INTERNATIONAL STANDARD ISO 10303-42:1994
TECHNICAL CORRIGENDUM 3

Published 2001-07-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION

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

Part 42: Integrated generic resources: Geometric and topological representation

TECHNICAL CORRIGENDUM 3

Systèmes d'automatisation industrielle et intégration — Représentation et échange de données de produits — Partie 42: Ressources génériques intégrées: Représentation géométrique et topologique

RECTIFICATIF TECHNIQUE 3

Technical Corrigendum 3 to International Standard ISO 10303-42:1994 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 4, *Industrial data*.

ICS 25.040.40 **Ref. No. ISO 10303-42:1994/Cor.3:2001(E)**

Introduction

This Technical Corrigendum corrects ISO 10303-42:1994, *Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resources: Geometric and topological representation*. It incorporates ISO 10303-42:1994/Cor.1:1999 and ISO 10303-42:1994/Cor.2:1999, and cancels and replaces these two documents.

The purpose of the modifications to the text of ISO 10303-42:1994 is to correct errors in the EXPRESS definitions likely to cause compilation problems, to include additional EXPRESS constructs required for usage with other parts of ISO 10303, to replace the URL in the annex for the computer-interpretable EXPRESS, and to replace the object identifier for the document and the applicable schema.

Modifications to the text of ISO 10303-42:1994

Clause 4, p. 11
The entity **founded_item** *is required to be referenced since it is now a supertype of* **composite_curve_segment** *and* **surface_patch**. *Add the following to the list of REFERENCE FROM representation_schema. This additional REFERENCE was defined in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness.*

```
    founded_item,
```

Clause 4.3, p. 14
The EXPRESS specification of **dummy_gri** *was defined in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. A constant is required to initialise ENTITYs constructed within ENTITY definitions or FUNCTIONs. The CONSTANT* **dummy_gri** *is a partial definition to be used when types of* **geometric_representation_item** *are constructed. Add the following new subsubclause 4.3.1 and renumber existing subsubclauses 4.3.1 through 4.3.13 accordingly.*

4.3.1 Constant definition

The constant **dummy_gri** is a partial entity definition to be used when types of **geometric_representation_item** are constructed. It provides the correct supertypes and the **name** attribute as an empty string.

EXPRESS specification:

```
*)
CONSTANT
 dummy_gri : geometric_representation_item := representation_item('')||
                    geometric_representation_item();
END_CONSTANT;
(*
```

Clause 4.4.4, p. 23
The EXPRESS specification of this entity is amended to include the newly introduced subtypes. Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
ENTITY cartesian_point
  SUPERTYPE OF (ONEOF(cylindrical_point, polar_point, spherical_point))
  SUBTYPE OF (point);
   coordinates  : LIST [1:3] OF length_measure;
END_ENTITY;
(*
```


Clause 4.4.5, p. 23

The subtypes of **cartesian_point** *defined below are required for reference from other parts of ISO 10303. Renumber existing clauses 4.4.5 to 4.4.73 as 4.4.8 to 4.4.76, respectively. Renumber existing Figures 1 through 12 as 2 through 13 respectively. Add the following as clause 4.4.5, 4.4.6, 4.4.7 and Figure 1.*

4.4.5 cylindrical_point

A **cylindrical_point** is a type of **cartesian_point** which uses a cylindrical polar coordinate system, centred at the origin of the corresponding Cartesian coordinate system, to define its location.

EXPRESS specification:

```
*)
ENTITY cylindrical_point
  SUBTYPE OF (cartesian_point);
    r     : length_measure;
    theta : plane_angle_measure;
    z     : length_measure;
  DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure :=
                    [r*cos(theta), r*sin(theta), z];
  WHERE
   WR1: r >= 0.0;
END_ENTITY;
(*
```

Attribute definitions:

r: The distance from the point to the z axis.

theta: The angle between the plane containing the point and the z axis and the xz plane.

z: The distance from the xy plane to the point.

Formal propositions:

WR1: The radius r shall be greater than, or equal to zero.

Informal propositions:

IP1: The value of **theta** shall lie in the range $0 \leq$ **theta** < 360 degrees.

4.4.6 spherical_point

A **spherical_point** is a type of **cartesian_point** which uses a spherical polar coordinate system, centred at the origin of the corresponding Cartesian coordinate system, to define its location.

EXPRESS specification:

```
*)
ENTITY spherical_point
  SUBTYPE OF (cartesian_point);
    r     : length_measure;
    theta : plane_angle_measure;
    phi   : plane_angle_measure;
  DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure :=
      [r*sin(theta)*cos(phi), r*sin(theta)*sin(phi), r*cos(theta)];
  WHERE
   WR1: r >= 0.0;
END_ENTITY;
(*
```

Attribute definitions:

r: The distance from the point to the origin.

theta: The angle θ between the z axis and the line joining the origin to the point.

phi: The angle ϕ, measured from the x axis to the projection onto the xy plane of the line from the origin to the point.

NOTE - See Figure 1 for an illustration of the attributes.

Formal propositions:

WR1: The radius r shall be greater than, or equal to zero.

Informal propositions:

IP1: The value of **theta** shall lie in the range $0 \leq$ **theta** ≤ 180 degrees.

IP2: The value of **phi** shall lie in the range $0 \leq$ **phi** < 360 degrees.

4.4.7 polar_point

A **polar_point** is a type of **cartesian_point** which uses a two dimensional polar coordinate system, centred at the origin of the corresponding Cartesian coordinate system, to define its location.

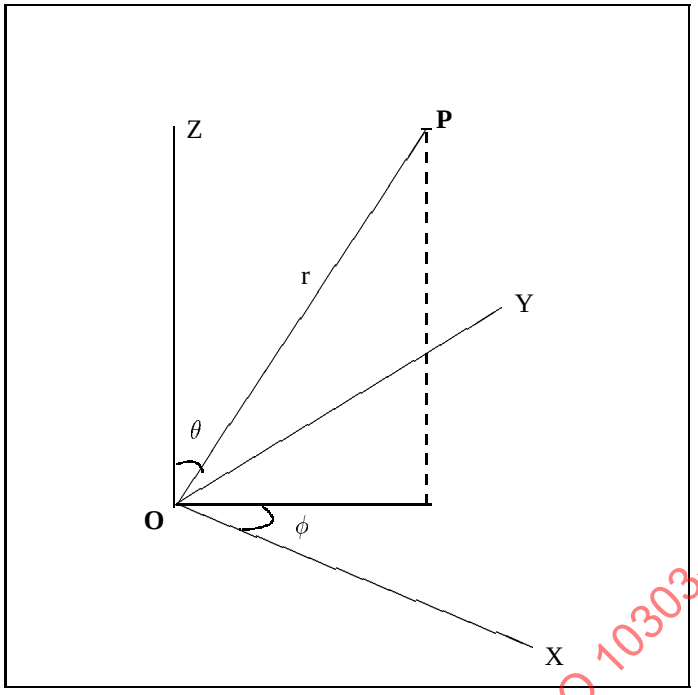

Figure 1 – Spherical_point attributes

EXPRESS specification:

