subtypes are within the scope of the **geometrically_bounded_-surface_shape_representation** and are valid input to this function:

– **b_spline_curve**;

– **composite_curve**;

– **conic**;

– **curve_replica**;

– **line**;

– **offset_curve_3d**;

– **pcurve**;

– **polyline**;

– **surface_curve**;

– **trimmed_curve**.

NOTE 1 - This function applies to those entity types that are marked in the lists below a recursive process to check their entity references for valid instantiations.

The **parent_curve** of a **curve_replica** shall both be of one of the following types:

– **b_spline_curve**;

– **composite_curve** (recursive);

– **conic**;

– **curve_replica** (recursive);

– **line**;

– **offset_curve_3d** (recursive);

– **pcurve** (recursive);

– **polyline**;

– **surface_curve** (recursive);

– **trimmed_curve** (recursive).

The **basis_curve** of an **offset_curve_3d** shall be of one of the following types:

– **b_spline_curve**;

– **composite_curve** (recursive);

– **conic**;

– **curve_replica** (recursive);

– **line**;

– **offset_curve_3d** (recursive);

– **pcurve** (recursive);

– **surface_curve** (recursive);

– **trimmed_curve** (recursive).

The one instance in the set of **items** of a **definitional_representation** that is referenced as **reference_-to_curve** by a **pcurve** shall be of one of the following types:

– **b_spline_curve**;

– **composite_curve** (recursive);

– **conic**;

– **curve_replica** (recursive);

– **line**;

– **polyline**;

– **trimmed_curve** (recursive).



 

The **curve_3d** of a **surface_curve** shall be of one of the following types:

- **b_spline_curve**;
- **composite_curve** (recursive);
- **conic**;
- **curve_replica** (recursive);
- **line**;
- **offset_curve_3d** (recursive);
- **polyline**;
- **surface_curve** (recursive);
- **trimmed_curve** (recursive).

**Pcurve** and **surface_curve** reference **surface**s. Function **gbsf_check_surface** is called for validation of these **surface**s. The **geometrically_bounded_surface_shape_representation** requires the same constraints on valid **surface** references for **pcurve**s and **surface_curve**s as specified in ISO 10303-42.

A valid **polyline** shall consist of at least three **cartesian_point**s.

The attribute **self_intersect** shall for B-spline and offset geometry be set to FALSE or UNKNOWN.

This function returns TRUE, if the types of all referenced geometries are within the scope of the **geometrically_bounded_surface_shape_representation** and if all constraints are satisfied, otherwise the function returns FALSE.

NOTE 2 - This function does not check the correctness of references with respect to ISO 10303-42. ISO 10303-42 requires, for example, that the **segments** of a **composite_curve** shall be of type **composite_curve_segment** and that the **parent_curve** of a **composite_curve_segment** shall be a **bounded_curve**. This constraint is not verified by this function. Only additional requirements due to the scope of the **geometrically_bounded_surface_shape_representation** are checked.

EXPRESS specification:

```
*)
FUNCTION gbsf_check_curve (cv : representation_item) : BOOLEAN;

(* This function varifies the validity of a curve in the context of a
   geometrically bounded surface model. Representation_items are
   valid input, however, they are supposed to be curves; otherwise
   this function will return false.
*)

 (* complex subtypes of curve that are both bounded_curve and oneof
    conic, curve_replica, line, or offset_curve_3d are not valid
 *)
 IF SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.BOUNDED_CURVE',
   'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CONIC',
   'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CURVE_REPLICA',
   'AIC_GEOMETRICALLY_BOUNDED_SURFACE.LINE',
   'AIC_GEOMETRICALLY_BOUNDED_SURFACE.OFFSET_CURVE_3D']
   * TYPEOF(cv)) > 1 THEN RETURN(FALSE);
 END_IF;

 (* circle, ellipse, and trimmed_curve are valid curves; they
    are bounded per definition
 *)
 IF SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.CIRCLE',
 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.ELLIPSE',
 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.TRIMMED_CURVE']
 * TYPEOF(cv)) = 1
 THEN RETURN(TRUE);
 ELSE

   (* b_spline_curves shall not self-intersect
   *)
   IF (('AIC_GEOMETRICALLY_BOUNDED_SURFACE.B_SPLINE_CURVE'
     IN TYPEOF(cv)) AND
     (cv\b_spline_curve.self_intersect = FALSE) OR
     (cv\b_spline_curve.self_intersect = UNKNOWN))
   THEN RETURN(TRUE);
   ELSE

     (* a composite_curve shall not self-intersect and all of
        its segments shall reference valid curves
     *)
     IF (('AIC_GEOMETRICALLY_BOUNDED_SURFACE.COMPOSITE_CURVE'
     IN TYPEOF(cv)) AND
       (cv\composite_curve.self_intersect = FALSE) OR
       (cv\composite_curve.self_intersect = UNKNOWN))
     THEN
       RETURN (SIZEOF (QUERY (seg <* cv\composite_curve.segments |
```



 

```
  NOT (gbsf_check_curve(seg.parent_curve)))) = 0);
ELSE

  (* a curve_replica shall reference a valid curve
  *)
  IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CURVE_REPLICA'
    IN TYPEOF(cv) THEN
    RETURN (gbsf_check_curve (cv\curve_replica.parent_curve));
  ELSE

    (* an offset_curve_3d shall not self-intersect and shall
       reference a valid curve; a polyline is not a valid
       basis_curve
    *)
    IF (('AIC_GEOMETRICALLY_BOUNDED_SURFACE.OFFSET_CURVE_3D'
      IN TYPEOF(cv))
      AND
      ((cv\offset_curve_3d.self_intersect = FALSE) OR
      (cv\offset_curve_3d.self_intersect = UNKNOWN))
      AND
      (NOT ('AIC_GEOMETRICALLY_BOUNDED_SURFACE.POLYLINE'
      IN TYPEOF(cv\offset_curve_3d.basis_curve)))) THEN RETURN
      (gbsf_check_curve (cv\offset_curve_3d.basis_curve));
    ELSE

      (* a pcurve shall reference a valid curve and a valid
         basis_surface
      *)
      IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.PCURVE'
        IN TYPEOF(cv) THEN RETURN ((gbsf_check_curve
        (cv\pcurve.reference_to_curve\representation.items[1]))
        AND
        (gbsf_check_surface (cv\pcurve.basis_surface)));
      ELSE

        (* a polyline shall have at least 3 points
        *)
        IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.POLYLINE'
          IN TYPEOF(cv) THEN
          IF (SIZEOF (cv\polyline.points) >= 3)
            THEN RETURN (TRUE);
          END_IF;
        ELSE

          (* a surface_curve references a curve_3d and one or two
             pcurves or one or two surfaces or one of each;
             all of these references shall be valid
          *)
          IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE_CURVE'
            IN TYPEOF(cv) THEN

            (* if the curve reference is correct, check also the rest
```

```
                        *)
                        IF gbsf_check_curve (cv\surface_curve.curve_3d) THEN
                          REPEAT i := 1 TO SIZEOF
                          (cv\surface_curve.associated_geometry);

                          (* do for one or two associated_geometrys:
                          *)
                          IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE' IN TYPEOF
                            (cv\surface_curve.associated_geometry[i]) THEN
                            IF NOT gbsf_check_surface
                              (cv\surface_curve.associated_geometry[i]) THEN
                              RETURN(FALSE);
                            END_IF;
                          ELSE
                            IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.PCURVE' IN TYPEOF
                              (cv\surface_curve.associated_geometry[i]) THEN
                              IF NOT gbsf_check_curve
                                (cv\surface_curve.associated_geometry[i]) THEN
                                RETURN(FALSE);
                              END_IF;
                            END_IF;
                          END_IF;
                          END_REPEAT;
                          RETURN(TRUE);
                        END_IF;
                      END_IF;
                    END_IF;
                  END_IF;
                END_IF;
              END_IF;
            END_IF;
          END_IF;
  END_IF;
  (* FALSE is returned if the input parameter cv is not a valid curve.
   *)
  RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

**cv:** (input) a **representation_item** that is supposed to be a **curve** that is checked for being valid in the context of a **geometrically_bounded_surface_shape_representation**.

**BOOLEAN:** (output) is TRUE if the **representation_item** is a valid **curve**; else FALSE.

### 4.3.3 gbsf_check_surface

The **gbsf_check_surface** function checks a **surface** instance for validity in the context of a **geometrically_bounded_surface_shape_representation**. One of the requirements for validity is that the resulting geometry of the **surface** instance is bounded. The fulfillment of this requirement is checked by this function by validating all geometry that is referenced by this **surface** instance, such as other **curve**s and **surface**s. Where appropriate an instance is investigated recursively. This means if a **surface** references another **surface** as a basis surface or parent surface, the **gbsf_check_surface** function is called again. If a **curve** is referenced, the **gbsf_check_curve** function is called. The recursive process terminates at entity types that do not reference any **curve**s or **surface**s.

Besides the requirement for bounding other criteria for validity apply; a **surface** shall not be defined to be self-intersecting.

EXAMPLE An example of a constraint that is validated by this function is whether the attribute for self-intersection of an instance of type **b_spline_surface**, is set to TRUE, FALSE, or UNKNOWN; only FALSE and UNKNOWN are valid.

The following surface types and their subtypes are within the scope of the **geometrically_bounded_surface_shape_representation** and are valid input to this function:

– **b_spline_surface**;

– **curve_bounded_surface**;

– **elementary_surface**;

– **offset_surface**;

– **rectangular_composite_surface**;

– **rectangular_trimmed_surface**;

– **surface_replica**;

– **surface_of_revolution**.

Both the **basis_surface** of an **offset_-surface** and the **parent_surface** of a **surface_replica** shall all be one of the **surface** types listed above.

A **surface_of_revolution** references a **curve**. Function **gbsf_check_curve** is called for validation of the **curve**. The **geometrically_bounded_surface_shape_representation** requires the same constraints on valid sweeping curves as specified in ISO 10303-42. All **curve**s that are in the scope of a **geometrically_bounded_surface_shape_representation** are valid **swept_curve**s.

The attribute **self_intersect** shall for B-spline and offset geometry be set to FALSE or UNKNOWN.

This function returns TRUE, if the types of all referenced geometries are within the scope of the **geometrically_bounded_surface_shape_representation** and if all constraints are satisfied, otherwise the function returns FALSE.

NOTE 1 - This function does not check the correctness of references with respect to ISO 10303-42. ISO 10303-42 requires, for example, that the **segments** of a **rectangular_composite_surface** shall be **surface_-patch**es and that these shall have **bounded_surface**s as their parent surfaces. This constraint is not verified by this function. Only additional requirements due to the scope of the **geometrically_bounded_surface_-shape_representation** are checked.

NOTE 2 - This function applies a recursive process to check the entity references of an input **surface** for valid instantiations.

EXPRESS specification:

