INTERNATIONAL STANDARD

ISO 10303-512

First edition
1999-12-15

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

Part 512:
Application interpreted construct: Faceted boundary representation

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

Partie 512: Construction interprétée d'application: Représentation délimitée des faces

Reference number
ISO 10303-512:1999(E)

PDF disclaimer

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.

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.

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

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

This part of ISO 10303 is a member of the application interpreted constructs series.

Annexes A and B form an integral part of this part of ISO 10303. Annexes C, D and E 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, generic 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 construct 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 definition of a boundary representation solid with planar faces and implicit topology. This is the final draft of a 500 series part edition of this AIC.

Industrial automation systems and integration — Product data representation and exchange — Part 512 : Application interpreted construct: Faceted boundary representation

1 Scope

This part of ISO 10303 specifies the interpretation of the generic resources in order to satisfy the following requirements:

— for the description of a three dimensional shape by means of a boundary representation model with planar faces and implicit straight line edges;

— for the composition of one or more such shapes as a **faceted_brep_shape_representation**.

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

— 3D geometry;

— B-reps;

— B-rep models;

— faceted B-reps;

— polyloops;

— unbounded geometry;

— use of topology to bound geometric entities;

— geometric transformations.

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

— 2D geometry;

— curves;

— explicit edge definitions;

— surfaces other than planes;

— offset curves and surfaces.

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: 1995, *Information technology - Open systems interconnection -Abstract syntax notation one (ASN.1) Part 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 TR 10303-12: 1997, *Industrial automation systems and integration - Product data representation and exchange - Part 12 : Description methods: The* EXPRESS-I *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: 1995, *Industrial automation systems and integration - Product data representation and exchange - Part 202: Application protocol: Associative draughting*

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.

— application;

— application context;

— application protocol;

— implementation method;

— integrated resource;

— interpretation;

— product data;

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.

— arcwise connected;

— boundary;

— boundary representation solid model (B-rep);

— bounds;

— coordinate space;

— curve;

— orientable;

— surface;

— topological sense.

3.3 Terms defined in ISO 10303-202

For the purposes of this part of ISO 10303, the following term defined in ISO 10303-202 applies.

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

3.4 Other definitions

3.4.1
faceted B-rep shape representation
a shape representation made up of one or more manifold faceted B-reps. Each constituent B-rep is required to have planar faces and implicitly defined edges.

3.4.2
manifold faceted B-rep
an arcwise connected faceted solid such that, for a very small sphere, centred at any point on the boundary of the solid, the interior of the sphere is divided into precisely two regions. One of these regions is inside the solid, the other is outside.

3.4.3
polyloop
a loop on a planar face consisting of linear segments. The edge geometry and topology is implicitly defined by a list of cartesian points.

3.5 Abbreviations

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

AIC	Application Interpreted Construct
AP	Application Protocol
B-rep	Boundary representation solid model

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 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 and topological entities for the definition of manifold solid models with planar faces and implicitly defined edges and vertices. The faces of the B-rep models are bounded by **poly_loop**s and each face is required to have an explicit outer bound.

The highest level entity in this AIC is the **faceted_brep_shape_representation.** A **faceted_brep_shape_representation** is a **shape_representation** (see: ISO 10303-41) consisting of **faceted_brep**s and **mapped_item**s defined as transformed copies of **faceted_brep**s.

EXPRESS specification:

```
*)
SCHEMA aic_faceted_brep;

 USE FROM geometry_schema
     (axis2_placement_3d,
      cartesian_point,
      cartesian_transformation_operator_3d,
      elementary_surface,
      plane);

 USE FROM geometric_model_schema
     (brep_with_voids,
      faceted_brep,
      manifold_solid_brep);

 REFERENCE FROM geometric_model_schema(msb_shells);

 USE FROM topology_schema
     (closed_shell,
      connected_face_set,
      face_bound,
      face_outer_bound,
      face_surface,
      oriented_closed_shell,
      poly_loop);

 USE FROM representation_schema(mapped_item);

 USE FROM product_property_representation_schema(shape_representation);

(*
```

NOTE 1 The **connected_face_set** entity is explicitly interfaced (i.e. included in the USE FROM lists) to allow rules in the **faceted_brep_shape_representation** entity to access attributes of this entity. For the use of this AIC this entity shall only be instantiated as one of its subtypes.