```
*)
ENTITY polar_point
  SUBTYPE OF (cartesian_point);
    r     : length_measure;
    theta : plane_angle_measure;
  DERIVE
    SELF\cartesian_point.coordinates : LIST [1:3] OF length_measure :=
                  [r*cos(theta), r*sin(theta)];
  WHERE
   WR1: r >= 0.0;
END_ENTITY;
(*
```

Attribute definitions:

r: The distance from the point to the origin.

theta: The angle between the x axis and the line joining the origin to the point.

Formal propositions:

WR1: The radius r shall be greater than, or equal to zero.

Informal propositions:

IP1: The value of **theta** shall lie in the range $0 \leq$ **theta** < 360 degrees.

Clause 4.4.13, p. 28

The EXPRESS specification of the entity **axis1_placement** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the* **z** *attribute. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
ENTITY axis1_placement
 SUBTYPE OF (placement);
   axis     : OPTIONAL direction;
 DERIVE
   z : direction := NVL(normalise(axis), dummy_gri ||
                                  direction([0.0,0.0,1.0]));
 WHERE
   WR1: SELF\geometric_representation_item.dim  = 3;
 END_ENTITY;
(*
```

Clause 4.4.17, p. 34

The EXPRESS specification of the entity **cartesian_transformation_operator_3d** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification of* **cartesian_transformation_operator_3d** *contained an error in the reference path for the* **dim** *attribute. Replace WR1 of the EXPRESS specification with the following:*

```
WR1: SELF\geometric_representation_item.dim = 3;
```

Clause 4.4.18, p. 36

The EXPRESS specification of the entity **cartesian_transformation_operator_2d** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification of* **cartesian_transformation_operator_2d** *contained an error in the reference path for the* **dim** *attribute. Replace WR1 of the EXPRESS specification with the following:*

```
WR1: SELF\geometric_representation_item.dim = 2;
```


Clause 4.4.34, p. 54

The EXPRESS specification of the entity **trimmed_curve** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification of* **trimmed_curve** *contained logical errors in the WHERE rules. Replace WR1 and WR2 of the EXPRESS specification with the following:*

```
WR1: (HIINDEX(trim_1) = 1) OR (TYPEOF(trim_1[1]) <> TYPEOF(trim_1[2]));
WR2: (HIINDEX(trim_2) = 1) OR (TYPEOF(trim_2[1]) <> TYPEOF(trim_2[2]));
```

Clause 4.4.36, p. 58

The EXPRESS specification of the entity **ccomposite_curve_segment** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS definition of the* **composite_curve_segment** *entity is revised to make it a subtype of* **founded_item** *in order to provide a representation context for the* **parent_curve** *attribute. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
ENTITY composite_curve_segment
 SUBTYPE OF (founded_item);
   transition    : transition_code;
   same_sense    : BOOLEAN;
   parent_curve  : curve;
 INVERSE
   using_curves  : BAG[1:?] OF composite_curve FOR segments;
 WHERE
   WR1 : ('GEOMETRY_SCHEMA.BOUNDED_CURVE' IN TYPEOF(parent_curve));
 END_ENTITY;
```

Add the following note at the end of the entity description:

> NOTE - – Since **composite_curve_segment** is not a subtype of **geometric_representation_item** the instance of **bounded_curve** which is the **parent_curve** attribute not associated in the usual way with the **geometric_representation_context** of each **representation** using a **composite_curve** containing this **composite_curve_segment**. The **geometric_representation_context** is associated via the **founded_item** supertype.

Clause 4.4.58, p. 77

The EXPRESS specification of **surface_of_revolution** *contains an error in the DERIVE statement. The correction provided by TC1 is not accepted by all compilers. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
```



```
ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
  axis_position       : axis1_placement;
DERIVE
  axis_line : line := representation_item('')||
                      geometric_representation_item()|| curve()||
                      line(axis_position.location, representation_item('')||
                      geometric_representation_item()||
                      vector(axis_position.z, 1.0));
END_ENTITY;
(*
```

Clause 4.4.67, p. 87

The EXPRESS specification of **curve_bounded_surface** *contains a logical error in WR1 making it incompatible with the description. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
ENTITY curve_bounded_surface
   SUBTYPE OF (bounded_surface);
   basis_surface    : surface;
   boundaries       : SET [1:?] OF boundary_curve;
   implicit_outer   : BOOLEAN;
 WHERE
   WR1: (NOT implicit_outer) OR
        (SIZEOF (QUERY (temp <* boundaries |
         'GEOMETRY_SCHEMA.OUTER_BOUNDARY_CURVE' IN TYPEOF(temp))) = 0);
   WR2: (NOT(implicit_outer)) OR
          ('GEOMETRY_SCHEMA.BOUNDED_SURFACE' IN TYPEOF(basis_surface));
   WR3: SIZEOF(QUERY(temp <* boundaries |
                'GEOMETRY_SCHEMA.OUTER_BOUNDARY_CURVE' IN
                                        TYPEOF(temp))) <= 1;
   WR4: SIZEOF(QUERY(temp <* boundaries |
           (temp\composite_curve_on_surface.basis_surface [1] <>
                                        SELF.basis_surface))) = 0;
END_ENTITY;
(*
```

Clause 4.4.70, p. 90

The EXPRESS specification of the entity **rectangular_composite_surface** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification of* **rectangular_composite_surface** *uses an invalid comparison to an empty set in WR1. Replace WR1 of the EXPRESS specification with the following:*