```
*)

FUNCTION gbsf_check_surface (sf : surface) : BOOLEAN;

(* This function verifies the validity of a surface in the context of a
   geometrically bounded surface model.
*)
  (* a b_spline_surface shall not self-intersect
  *)
  IF (('AIC_GEOMETRICALLY_BOUNDED_SURFACE.B_SPLINE_SURFACE'
    IN TYPEOF(sf)) AND
    (sf\b_spline_surface.self_intersect = FALSE) OR
    (sf\b_spline_surface.self_intersect = UNKNOWN))
  THEN
    RETURN(TRUE);
  ELSE

    (* spherical_surface and toroidal_surface are valid;
       also curve_bounded_surface and rectangular_trimmed_surface
       are per definition bounded and, thus, valid
    *)
    IF SIZEOF (['AIC_GEOMETRICALLY_BOUNDED_SURFACE.SPHERICAL_SURFACE',
      'AIC_GEOMETRICALLY_BOUNDED_SURFACE.TOROIDAL_SURFACE',
      'AIC_GEOMETRICALLY_BOUNDED_SURFACE.CURVE_BOUNDED_SURFACE',
      'AIC_GEOMETRICALLY_BOUNDED_SURFACE.RECTANGULAR_TRIMMED_SURFACE']
      * TYPEOF(sf)) = 1 THEN RETURN(TRUE);
    ELSE

      (* an offset_surface shall not self-intersect and shall
         reference a valid surface
      *)
      IF (('AIC_GEOMETRICALLY_BOUNDED_SURFACE.OFFSET_SURFACE'
        IN TYPEOF(sf)) AND
        (sf\offset_surface.self_intersect = FALSE) OR
        (sf\offset_surface.self_intersect = UNKNOWN))
      THEN
        RETURN (gbsf_check_surface (sf\offset_surface.basis_surface));
      ELSE
```



 

```
          (* a rectangular_composite_surface shall reference a matrix of
             valid surfaces
          *)
          IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.RECTANGULAR_COMPOSITE_SURFACE'
            IN TYPEOF(sf)
          THEN
            REPEAT i := 1 TO SIZEOF
              (sf\rectangular_composite_surface.segments);
              REPEAT j := 1 TO SIZEOF
                (sf\rectangular_composite_surface.segments[i]);
                IF NOT (gbsf_check_surface
                  (sf\rectangular_composite_surface.segments[i][j].
                  parent_surface))
                  THEN RETURN (FALSE);
                END_IF;
              END_REPEAT;
            END_REPEAT;
            RETURN(TRUE);
          ELSE

            (* a surface_replica shall have a valid parent surface
            *)
            IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE_REPLICA'
              IN TYPEOF (sf) THEN RETURN (gbsf_check_surface
              (sf\surface_replica.parent_surface));
            ELSE

              (* a surface_of_revolution shall have a valid sweeping
                 curve; surface_of_linear_extrusion is not valid, as
                 it is unbounded in the direction of the extrusion;
                 it may, however, be valid if trimmed
              *)
              IF 'AIC_GEOMETRICALLY_BOUNDED_SURFACE.SURFACE_OF_REVOLUTION'
                IN TYPEOF(sf) THEN RETURN
                (gbsf_check_curve (sf\swept_surface.swept_curve));
              END_IF;
            END_IF;
          END_IF;
        END_IF;
      END_IF;
    END_IF;
    RETURN(FALSE);
END_FUNCTION;
(*
```

Argument definitions:

**sf:** (input) the **surface** that is being checked for a valid **surface** in a **geometrically_bounded_surface_-shape_representation**.

**BOOLEAN:** (return) is TRUE if the **surface** is a valid **surface**; else FALSE.

