INTERNATIONAL STANDARD ISO 10303-42:1994
TECHNICAL CORRIGENDUM 1

Published 1999-07-15

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

Part 42:
Integrated generic resources: Geometric and topological representation

TECHNICAL CORRIGENDUM 1

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

Partie 42: Ressources génériques intégrées: Représentation géométrique et topologique

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 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*.

Introduction

This document corrects ISO 10303-42:1994, Product data representation and exchange - Part 42: Integrated generic resources: Geometric and topological representation. The corrected document supersedes ISO 10303-42:1994.

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 correct a logical error in the founding of composite_curve_segment and surface_patch, to replace the annex for the computer-interpretable EXPRESS with a URL reference, to replace the object identifier for the the document and the applicable schema, and to correct a logical error in an entity definition.

ICS 25.040.40 **Ref. No. ISO 10303-42:1994/Cor.1:1999(E)**

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.*

```
    founded_item,
```

Clause 4.3, p. 14

A constant is required to initialise ENTITYs constructed within ENTITY definitions or FUNCTION s. 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.

<u>EXPRESS specification:</u>

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

Clause 4.4.13, p. 28

The EXPRESS specification does not give a correct initialisation of the **z** *attribute. Remove the EXPRESS specification and replace with the following:*

<u>EXPRESS specification:</u>

```
*)
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 **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 **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 **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 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 does not give a correct initialisation of the **axis_line** *attribute. 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 := dummy_gri || curve() || line(axis_position.location,
                         dummy_gri || vector(axis_position.z, 1.0));
 END_ENTITY;
(*
```

Clause 4.4.70, p. 90

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 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. 96

The EXPRESS specification of function **dimension_of** *requires a further explanatory note.*

Remove the EXPRESS specification and replace with the following:

EXPRESS specification:

```
*)
FUNCTION dimension_of(item : geometric_representation_item) :
  dimension_count;
  LOCAL
    x : SET OF representation;
    y : representation_context;
  END_LOCAL;

  -- 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;
    RETURN (y\geometric_representation_context.coordinate_space_dimension);

END_FUNCTION;
(*
```

Clause 4.6.6, p. 99

The EXPRESS specification does not give a correct initialisation of 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;
  END_LOCAL;

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

Clause 4.6.7, p. 100

The EXPRESS specification does not give a correct initialisation of the **u** *variable. Remove the EXPRESS specification and replace with the following:*

EXPRESS specification:

```
*)
FUNCTION build_2axes(ref_direction : direction) : LIST [2:2] OF
direction;
  LOCAL
    u : LIST[2:2] OF direction;
  END_LOCAL;

  u[1] := NVL(normalise(ref_direction), dummy_gri ||
direction([1.0,0.0]));
  u[2] := orthogonal_complement(u[1]);
  RETURN(u);
END_FUNCTION;
(*
```

Clause 4.6.8, p. 100
The EXPRESS specification does not give a correct initialisation of the **u** *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
    u : LIST[3:3] OF direction;
    d : direction;
  END_LOCAL;

     d := NVL(normalise(axis), dummy_gri || direction([0.0,0.0,1.0]));
  u[1] := first_proj_axis(d,ref_direction);
  u[2] := normalise(cross_product(d,u[1])).orientation;
  u[3] := d;
 RETURN(u);
END_FUNCTION;
(*
```

Clause 4.6.9, p. 101
The EXPRESS specification does not give a correct initialisation of the **result.name** *attribute. 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.name := '' ;
    result.direction_ratios[1] := -vec.direction_ratios[2];
    result.direction_ratios[2] := vec.direction_ratios[1];
    RETURN(result);
  END_IF;
END_FUNCTION;
(*
```

Clause 4.6.10, p. 102

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 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]) 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 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 does not give a correct initialisation of the **res.name** *attribute. 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.name := '' ;
      res.direction_ratios[1] := (v1[2]*v2[3] - v1[3]*v2[2]);
      res.direction_ratios[2] := (v1[3]*v2[1] - v1[1]*v2[3]);
      res.direction_ratios[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.orientation := res;
        result.magnitude := SQRT(mag);
      ELSE
        result.orientation := arg1;
        result.magnitude := 0.0;
      END_IF;
      result.name := '' ;
      RETURN(result);
    END;
  END_IF;
END_FUNCTION;
(*
```

Clause 4.6.15, p. 107

The EXPRESS specification does not give a correct initialisation of the **result.name** *attribute. The assignment of the* **result** *vector as an indeterminate (?) value was deleted, the function should return an indeterminate value, not the* **result** *vector. Remove the EXPRESS specification and replace with the following:*

<ins>EXPRESS specification:</ins>

```
*)
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   := vec.orientation;
      mag := scalar * vec.magnitude;
    ELSE
      v   := vec;
      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.name := '' ;
    result.orientation := normalise(v);
    result.magnitude   := mag;
  END_IF;
  RETURN (result);
END_FUNCTION;
(*
```

Clause 4.6.16, p. 108

The EXPRESS specification does not give a correct initialisation of the **result.name** *attribute. The assignment of the* **result** *vector as an indeterminate (?) value was deleted, the function should return an indeterminate value, not the* **result** *vector. Remove the EXPRESS specification and replace with the following:*

<u>EXPRESS specification:</u>