```
WR1: SIZEOF(QUERY (s <* segments | n_v <> SIZEOF (s))) = 0 ;
```


Clause 4.4.71, p. 92

The EXPRESS specification of the entity **surface_patch** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS definition of the* **surface_patch** *entity is revised to make it a subtype of* **founded_item**. *An explanatory note is added. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
ENTITY surface_patch
 SUBTYPE OF (founded_item);
   parent_surface : bounded_surface;
   u_transition   : transition_code;
   v_transition   : transition_code;
   u_sense        : BOOLEAN;
   v_sense        : BOOLEAN;
 INVERSE
   using_surfaces : BAG[1:?] OF rectangular_composite_surface FOR segments;
 WHERE
   WR1: (NOT ('GEOMETRY_SCHEMA.CURVE_BOUNDED_SURFACE'
                IN TYPEOF(parent_surface)));
 END_ENTITY;
```

Add the following to the description of the entity:

> NOTE - – Since **surface_patch** is not a subtype of **geometric_representation_item** the instance of **bounded_surface** which is the **parent_surface** attribute is not associated, in the usual way, with the **geometric_representation_context** of each **representation** using a **rectangular_composite_surface** containing this **surface_patch**. The **geometric_representation_context** is associated via the **founded_item** supertype.

Clause 4.6.1, p. 95

The EXPRESS specification does not provide a **dimension_count** *for* **cartesian_point**, **direction** *or* **vector** *instances created within functions. return variable. Remove the description and EXPRESS specification of the function* **dimension_of** *and replace with the following:* The function **dimension_of** returns the dimensionality of the input **geometric_representation_item**.
If the item is a **cartesian_point**, **direction**, or **vector**, the dimensionality is obtained directly by counting components.

For all other other subtypes, the dimensionality is the integer **dimension_count** of a **geometric_representation_context** in which the input **geometric_representation_item** is geometrically founded.

By virtue of the constraints in global rule **compatible_dimension**, this value is the **coordinate_space_dimension** of the input **geometric_representation_item**. See 4.5.1 for definition of this rule.

EXPRESS specification:

```
 *)
FUNCTION dimension_of(item : geometric_representation_item) :
  dimension_count;
  LOCAL
    x   : SET OF representation;
    y   : representation_context;
    dim : dimension_count;
  END_LOCAL;
  -- For cartesian_point, direction, or vector dimension is determined by
  -- counting components.
    IF 'GEOMETRY_SCHEMA.CARTESIAN_POINT' IN TYPEOF(item) THEN
       dim := SIZEOF(item\cartesian_point.coordinates);
       RETURN(dim);
    END_IF;
    IF 'GEOMETRY_SCHEMA.DIRECTION' IN TYPEOF(item) THEN
       dim := SIZEOF(item\direction.direction_ratios);
       RETURN(dim);
    END_IF;
    IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(item) THEN
       dim := SIZEOF(item\vector.orientation\direction.direction_ratios);
       RETURN(dim);
    END_IF;
  -- For all other types of geometric_representation_item dim is obtained
  -- via context.
  -- Find the set of representation in which the item is used.

  x := using_representations(item);

  -- Determines the dimension_count of the
  -- geometric_representation_context. Note that the
  -- RULE compatible_dimension ensures that the context_of_items
  -- is of type geometric_representation_context and has
  -- the same dimension_count for all values of x.
  -- The SET x is non-empty since this is required by WR1 of
  -- representation_item.
    y := x[1].context_of_items;
    dim := y\geometric_representation_context.coordinate_space_dimension;
    RETURN (dim);

END_FUNCTION;
(*
```

Clause 4.6.6, p. 99

The EXPRESS specification of the function **base_axis** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **u** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION base_axis(dim : INTEGER; axis1, axis2, axis3 : direction) :
                                                    LIST [2:3] OF
direction;
  LOCAL
    u      : LIST [2:3] OF direction;
    factor : REAL;
    d1, d2 : direction;
  END_LOCAL;

  IF (dim = 3) THEN
    d1 := NVL(normalise(axis3),  dummy_gri || direction([0.0,0.0,1.0]));
    d2 := first_proj_axis(d1,axis1);
    u := [d2, second_proj_axis(d1,d2,axis2), d1];
  ELSE
     IF EXISTS(axis1) THEN
      d1 := normalise(axis1);
      u := [d1, orthogonal_complement(d1)];
      IF EXISTS(axis2) THEN
        factor := dot_product(axis2,u[2]);
        IF (factor < 0.0) THEN
          u[2].direction_ratios[1] := -u[2].direction_ratios[1];
          u[2].direction_ratios[2] := -u[2].direction_ratios[2];
        END_IF;
      END_IF;
    ELSE
      IF EXISTS(axis2) THEN
        d1 := normalise(axis2);
        u := [orthogonal_complement(d1), d1];
        u[1].direction_ratios[1] := -u[1].direction_ratios[1];
        u[1].direction_ratios[2] := -u[1].direction_ratios[2];
      ELSE
u := [dummy_gri || direction([1.0, 0.0]), dummy_gri ||
direction([0.0, 1.0])];
      END_IF;
    END_IF;
  END_IF;
  RETURN(u);
END_FUNCTION;
(*
```

Clause 4.6.7, p. 100

The EXPRESS specification of the function **build2_axes** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the return variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
 FUNCTION build_2axes(ref_direction : direction) : LIST [2:2] OF direction;
   LOCAL
     d : direction := NVL(normalise(ref_direction),
                          dummy_gri || direction([1.0,0.0]));
   END_LOCAL;

   RETURN([d, orthogonal_complement(d)]);
 END_FUNCTION;
(*
```