```
*)
END_SCHEMA; -- aic_geometrically_bounded_surface
(*
```



 

# 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 the short names are found in the implementation methods included in ISO 10303.

### Table A.1 – Short names of entities

| Entity name | Short name |
|---|---|
| GEOMETRICALLY_BOUNDED_SURFACE_SHAPE_REPRESENTATION | GBSSR |



 

# 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(507) 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

To provide for unambiguous identification of the aic-geometrically-bounded-surface in an open information system, the object identifier

{ iso standard 10303 part(507) version(1) object(1) aic-geometrically-bounded-surface(1) }

is assigned to the **aic_geometrically_bounded_surface** schema (see clause 4). The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.



 

# Annex C
(informative)

## EXPRESS-G diagrams

Figures C.1 through C.18 correspond to the EXPRESS generated from the short listing given in clause 4 using the interface specifications of ISO 10303-11. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex D of ISO 10303-11.

NOTE - The following select types are interfaced into the AIC expanded listing according to the implicit interface rules of ISO 10303-11. These select types are not referenced by other entities in this part of ISO 10303.

— curve_on_surface;

— founded_item_select;

— measure_value;

— transformation;

— vector_or_direction.

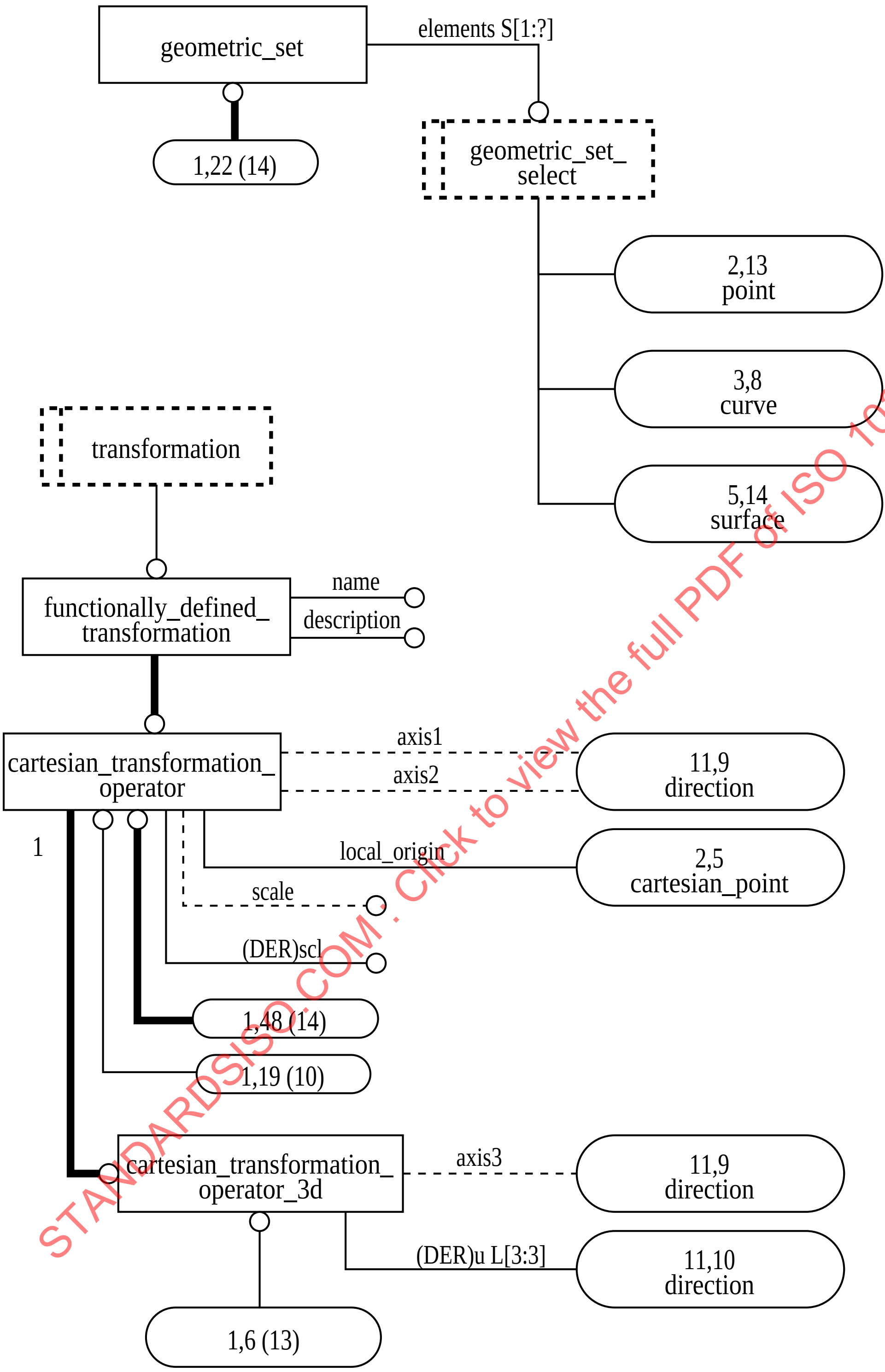




**Figure C.1 – EXPRESS-G diagram 1 of 18**

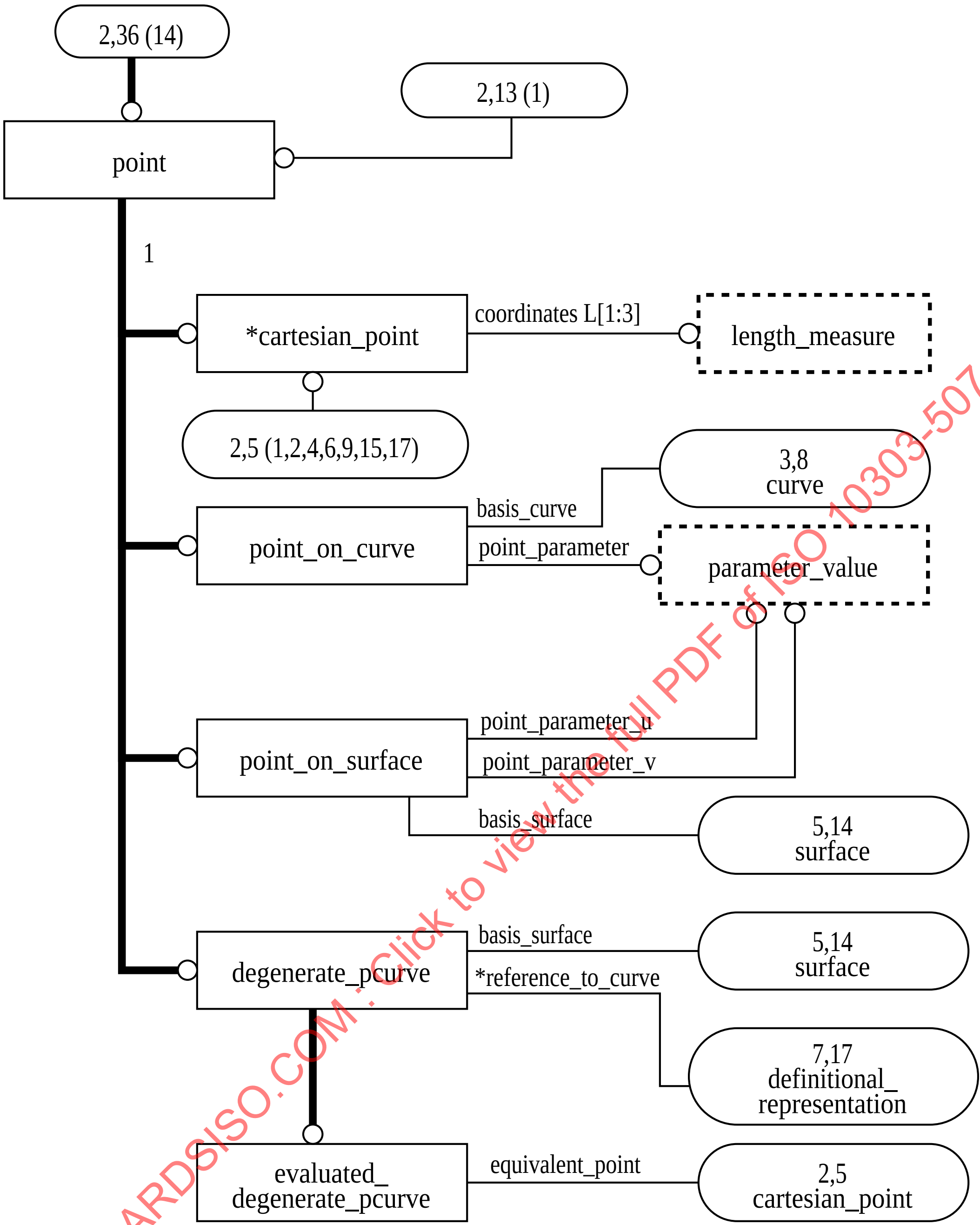




**Figure C.2 – EXPRESS-G diagram 2 of 18**

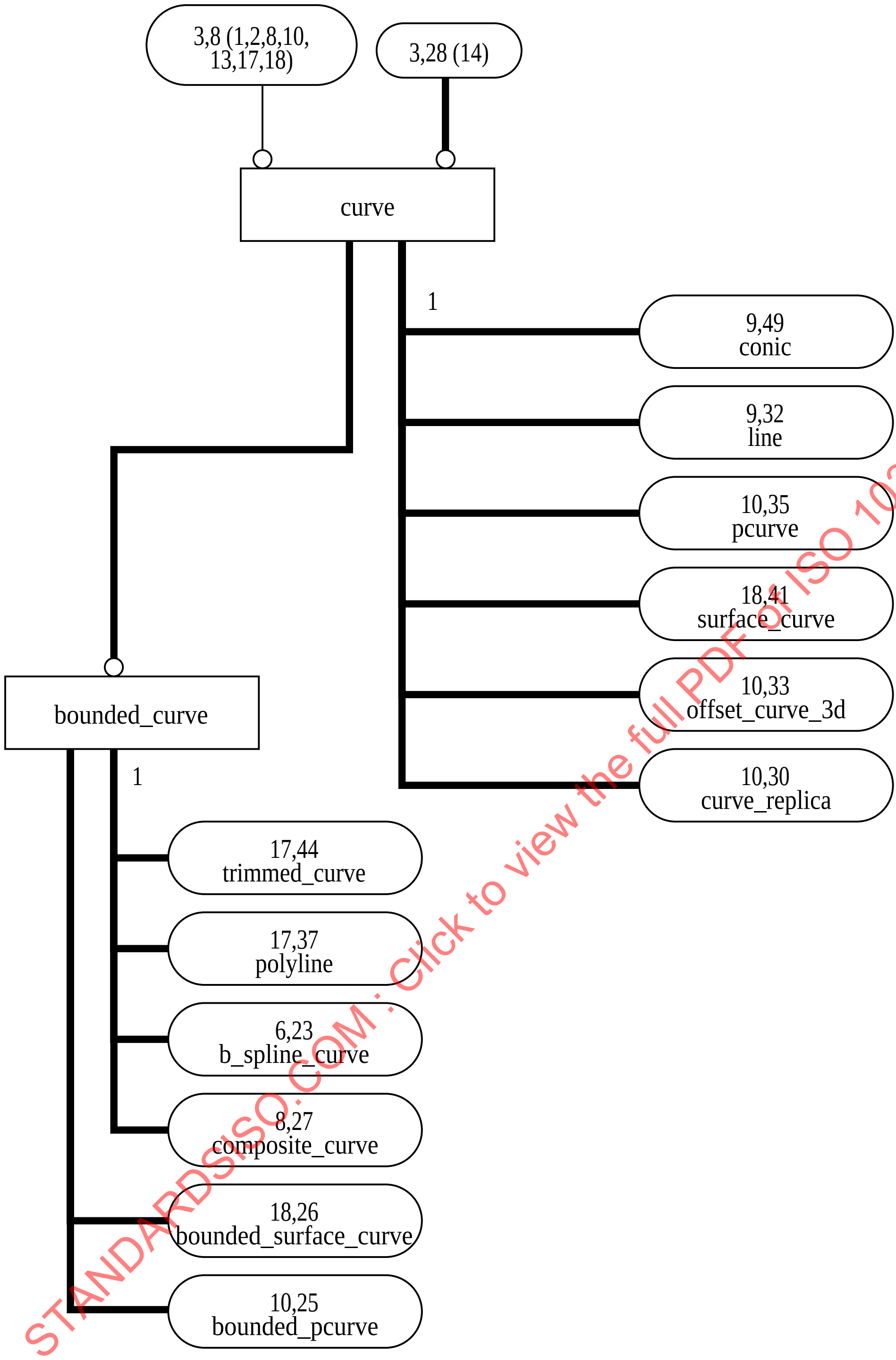




**Figure C.3 – EXPRESS-G diagram 3 of 18**

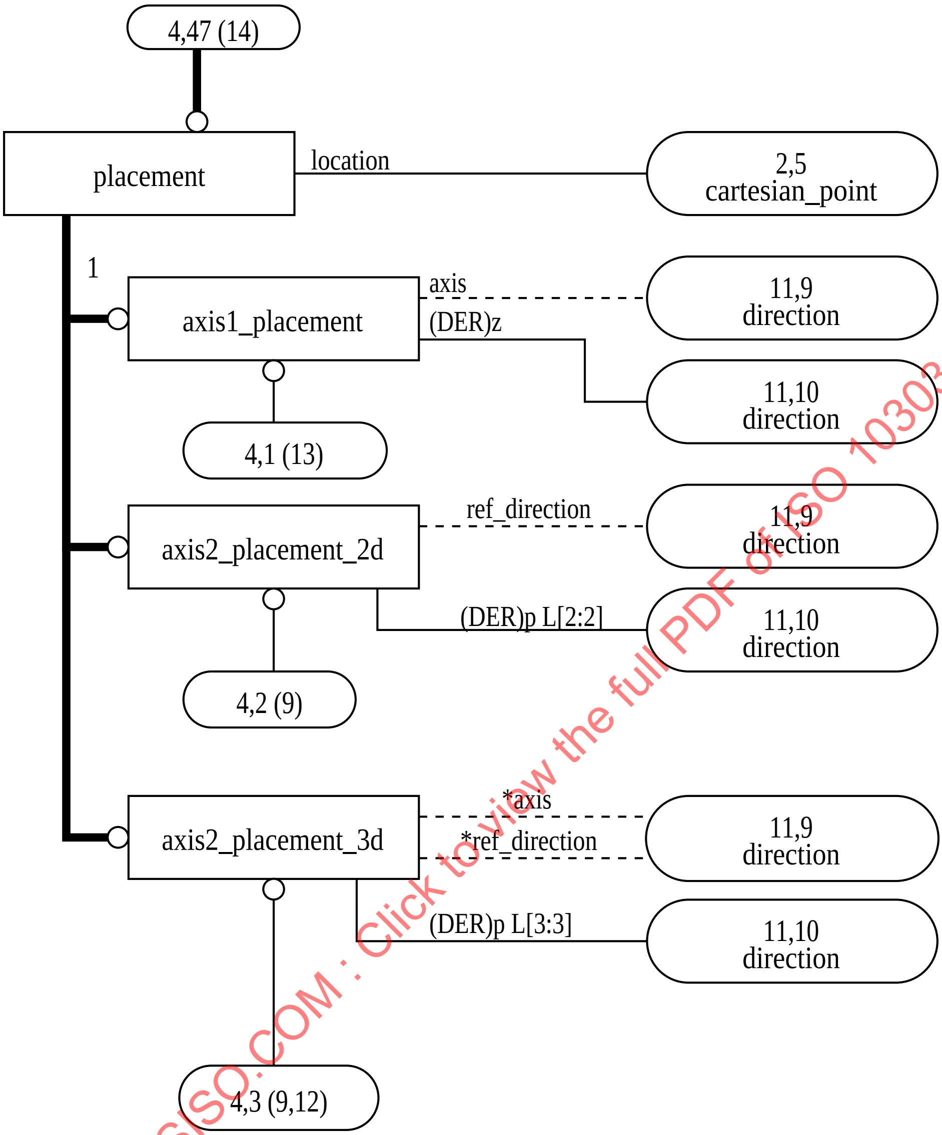