```
*)
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;
      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;
      result.name := '' ;
      IF (mag > 0.0 ) THEN
        result.magnitude := SQRT(mag);
        result.orientation := res;
      ELSE
        result.magnitude := 0.0;
        result.orientation := vec1;
      END_IF;
    END;
  END_IF;
  result.name := '' ;
  RETURN (result);
END_FUNCTION;
(*
```

Clause 4.6.17, p. 109

The EXPRESS specification does not give a correct initialisation of the **result.name** *attribute. The assignment of the* **result** *vector as an indeterminate (?) value was deleted, the function should return an indeterminate value, not the* **result** *vector. 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;
      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.magnitude := SQRT(mag);
        result.orientation := res;
      ELSE
        result.magnitude := 0.0;
        result.orientation := vec1;
      END_IF;
    END;
  END_IF;
  result.name := '' ;
  RETURN (result);
END_FUNCTION;
(*
```

Clause 4.6.25, p. 116

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 5.5.3, p. 154

The EXPRESS specification does not give a correct initialisation of the **the_reverse.name** *attribute. 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;
  the_reverse.name := '' ;
  the_reverse.edge_start := an_edge.edge_end ;
  the_reverse.edge_end := an_edge.edge_start ;
  IF ('TOPOLOGY_SCHEMA.ORIENTED_EDGE' IN TYPEOF (an_edge) ) THEN
    the_reverse.edge_element := an_edge\oriented_edge.edge_element ;
    the_reverse.orientation := NOT (an_edge\oriented_edge.orientation) ;
  ELSE
    the_reverse.edge_element := an_edge ;
    the_reverse.orientation :=  FALSE ;
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;
(*
```

Remove the output description 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 does not give a correct initialisation of the **the_reverse.name** *attribute. 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 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;
  the_reverse.name := '' ;
  the_reverse.edge_list := list_of_topology_reversed (a_path.edge_list) ;
  IF ('TOPOLOGY_SCHEMA.ORIENTED_PATH' IN TYPEOF (a_path) ) THEN
    the_reverse.path_element := a_path\oriented_path.path_element ;
    the_reverse.orientation := NOT(a_path\oriented_path.orientation);
  ELSE
    the_reverse.path_element := a_path ;
    the_reverse.orientation := FALSE ;
  END_IF;

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

Remove the output description 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 does not give a correct initialisation of the **the_reverse.name** *attribute. Additional clarification to values of the attribute are also given. 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;
  the_reverse.name := '' ;
  IF ('TOPOLOGY_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF (a_face_bound) ) THEN
    the_reverse.bound := a_face_bound\face_bound.bound ;
   the_reverse.orientation := NOT (a_face_bound\face_bound.orientation);
  ELSE
    the_reverse.bound := a_face_bound.bound ;
    the_reverse.orientation := NOT (a_face_bound.orientation);
  END_IF;
 RETURN (the_reverse);
END_FUNCTION;
(*
```

Clause 5.5.6, p. 156

The EXPRESS specification does not give a correct initialisation of the **the_reverse.name** *attribute. 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 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 specification and replace with the following:

EXPRESS specification:

```
*)
FUNCTION face_reversed (a_face : face) : oriented_face;
  LOCAL
    the_reverse : oriented_face;
  END_LOCAL;
  the_reverse.name := '' ;
  the_reverse.bounds := set_of_topology_reversed(a_face.bounds);
  IF ('TOPOLOGY_SCHEMA.ORIENTED_FACE' IN TYPEOF (a_face) ) THEN
    the_reverse.face_element := a_face\oriented_face.face_element ;
    the_reverse.orientation := NOT (a_face\oriented_face.orientation) ;
  ELSE
    the_reverse.face_element := a_face ;
    the_reverse.orientation :=  FALSE ;
  END_IF;
     RETURN (the_reverse);
END_FUNCTION;
(*
```

Remove the output description 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 does not give a correct initialisation of the **the_reverse** *variable and intermediate values of the variable.*

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;
  LOCAL
    the_reverse : shell;
  END_LOCAL;

  IF ('TOPOLOGY_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
    the_reverse := representation_item ('') ||
                   topological_representation_item () ||
                   connected_face_set (set_of_topology_reversed (
                   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
    IF ('TOPOLOGY_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
      the_reverse := representation_item ('') ||
                   topological_representation_item () ||
                   connected_face_set (set_of_topology_reversed (
                   a_shell\connected_face_set.cfs_faces)) ||
                   open_shell () ||  oriented_open_shell (a_shell, FALSE);
    ELSE
      IF ('TOPOLOGY_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF (a_shell)) THEN
        the_reverse := representation_item ('') ||
                   topological_representation_item () ||
                   connected_face_set (set_of_topology_reversed (
                   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
        IF ('TOPOLOGY_SCHEMA.CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
          the_reverse := representation_item ('') ||
                topological_representation_item () ||
                connected_face_set (set_of_topology_reversed (
                a_shell\connected_face_set.cfs_faces)) ||
                 closed_shell () || oriented_closed_shell (a_shell, FALSE);
        ELSE
          RETURN (?);
        END_IF;
      END_IF;
    END_IF;
 END_IF;
 RETURN (the_reverse);
```