Clause 4.6.8, p. 100

The EXPRESS specification of the function **build_axes** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the return variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION build_axes(axis, ref_direction : direction) :
                                          LIST [3:3] OF direction;

   LOCAL
     d1, d2 : direction;
   END_LOCAL;
   d1 := NVL(normalise(axis), dummy_gri || direction([0.0,0.0,1.0]));
   d2 := first_proj_axis(d1, ref_direction);
   RETURN([d2, normalise(cross_product(d1,d2)).orientation, d1]);
END_FUNCTION;
(*
```

Clause 4.6.9, p. 101

The EXPRESS specification of the function **orthogonal_complement** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **result** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
 FUNCTION orthogonal_complement(vec : direction) : direction;
   LOCAL
     result :  direction ;
   END_LOCAL;
```



```
   IF (vec.dim <> 2) OR NOT EXISTS (vec) THEN
     RETURN(?);
   ELSE
     result := dummy_gri || direction([-vec.direction_ratios[2],
                                        vec.direction_ratios[1]]);
     RETURN(result);
   END_IF;
 END_FUNCTION;
(*
```

Clause 4.6.10, p. 102

The EXPRESS specification of function **first_proj_axis** *does not give a correct initialisation of the* **v** *variable. The function does not provide a valid result when called with* **z_axis** *in the direction [-1,0,0] and* **arg** *defaulted. Replace the second sentence of the description with:*

With **arg** defaulted the result is the projection of [1,0,0] onto this plane; except that, if **z_axis** = [1,0,0], or, **z_axis** = [-1,0,0], [0,1,0] is the default for **arg**.

Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
  LOCAL
    x_axis : direction;
    v      : direction;
    z      : direction;
    x_vec  : vector;
  END_LOCAL;

  IF (NOT EXISTS(z_axis)) THEN
    RETURN (?) ;
  ELSE
    z := normalise(z_axis);
    IF NOT EXISTS(arg) THEN
      IF ((z.direction_ratios <> [1.0,0.0,0.0]) AND
          (z.direction_ratios <> [-1.0,0.0,0.0]))  THEN
        v :=  dummy_gri || direction([1.0,0.0,0.0]);
      ELSE
        v := dummy_gri || direction([0.0,1.0,0.0]);
      END_IF;
    ELSE
      IF  (arg.dim <> 3) THEN
        RETURN (?) ;
      END_IF;
      IF ((cross_product(arg,z).magnitude) = 0.0) THEN
        RETURN (?);
```



```
      ELSE
        v := normalise(arg);
      END_IF;
    END_IF;
    x_vec := scalar_times_vector(dot_product(v, z), z);
    x_axis := vector_difference(v, x_vec).orientation;
    x_axis := normalise(x_axis);
  END_IF;
  RETURN(x_axis);
END_FUNCTION;
(*
```

Clause 4.6.11, p. 103

The EXPRESS specification of the function **second_proj_axis** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the* **v** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION second_proj_axis(z_axis, x_axis, arg: direction) : direction;
  LOCAL
    y_axis : vector;
    v      : direction;
    temp   : vector;
  END_LOCAL;

  IF NOT EXISTS(arg) THEN
    v := dummy_gri || direction([0.0,1.0,0.0]);
  ELSE
    v := arg;
  END_IF;

  temp   := scalar_times_vector(dot_product(v, z_axis), z_axis);
  y_axis := vector_difference(v, temp);
  temp   := scalar_times_vector(dot_product(v, x_axis), x_axis);
  y_axis := vector_difference(y_axis, temp);
  y_axis := normalise(y_axis);
  RETURN(y_axis.orientation);
END_FUNCTION;
(*
```

Clause 4.6.12, p. 103

The EXPRESS specification of the function **cross_product** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **res** *and* **result** *variables. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION cross_product (arg1, arg2 : direction) : vector;
  LOCAL
    mag    : REAL;
    res    : direction;
    v1,v2  : LIST[3:3] OF REAL;
    result : vector;
  END_LOCAL;

  IF ( NOT EXISTS (arg1) OR (arg1.dim = 2)) OR
     ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
    RETURN(?);
  ELSE
    BEGIN
      v1  := normalise(arg1).direction_ratios;
      v2  := normalise(arg2).direction_ratios;
      res := dummy_gri || direction([(v1[2]*v2[3] - v1[3]*v2[2]),
    (v1[3]*v2[1] - v1[1]*v2[3]), (v1[1]*v2[2] - v1[2]*v2[1])]);
      mag := 0.0;
      REPEAT i := 1 TO 3;
        mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
      END_REPEAT;
      IF (mag > 0.0) THEN
        result := dummy_gri || vector(res, SQRT(mag));
      ELSE
        result := dummy_gri || vector(arg1, 0.0);
      END_IF;
      RETURN(result);
    END;
  END_IF;
END_FUNCTION;
(*
```

Clause 4.6.14, p. 105

The EXPRESS specification of the function **normalise** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **v** *variable and may corrupt the input parameter* **arg**. *Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION normalise (arg : vector_or_direction) : vector_or_direction;
   LOCAL
     ndim   : INTEGER;
     v      : direction;
     result : vector_or_direction;
```



```
    vec    : vector;
    mag    : REAL;
  END_LOCAL;

  IF NOT EXISTS (arg) THEN
    result := ?;
    (* When function is called with invalid data a NULL result is returned *)
  ELSE
    ndim := arg.dim;
    IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
      BEGIN
            v := dummy_gri || direction(arg.orientation.direction_ratios);
        IF arg.magnitude = 0.0 THEN
          RETURN(?);
        ELSE
         vec := dummy_gri || vector (v, 1.0);
        END_IF;
      END;
    ELSE
      v := dummy_gri || direction (arg.direction_ratios);
    END_IF;
    mag := 0.0;
    REPEAT  i := 1 TO ndim;
      mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
    END_REPEAT;
    IF mag > 0.0 THEN
      mag := SQRT(mag);
      REPEAT  i := 1 TO ndim;
        v.direction_ratios[i] := v.direction_ratios[i]/mag;
      END_REPEAT;
      IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
        vec.orientation := v;
        result := vec;
      ELSE
        result := v;
      END_IF;
    ELSE
      RETURN(?);
    END_IF;
  END_IF;
  RETURN (result);
END_FUNCTION;
(*
```