**Figure C.4 – EXPRESS-G diagram 4 of 18**

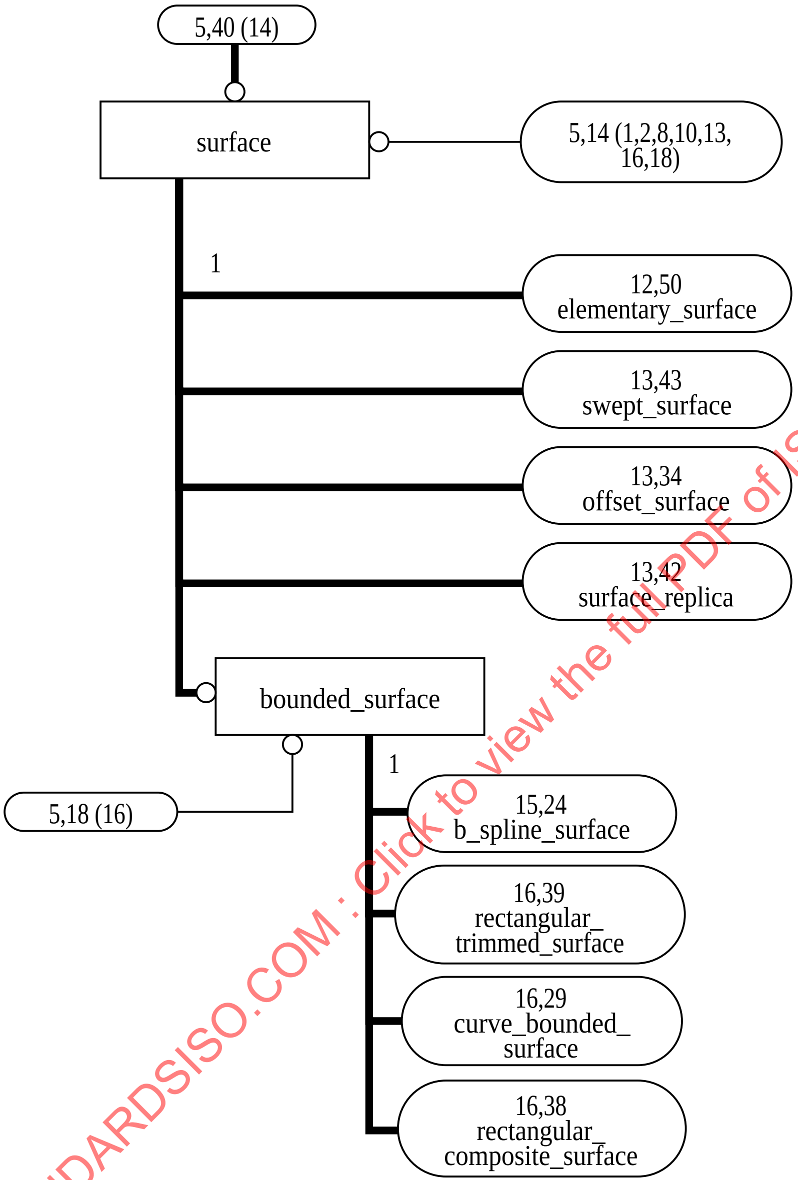




**Figure C.5 – EXPRESS-G diagram 5 of 18**

 

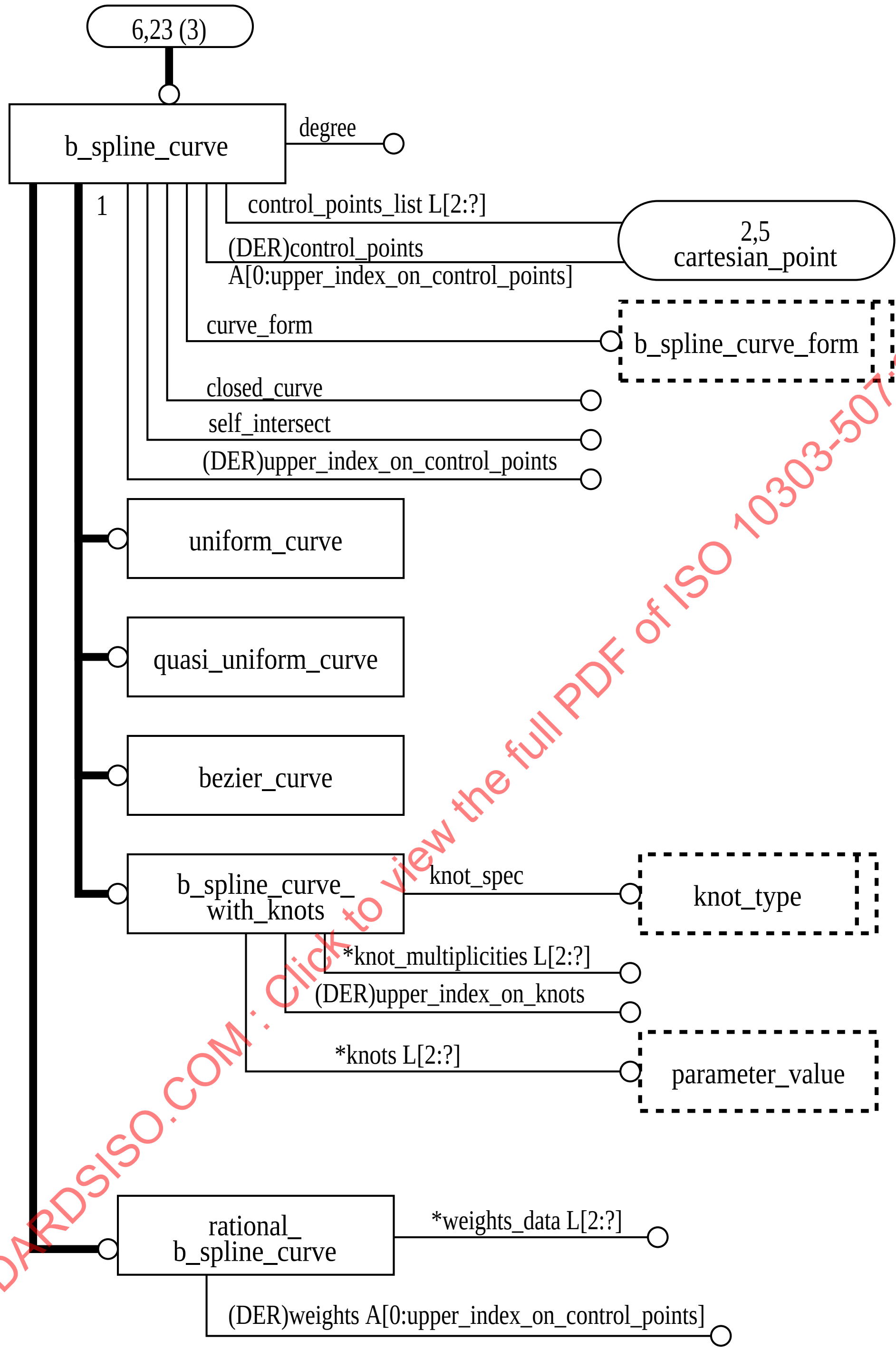




**Figure C.6 – EXPRESS-G diagram 6 of 18**

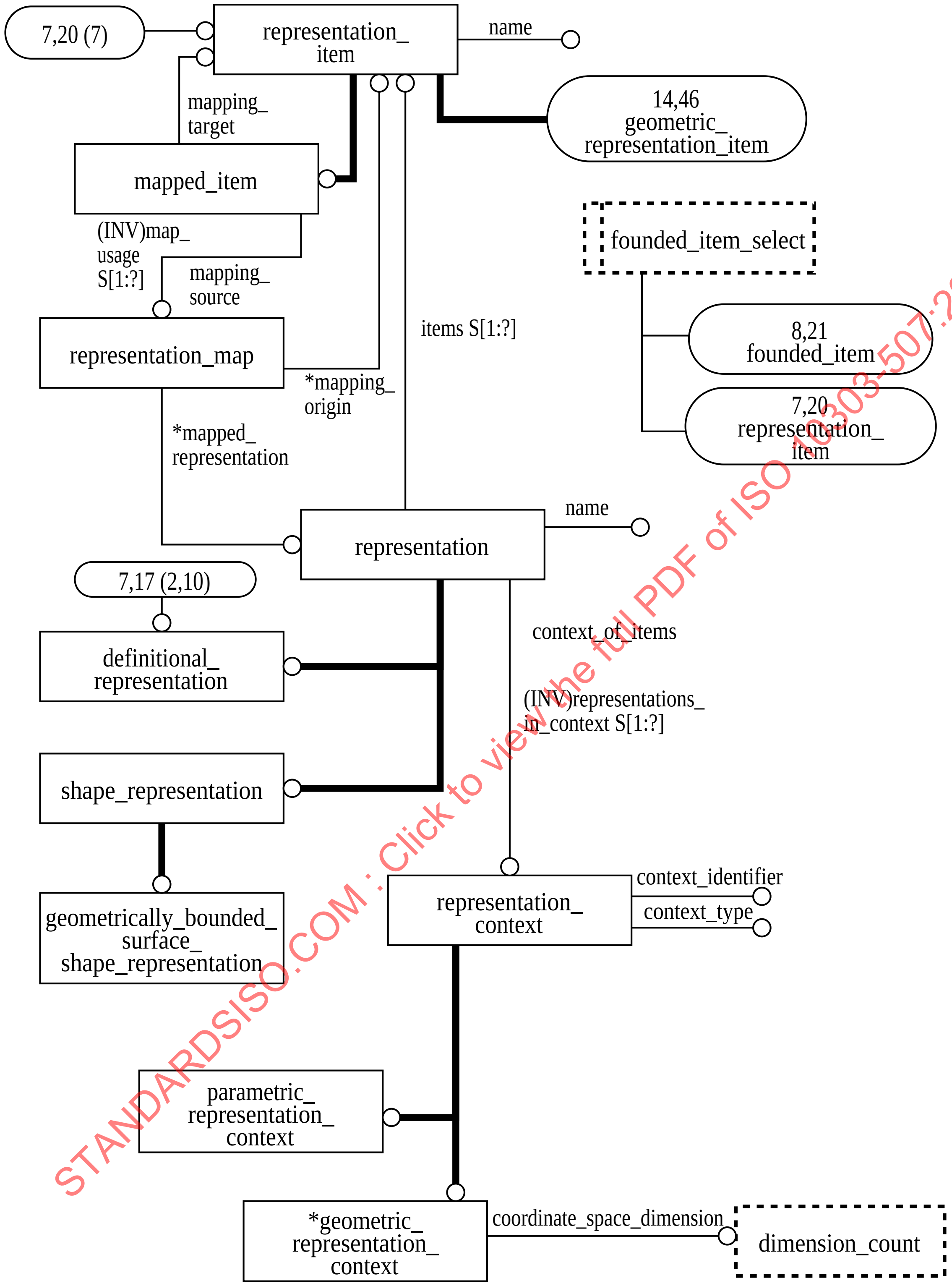




**Figure C.7 – EXPRESS-G diagram 7 of 18**

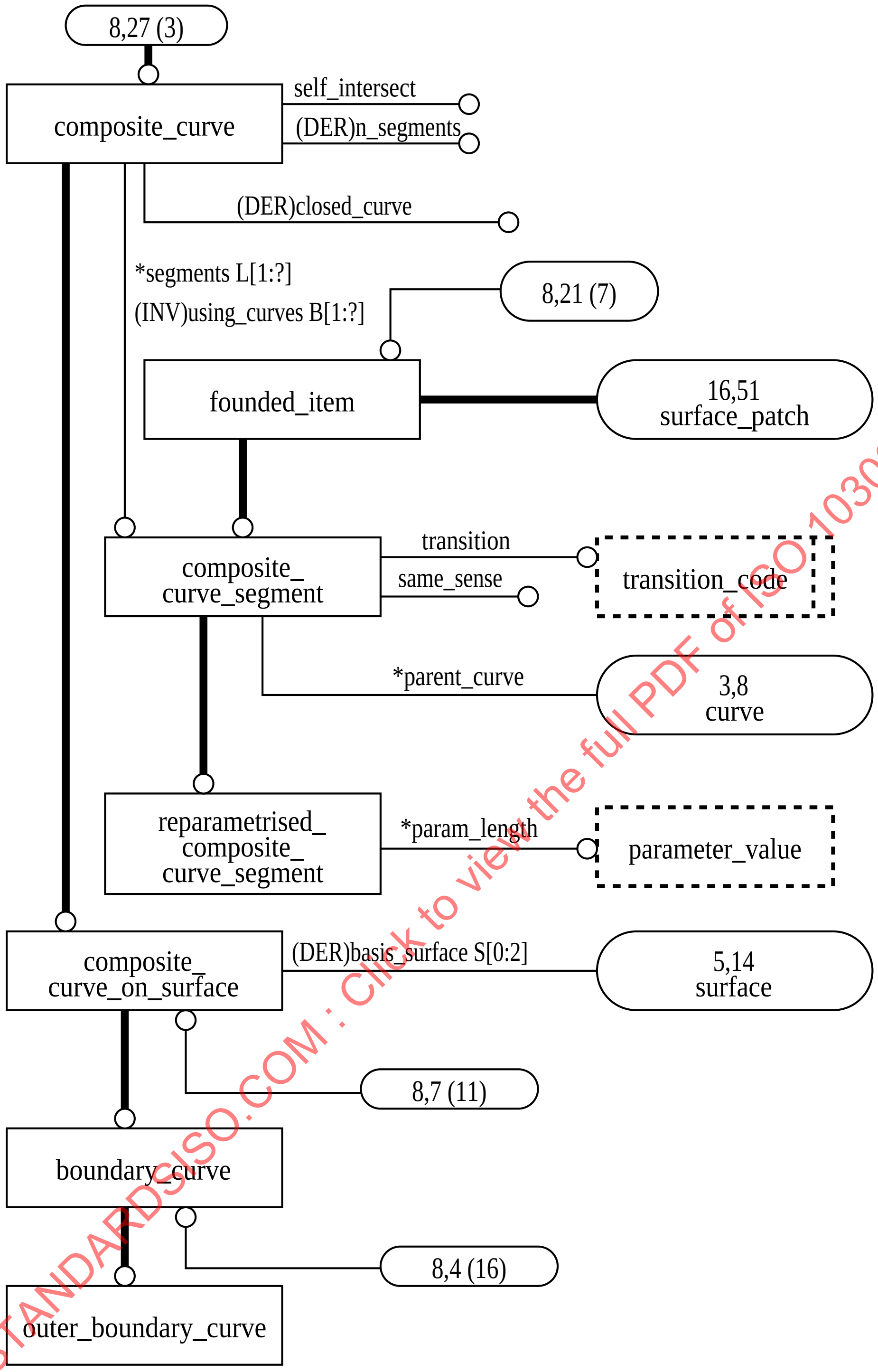




**Figure C.8 – EXPRESS-G diagram 8 of 18**

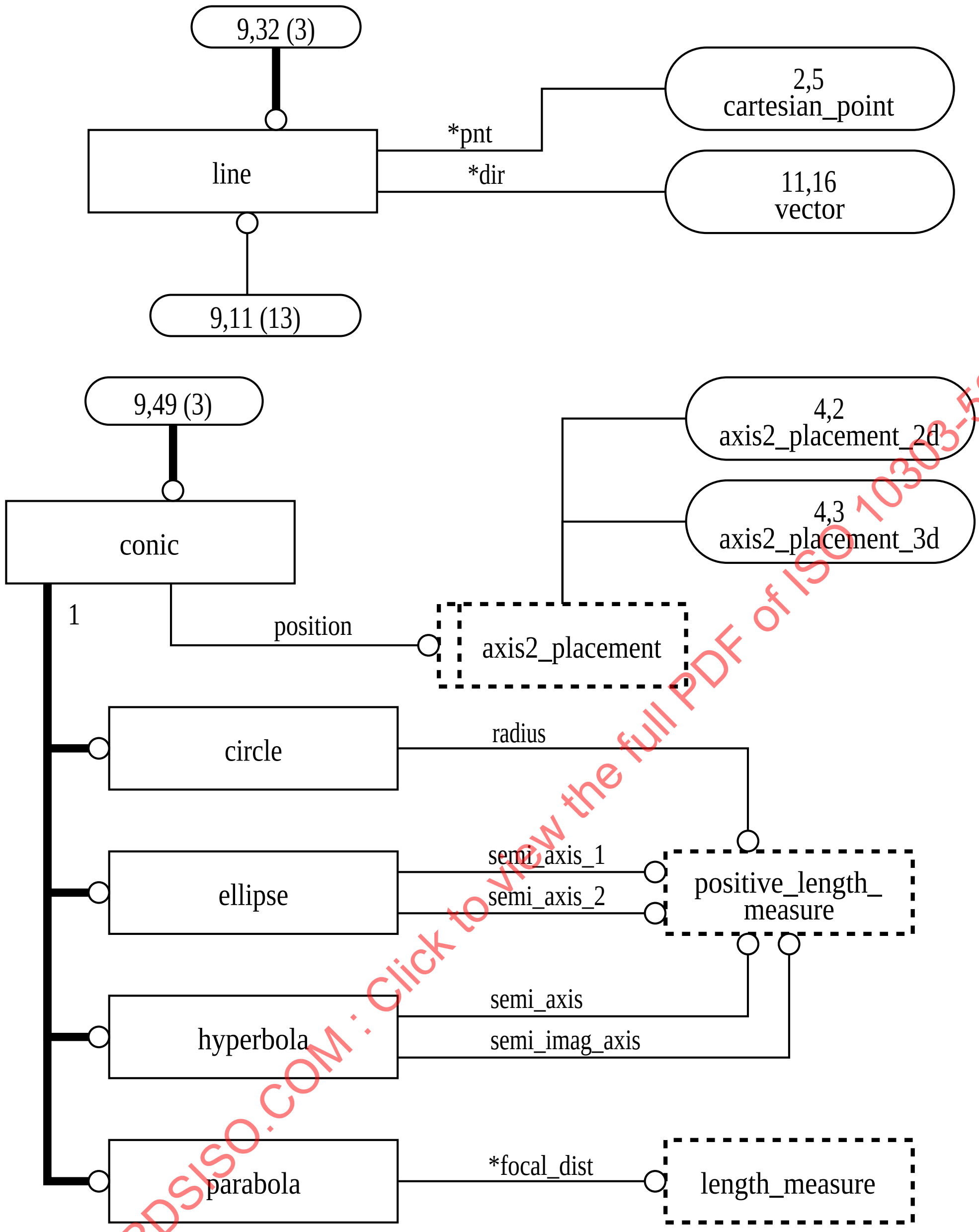