Clause 4.6.15, p. 107

The EXPRESS specification of the function **scalar_times_vector** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **result** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION scalar_times_vector (scalar : REAL; vec : vector_or_direction)
                                      : vector;
  LOCAL
    v      : direction;
    mag    : REAL;
    result : vector;
  END_LOCAL;

  IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
    RETURN (?) ;
   ELSE
    IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF (vec) THEN
      v   := dummy_gri || direction(vec.orientation.direction_ratios);
      mag := scalar * vec.magnitude;
    ELSE
      v   := dummy_gri || direction(vec.direction_ratios);
      mag := scalar;
    END_IF;
    IF (mag < 0.0 ) THEN
      REPEAT i := 1 TO SIZEOF(v.direction_ratios);
        v.direction_ratios[i] := -v.direction_ratios[i];
      END_REPEAT;
      mag := -mag;
    END_IF;
    result := dummy_gri || vector(normalise(v), mag);
  END_IF;
  RETURN (result);
END_FUNCTION;
(*
```

Clause 4.6.16, p. 108

The EXPRESS specification of the function **vector_sum** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **res** *and* **result** *variables. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION vector_sum(arg1, arg2 : vector_or_direction) : vector;
  LOCAL
    result          : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim            : INTEGER;
  END_LOCAL;
```



```
  IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <> arg2.dim)
      THEN
    RETURN (?) ;

  ELSE
    BEGIN
      IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1.0;
        vec1 := arg1;
      END_IF;
      IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1.0;
        vec2 := arg2;
      END_IF;
      vec1 := normalise (vec1);
      vec2 := normalise (vec2);
      ndim := SIZEOF(vec1.direction_ratios);
      mag := 0.0;
      res := dummy_gri || direction(vec1.direction_ratios);
      REPEAT i := 1 TO ndim;
        res.direction_ratios[i] := mag1*vec1.direction_ratios[i] +
                                   mag2*vec2.direction_ratios[i];
        mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
      END_REPEAT;
      IF (mag > 0.0 ) THEN
      result := dummy_gri || vector( res, SQRT(mag));
      ELSE
        result := dummy_gri || vector( vec1,  0.0);
      END_IF;
    END;
  END_IF;
  RETURN (result);
END_FUNCTION;
(*
```

Clause 4.6.17, p. 109

The EXPRESS specification of the function **vector_difference** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **res** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION vector_difference(arg1, arg2 : vector_or_direction) : vector;
  LOCAL
    result          : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim            : INTEGER;
  END_LOCAL;

  IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <> arg2.dim)
      THEN
    RETURN (?) ;
   ELSE
    BEGIN
      IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1.0;
        vec1 := arg1;
      END_IF;
      IF 'GEOMETRY_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1.0;
        vec2 := arg2;
      END_IF;
      vec1 := normalise (vec1);
      vec2 := normalise (vec2);
      ndim := SIZEOF(vec1.direction_ratios);
      mag := 0.0;
      res := dummy_gri || direction(vec1.direction_ratios);
      REPEAT i := 1 TO ndim;
        res.direction_ratios[i] := mag1*vec1.direction_ratios[i] +
                                   mag2*vec2.direction_ratios[i];
        mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
      END_REPEAT;
      IF (mag > 0.0 ) THEN
      result := dummy_gri || vector( res, SQRT(mag));
      ELSE
        result := dummy_gri || vector( vec1,  0.0);
      END_IF;
    END;
  END_IF;
  RETURN (result);
END_FUNCTION;
(*
```


Clause 4.6.18, p. 110

The EXPRESS specification of the function **default_b_spline_knot_mult** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **knot_mult** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
 FUNCTION default_b_spline_knot_mult(degree, up_knots : INTEGER;
                                          uniform : knot_type)
                                                  : LIST [2:?] OF INTEGER;
   LOCAL
     knot_mult : LIST [1:up_knots] OF INTEGER;
   END_LOCAL;

   IF uniform = uniform_knots THEN
     knot_mult := [1:up_knots];
   ELSE
     IF uniform = quasi_uniform_knots THEN
       knot_mult := [1:up_knots];
       knot_mult[1] := degree + 1;
       knot_mult[up_knots] := degree + 1;
     ELSE
       IF uniform = piecewise_bezier_knots THEN
         knot_mult := [degree:up_knots];
         knot_mult[1] := degree + 1;
         knot_mult[up_knots] := degree + 1;
       ELSE
         knot_mult := [0:up_knots];
       END_IF;
     END_IF;
   END_IF;
   RETURN(knot_mult);
 END_FUNCTION;
(*
```

Clause 4.6.19, p. 111

The EXPRESS specification of the function **default_b_spline_knots** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the* **knots** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION default_b_spline_knots(degree,up_knots : INTEGER;
```



```
                                uniform : knot_type)
                                        : LIST [2:?] OF parameter_value;
  LOCAL
    knots  : LIST [1:up_knots] OF parameter_value := [0:up_knots];
    ishift : INTEGER := 1;
  END_LOCAL;

  IF (uniform = uniform_knots) THEN
     ishift := degree + 1;
  END_if;
  IF (uniform = uniform_knots) OR
     (uniform = quasi_uniform_knots) OR
     (uniform = piecewise_bezier_knots) THEN

    REPEAT i := 1 TO up_knots;
      knots[i] := i - ishift;
    END_REPEAT;
  END_IF;
  RETURN(knots);
END_FUNCTION;
(*
```

Clause 4.6.20, p. 112

The EXPRESS specification of the function **default_b_spline_curve_weights** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the return variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
 FUNCTION default_b_spline_curve_weights(up_cp : INTEGER)
                                        : ARRAY [0:up_cp] OF REAL;
   RETURN([1:up_cp + 1]);
 END_FUNCTION;
(*
```

Clause 4.6.21, p. 113

The EXPRESS specification of the function **default_b_spline_surface_weights** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not assign correct values to the return variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
```



```
FUNCTION default_b_spline_surface_weights(u_upper, v_upper: INTEGER)
                                        : ARRAY [0:u_upper] OF
                                          ARRAY [0:v_upper] OF REAL;
   RETURN([[1:v_upper + 1]:u_upper +1]);
END_FUNCTION;
(*
```

Clause 4.6.22, p. 114

The EXPRESS specification of the function **constraints_param_b_spline** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification contained some redundant variable declarations. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION constraints_param_b_spline(degree, up_knots, up_cp : INTEGER;
                                                    knot_mult : LIST OF INTEGER;
                          knots : LIST OF parameter_value) : BOOLEAN;
   LOCAL
     result  : BOOLEAN := TRUE;
     k, sum  : INTEGER;
   END_LOCAL;

   (* Find sum of knot multiplicities. *)
   sum := knot_mult[1];

   REPEAT i := 2 TO up_knots;
     sum := sum + knot_mult[i];
   END_REPEAT;

   (* Check limits holding for all B-spline parametrisations *)
   IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR
         (sum <> (degree + up_cp + 2)) THEN
     result := FALSE;
     RETURN(result);
   END_IF;

   k := knot_mult[1];

   IF (k < 1) OR (k > degree + 1) THEN
     result := FALSE;
     RETURN(result);
   END_IF;

   REPEAT i := 2 TO up_knots;
     IF (knot_mult[i] < 1) OR (knots[i] <= knots[i-1]) THEN
       result := FALSE;
       RETURN(result);
```

```
    END_IF;

    k := knot_mult[i];

    IF (i < up_knots) AND (k > degree) THEN
      result := FALSE;
      RETURN(result);
    END_IF;

    IF (i = up_knots) AND (k > degree + 1) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
END_FUNCTION;
(*
```

Clause 4.6.25, p. 116

The EXPRESS specification of the function **get_basis_surface** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification of the* **get_basis_surface** *function contains an error in the reference path to the* **segmemts** *attribute for a* **composite_curve_on_surface**. *Remove the EXPRESS specification and replace with the folowing:*

EXPRESS specification:

```
*)
FUNCTION get_basis_surface (c : curve_on_surface) : SET[0:2] OF surface;
  LOCAL
    surfs  : SET[0:2] OF surface;
    n      : INTEGER;
  END_LOCAL;
  surfs := [];
  IF 'GEOMETRY_SCHEMA.PCURVE' IN TYPEOF (c) THEN
    surfs := [c\pcurve.basis_surface];
  ELSE
    IF 'GEOMETRY_SCHEMA.SURFACE_CURVE' IN TYPEOF (c) THEN
      n := SIZEOF(c\surface_curve.associated_geometry);
      REPEAT i := 1 TO n;
      surfs := surfs +
               associated_surface(c\surface_curve.associated_geometry[i]);
      END_REPEAT;
    END_IF;
  END_IF;
  IF 'GEOMETRY_SCHEMA.COMPOSITE_CURVE_ON_SURFACE' IN TYPEOF (c) THEN
   (* For a composite_curve_on_surface the basis_surface is the intersection
    of the basis_surfaces of all the segments. *)
     n := SIZEOF(c\composite_curve.segments);
     surfs := get_basis_surface(c\composite_curve.segments[1].parent_curve);
```



```
    IF n > 1 THEN
      REPEAT i := 2 TO n;
        surfs := surfs * get_basis_surface(c\composite_curve.
                                           segments[i].parent_curve);
      END_REPEAT;
    END_IF;
  END_IF;
  RETURN(surfs);
END_FUNCTION;
(*
```

Clause 4.6.28, p. 119

The EXPRESS specification of the function **list_to_array** *does not correctly initialise the array* **res**. *The correction provided in ISO 10303-42:1994/Cor. 2:1999 introduced a new error by omitting the array dimensions for the return variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
 *)
 FUNCTION list_to_array(lis : LIST [0:?] OF GENERIC : T;
                        low,u : INTEGER) : ARRAY [low:u] OF GENERIC : T;
   LOCAL
     n   : INTEGER;
     res : ARRAY [low:u] OF GENERIC : T;
   END_LOCAL;

   n := SIZEOF(lis);
   IF (n <> (u-low +1)) THEN
     RETURN(?);
   ELSE
     res := [lis[1] : n];
     REPEAT i := 2 TO n;
       res[low+i-1] := lis[i];
     END_REPEAT;
     RETURN(res);
   END_IF;
 END_FUNCTION;
(*
```

Clause 4.6.29, p. 120

The EXPRESS specification of the function **make_array_of_array** *does not correctly initialise the array* **res**. *The correction provided in ISO 10303-42:1994/Cor. 2:1999 introduced a new error by omitting the array dimensions for the return variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
 *)
 FUNCTION make_array_of_array(lis : LIST[1:?] OF LIST [1:?] OF GENERIC : T;
                              low1, u1, low2, u2 : INTEGER):
                 ARRAY [low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
   LOCAL
     res   : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
   END_LOCAL;

(* Check input dimensions for consistency *)
   IF (u1-low1+1) <> SIZEOF(lis) THEN
     RETURN (?);
   END_IF;
   IF (u2 - low2 + 1 ) <> SIZEOF(lis[1]) THEN
     RETURN (?) ;
   END_IF;
(* Initialise res with values from lis[1] *)
   res := [list_to_array(lis[1], low2, u2) : (u1-low1 + 1)];
   REPEAT i := 2 TO HIINDEX(lis);
     IF (u2-low2+1) <> SIZEOF(lis[i]) THEN
       RETURN (?);
     END_IF;
     res[low1+i-1] := list_to_array(lis[i], low2, u2);
   END_REPEAT;

   RETURN (res);
 END_FUNCTION;
(*
```

Clause 5.3, p. 127

The constant **dummy_tri** *was added in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. A constant is required to initialise ENTITYs constructed within ENTITY definitions or FUNCTIONS. The CONSTANT* **dummy_tri** *is a partial definition to be used when types of* **topological_representation_item** *are constructed. Add the following new clause 5.3.1 and renumber existing clauses 5.3.1 through 5.3.5 accordingly.*

5.3.1 Constant definition

The constant **dummy_tri** is a partial entity definition to be used when types of **topological_representation_item** are constructed. It provides the correct supertypes and the **name** attribute as an empty string.