**Figure C.9 – EXPRESS-G diagram 9 of 18**

 

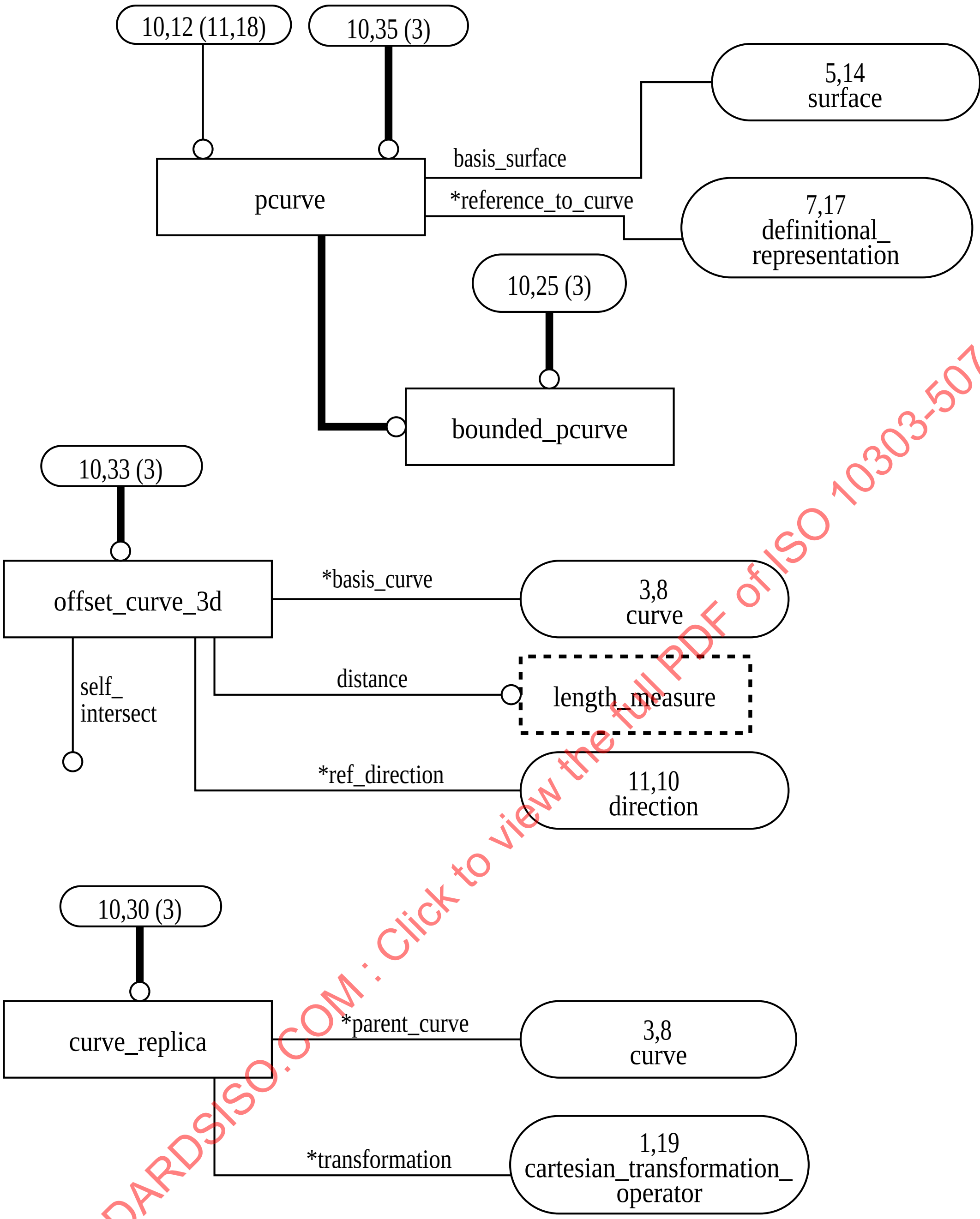




**Figure C.10 – EXPRESS-G diagram 10 of 18**

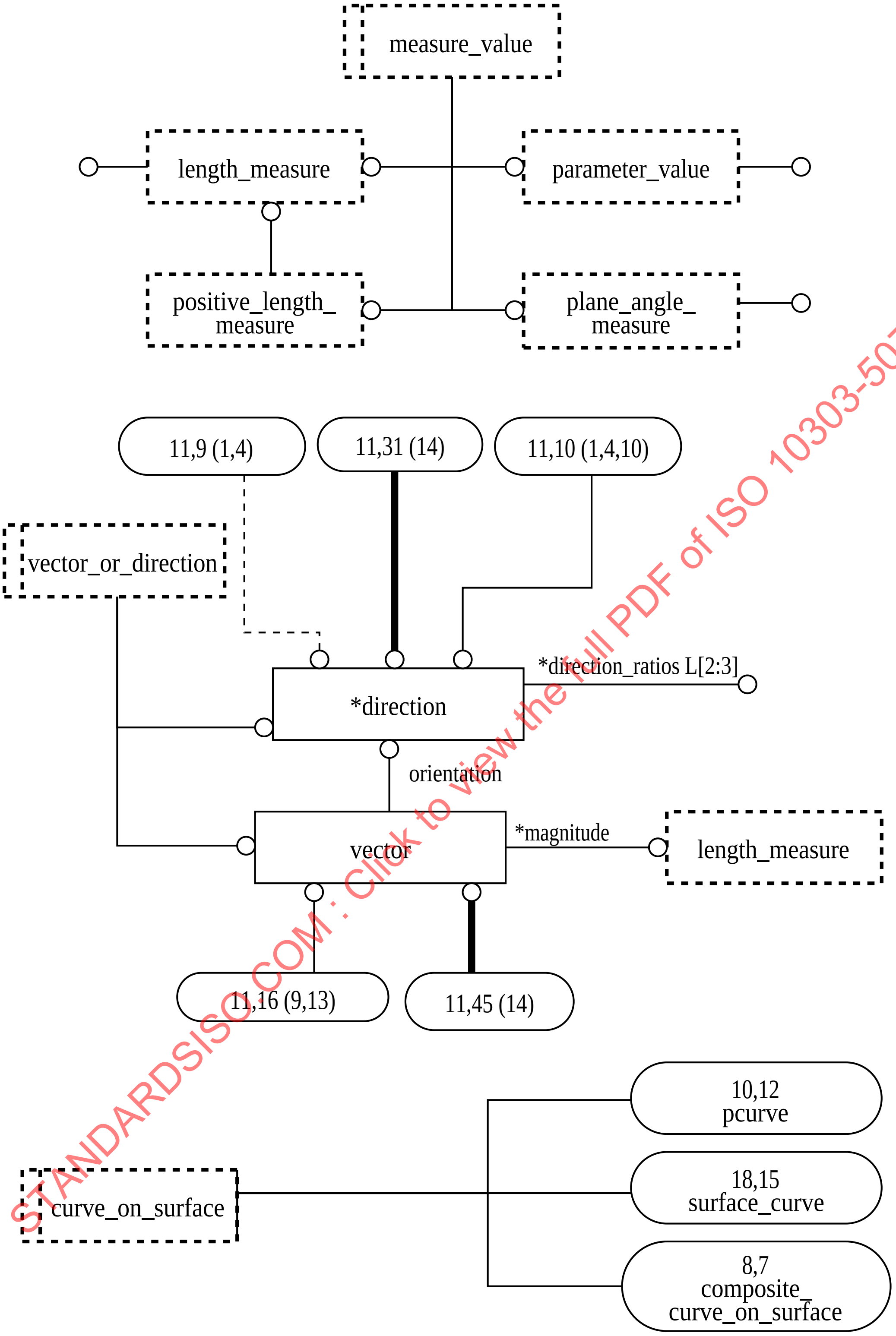




**Figure C.11 – EXPRESS-G diagram 11 of 18**

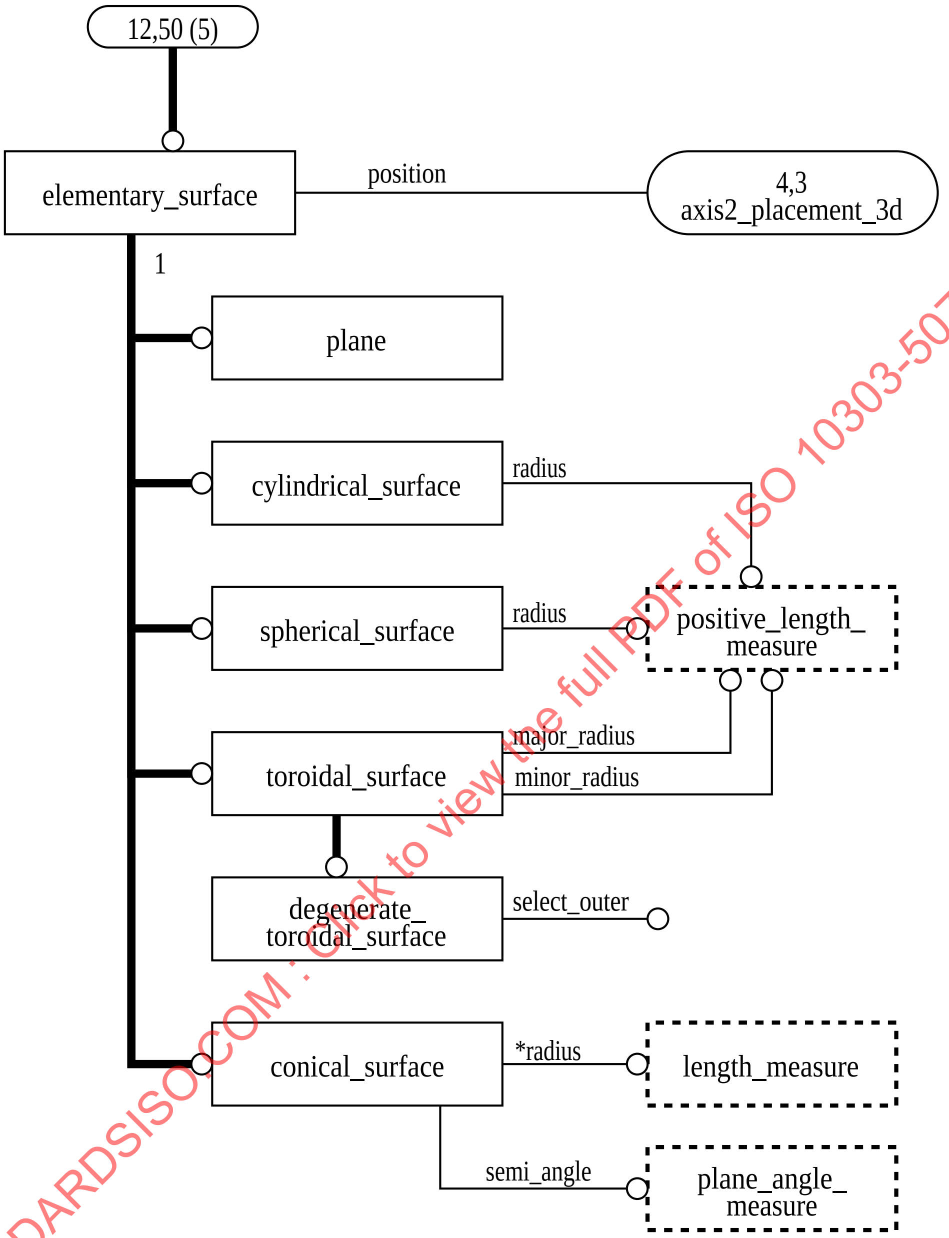




**Figure C.12 – EXPRESS-G diagram 12 of 18**

 

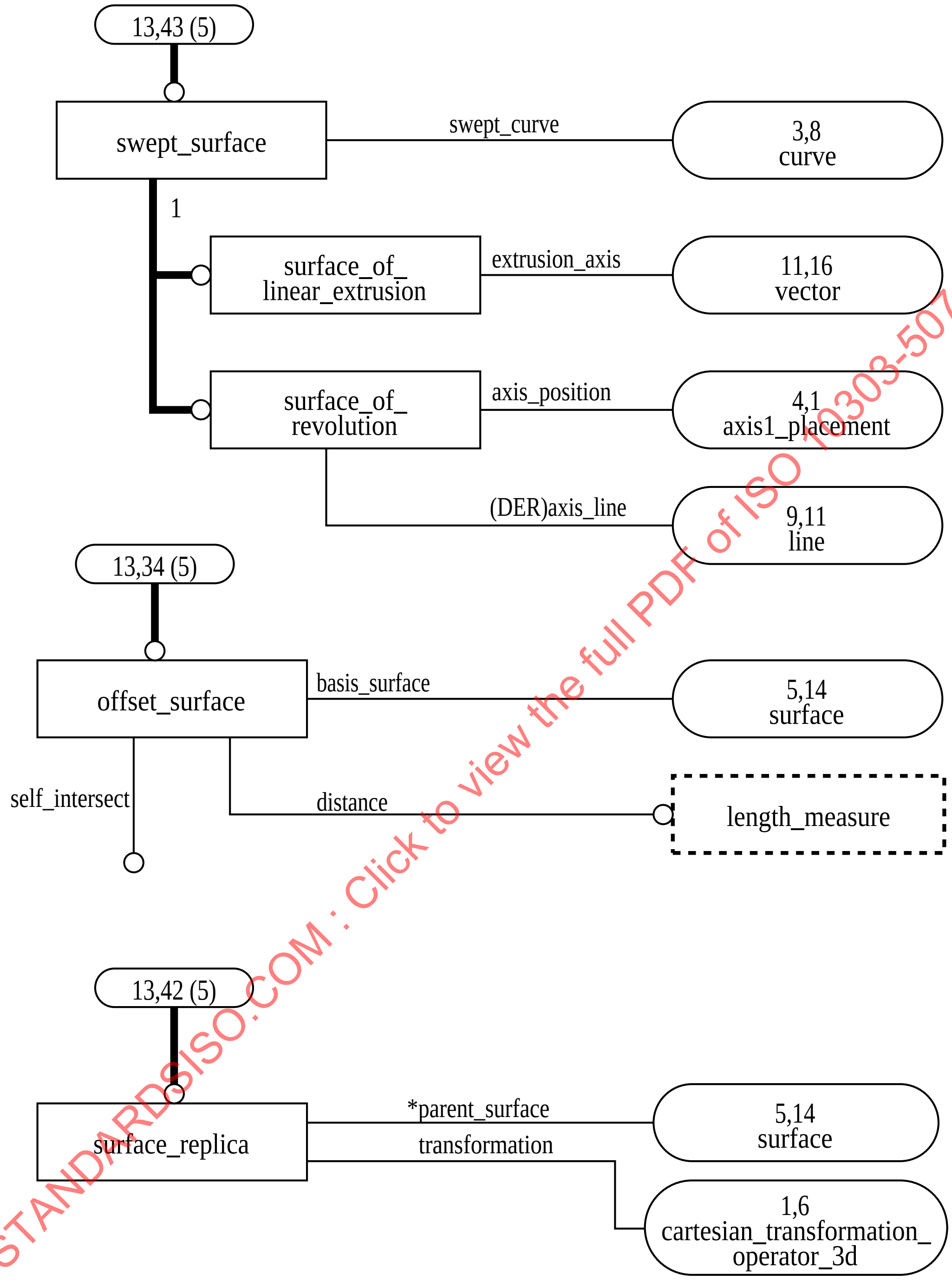


**Figure C.13 – EXPRESS-G diagram 13 of 18**