EXPRESS specification:

```
*)
 CONSTANT
    dummy_tri : topological_representation_item := representation_item('')||
                   topological_representation_item();
```



```
 END_CONSTANT;
(*
```

Clause 5.5.3, p. 154

The EXPRESS specification of the function **edge_reversed** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the* **the_reverse** *variable. Additional clarification to values of the attribute are given. The declared types for the return variable and the local variable are changed to the subtype actually returned.*

Remove:
This function returns an **edge** equivalent to the input **edge** except that the orientation is reversed.

Replace with:
This function returns an **oriented_edge** equivalent to the input **edge** except that the orientation is reversed.

Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
FUNCTION edge_reversed (an_edge : edge) : oriented_edge;
  LOCAL
    the_reverse : oriented_edge;
  END_LOCAL;

  IF ('TOPOLOGY_SCHEMA.ORIENTED_EDGE' IN TYPEOF (an_edge) ) THEN
    the_reverse := dummy_tri ||
            edge(an_edge.edge_end, an_edge.edge_start) ||
            oriented_edge(an_edge\oriented_edge.edge_element,
                      NOT (an_edge\oriented_edge.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
            edge(an_edge.edge_end, an_edge.edge_start) ||
            oriented_edge(an_edge, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;
(*
```

Remove the argument definition and replace with the following:

the_reverse: (output) The **oriented_edge** that is the result of the orientation reversal.

Clause 5.5.4, p. 155

The EXPRESS specification of the function **path_reversed** *was corrected in ISO 10303-42: 1994/Cor.*

2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the **the_reverse** *variable. The declared types for the return variable and the local variable are changed to the subtype actually returned.*

Remove:
This function returns a **path** equivalent to the input **path** except that the orientation is reversed.

Replace with:
This function returns an **oriented_path** equivalent to the input **path** except that the orientation is reversed.

Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
FUNCTION path_reversed (a_path : path) : oriented_path;
  LOCAL
    the_reverse : oriented_path ;
  END_LOCAL;
  IF ('TOPOLOGY_SCHEMA.ORIENTED_PATH' IN TYPEOF (a_path) ) THEN
    the_reverse := dummy_tri ||
       path(list_of_topology_reversed (a_path.edge_list)) ||
          oriented_path(a_path\oriented_path.path_element,
                        NOT(a_path\oriented_path.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
                   path(list_of_topology_reversed (a_path.edge_list)) ||
                       oriented_path(a_path, FALSE);
  END_IF;

  RETURN (the_reverse);
END_FUNCTION;
(*
```

Remove the argument definition and replace with the following:

the_reverse: (output) The **oriented_path** which is the result of the orientation reversal.

Clause 5.5.5, p. 155

The EXPRESS specification of the function **face_bound_reversed** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the* **the_reverse** *variable, the return value is of type* **face_outer_bound** *when* **a_face_bound** *is of this type.*

Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
FUNCTION face_bound_reversed (a_face_bound : face_bound) : face_bound;
  LOCAL
    the_reverse : face_bound ;
  END_LOCAL;
  IF ('TOPOLOGY_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF (a_face_bound) ) THEN
    the_reverse := dummy_tri ||
                     face_bound(a_face_bound\face_bound.bound,
                          NOT (a_face_bound\face_bound.orientation))
                           || face_outer_bound() ;
  ELSE
    the_reverse := dummy_tri ||
                face_bound(a_face_bound.bound, NOT(a_face_bound.orientation));
  END_IF;
 RETURN (the_reverse);
END_FUNCTION;
(*
```

Clause 5.5.6, p. 156

The EXPRESS specification of the function **face_reversed** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the* **the_reverse** *variable. The declared types for the return variable and the local variable are changed to the subtype actually returned.*

Remove:

This function returns a **face** equivalent to input **face** except that the orientation is reversed.

Replace with:

This function returns an **oriented_face** equivalent to input **face** except that the orientation is reversed.

Remove the EXPRESS definition and replace with the following:

EXPRESS specification:

```
*)
FUNCTION face_reversed (a_face : face) : oriented_face;
  LOCAL
    the_reverse : oriented_face ;
  END_LOCAL;
  IF ('TOPOLOGY_SCHEMA.ORIENTED_FACE' IN TYPEOF (a_face) ) THEN
    the_reverse := dummy_tri ||
      face(set_of_topology_reversed(a_face.bounds)) ||
         oriented_face(a_face\oriented_face.face_element,
                          NOT (a_face\oriented_face.orientation)) ;
  ELSE
```



```
    the_reverse := dummy_tri ||
      face(set_of_topology_reversed(a_face.bounds)) ||
                              oriented_face(a_face, FALSE) ;
  END_IF;
     RETURN (the_reverse);
END_FUNCTION;
(*
```

Remove the argument definition and replace with the following:

the_reverse: (output) The **oriented_face** which is the result of the orientation reversal.

Clause 5.5.7, p. 156

The EXPRESS specification of the function **shell_reversed** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification does not give a correct initialisation of the* **the_reverse** *variable and intermediate values of the variable. The structure of the function is simplified by making use of two new simpler functions.*

Remove:
This function returns a **shell** equivalent to the input **shell** except that the orientation is reversed.

Replace with:
This function returns an **oriented_open_shell** or **oriented_closed_shell** equivalent to the input **shell** except that the orientation is reversed.

Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
 FUNCTION shell_reversed (a_shell : shell) : shell;
   IF ('TOPOLOGY_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
     RETURN (open_shell_reversed (a_shell));
   ELSE
     IF ('TOPOLOGY_SCHEMA.CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
       RETURN (closed_shell_reversed (a_shell));
     ELSE
       RETURN (?);
     END_IF;
   END_IF;
 END_FUNCTION;
(*
```

Clause 5.5.8, p. 157

With the changes identified in this Technical Corrigendum additional topology_schema functions are required. Re-number existing clauses 5.5.8 to 5.5.20 as 5.5.10 to 5.5.22 respectively. Add the following functions as clauses 5.5.8 and 5.5.9.

5.5.8 closed_shell_reversed

This function returns an **oriented_closed_shell** or equivalent to the input **closed_shell** except that the orientation is reversed.

EXPRESS specification:

```
*)
 FUNCTION closed_shell_reversed (a_shell : closed_shell) :
                                        oriented_closed_shell;
  LOCAL
    the_reverse : oriented_closed_shell;
  END_LOCAL;
   IF ('TOPOLOGY_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
      the_reverse := dummy_tri ||
                     connected_face_set (
                        a_shell\connected_face_set.cfs_faces) ||
                     closed_shell () || oriented_closed_shell(
                        a_shell\oriented_closed_shell.closed_shell_element,
                        NOT(a_shell\oriented_closed_shell.orientation));
   ELSE
      the_reverse := dummy_tri ||
                    connected_face_set (
                      a_shell\connected_face_set.cfs_faces) ||
                    closed_shell () || oriented_closed_shell (a_shell, FALSE);
   END_IF;
   RETURN (the_reverse);
 END_FUNCTION;
 (*
```

Argument definitions:

a_shell: (input) The **closed_shell** which is to have its orientation reversed.

the_reverse: (output) The result of the orientation reversal.

5.5.9 open_shell_reversed

This function returns an **oriented_open_shell** or equivalent to the input **open_shell** except that the orientation is reversed.

EXPRESS specification:

```
*)
 FUNCTION open_shell_reversed ( a_shell : open_shell) :
```

```
                                          oriented_open_shell;
  LOCAL
    the_reverse : oriented_open_shell;
  END_LOCAL;
  IF ('TOPOLOGY_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
    the_reverse := dummy_tri ||
                   connected_face_set (
                       a_shell\connected_face_set.cfs_faces) ||
                   open_shell () || oriented_open_shell(
                     a_shell\oriented_open_shell.open_shell_element,
                       (NOT (a_shell\oriented_open_shell.orientation)));
  ELSE
    the_reverse := dummy_tri ||
                   connected_face_set (
                       a_shell\connected_face_set.cfs_faces) ||
                   open_shell () ||  oriented_open_shell (a_shell, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;
(*
```

Argument definitions:

a_shell: (input) The **open_shell** which is to have its orientation reversed.

the_reverse: (output) The result of the orientation reversal.

Clause 5.5.10, p. 158

The EXPRESS specification of the function **boolean_choose** *was corrected in ISO 10303-42: 1994/Cor. 1:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS in FUNCTION* **boolean_choose** *uses an unlabeled GENERIC data type as the result type. According to ISO 10303-11 clause 9.5.3.2 Generic data type, rule b, when a GENERIC data type is used as the result of a FUNCTION, a type label is required, and shall refer to type labels declared by the formal parameters. The formal parameters and the return types must be given type labels. The type label proposed is "item". Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION boolean_choose (b : boolean;
        choice1, choice2 : generic : item) : generic : item;

  IF b THEN
    RETURN (choice1);
  ELSE
    RETURN (choice2);
```



```
   END_IF;
END_FUNCTION;
(*
```

Clause 5.5.17, p. 162

The EXPRESS specification of the function **mixed_loop_type_set** *was corrected in ISO 10303-42: 1994/Cor. 2:1999 and is included in this Technical Corrigendum for completeness. The EXPRESS specification contains a declaration of an unused variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
 FUNCTION mixed_loop_type_set(l: SET[0:?] OF loop): LOGICAL;
    LOCAL
      poly_loop_type: LOGICAL;
    END_LOCAL;
    IF(SIZEOF(l) <= 1) THEN
      RETURN(FALSE);
    END_IF;
    poly_loop_type := ('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF(l[1]));
    REPEAT i := 2 TO SIZEOF(l);
      IF(('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF(l[i])) <> poly_loop_type) THEN
         RETURN(TRUE);
       END_IF;
    END_REPEAT;
    RETURN(FALSE);
 END_FUNCTION;
(*
```

Clause 6, p. 166

The EXPRESS specification of **geometric_model_schema** *requires an additional external reference to support the corrected definition of the* **box_domain** *entity. Remove the definition and the EXPRESS specification and replace with the following:*

The following EXPRESS declaration begins the **geometric_model_schema** and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA geometric_model_schema;
   REFERENCE FROM geometry_schema;
   REFERENCE FROM topology_schema;
   REFERENCE FROM measure_schema(length_measure,
                                 positive_length_measure,
```



```
                                  plane_angle_measure,
                                  plane_angle_unit,
                                  positive_plane_angle_measure);
  REFERENCE FROM representation_schema(founded_item);
(*
```

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

`geometry_schema`	Clause 4 of this part of ISO 10303
`topology_schema`	Clause 5 of this part of ISO 10303
`measure_schema`	ISO 10303-41
`representation_schema`	ISO 10303-43

NOTE 2 - See annex D, figures D.16 - D.18, for a graphical presentation of this schema.

Clause 6.4.15, p. 181

The EXPRESS specification of **revolved_face_solid** *contains an error in the DERIVE statement. The correction provided by TC1 is not accepted by all compilers. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
ENTITY revolved_face_solid  SUBTYPE OF (swept_face_solid);
  axis  : axis1_placement;
  angle : plane_angle_measure;
DERIVE
  axis_line : line := representation_item('')||
                      geometric_representation_item()|| curve()||
                      line(axis.location, representation_item('')||
                      geometric_representation_item()||
                      vector(axis.z, 1.0));
END_ENTITY;
(*
```

Clause 6.4.18, p. 184

The EXPRESS specification of **revolved_area_solid** *contains an error in the DERIVE statement. The correction provided by TC1 is not accepted by all compilers. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
ENTITY revolved_area_solid
```