NOTE 2 The entity **manifold_solid_brep** is explicitly interfaced to enable compilation of the function **msb_shells**. The entity **elementary_surface** is explicitly interfaced to allow **faceted_brep_shape_representation** to access attributes of this entity. For the use of this AIC, these entities shall only be instantiated as one of their subtypes.

NOTE 3 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
topology_schema	ISO 10303-42
representation_schema	ISO 10303-43
product_property_representation_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:

— axis2_placement_3d;

— brep_with_voids;

— cartesian_point;

— cartesian_transformation_operator_3d;

— closed_shell;

— direction;

— face_bound;

— face_outer_bound;

— face_surface;

— faceted_brep;

— mapped_item;

— oriented_closed_shell;

— plane;

— poly_loop;

— representation_map.

An application protocol that uses this AIC shall require that a **shape_representation** entity is instantiated as a **faceted_brep_shape_representation**.

4.2 aic_faceted_brep schema entity definition: faceted_brep_shape_representation

The **faceted_brep_shape_representation** is a type of shape representation in which the representation items are specialisations of faceted brep entities. These differ from the more general B-rep in having only planar faces and implicit edge geometry.

EXPRESS specification:

```
*)
ENTITY faceted_brep_shape_representation
  SUBTYPE OF (shape_representation);
WHERE
  WR1 : SIZEOF (QUERY (it <* items |
         NOT (SIZEOF(['AIC_FACETED_BREP.FACETED_BREP',
            'AIC_FACETED_BREP.MAPPED_ITEM',
            'AIC_FACETED_BREP.AXIS2_PLACEMENT_3D'] *
             TYPEOF(it)) = 1))) = 0;
  WR2 : SIZEOF (QUERY (it <* items |
         SIZEOF(['AIC_FACETED_BREP.FACETED_BREP',
         'AIC_FACETED_BREP.MAPPED_ITEM'] * TYPEOF(it)) = 1)) > 0;
  WR3 : SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
         'AIC_FACETED_BREP.FACETED_BREP' IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep) |
         NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
         NOT (('AIC_FACETED_BREP.FACE_SURFACE' IN TYPEOF (fcs)) AND
         (('AIC_FACETED_BREP.PLANE' IN TYPEOF
              (fcs\face_surface.face_geometry)) AND
         ('AIC_FACETED_BREP.CARTESIAN_POINT' IN TYPEOF (
              fcs\face_surface.face_geometry\
                   elementary_surface.position.location))))))
                     = 0))) = 0))) = 0;
  WR4 : SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
         'AIC_FACETED_BREP.FACETED_BREP' IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep) |
         NOT (SIZEOF (QUERY (fcs <* csh\connected_face_set.cfs_faces |
         NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
          'AIC_FACETED_BREP.FACE_OUTER_BOUND' IN TYPEOF(bnds)))
            = 1))) = 0))) = 0))) = 0;
  WR5 : SIZEOF (QUERY (msb <* QUERY (it <* items |
         'AIC_FACETED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
         'AIC_FACETED_BREP.ORIENTED_CLOSED_SHELL' IN
            TYPEOF (msb\manifold_solid_brep.outer))) = 0;
  WR6 : SIZEOF (QUERY (brv <* QUERY (it <* items |
         'AIC_FACETED_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids.voids |
          csh\oriented_closed_shell.orientation)) = 0))) = 0;
  WR7 : SIZEOF (QUERY (mi <*  QUERY (it <* items |
              'AIC_FACETED_BREP.MAPPED_ITEM' IN TYPEOF(it)) |
```

```
          NOT ('AIC_FACETED_BREP.FACETED_BREP_SHAPE_REPRESENTATION' IN
            TYPEOF(mi\mapped_item.mapping_source.mapped_representation))))
                = 0;
 END_ENTITY;
(*
```

Formal propositions:

WR1: The **items** attribute of the **representation** supertype shall contain **faceted_brep**s, **mapped_item**s and **axis2_placement_3d**s only.

WR2: At least one item in the **items** set shall be a **faceted_brep** entity, or a **mapped_item**, (See also WR7).

WR3: For each **faceted_brep** in the **items** set: the faces shall be **face_surface**s, the associated surface for each face shall be a **plane**, and each **plane** shall use a **cartesian_point** for its location.

NOTE 1 The call to function **msb_shells** in WR3 and WR4 is correct since, although the generic type of the argument 'fbrep' is **representation_item**, 'fbrep' has been selected by QUERY to be a subtype of **manifold_solid_brep**.

WR4: An explicit outer bound shall be specified for each **face** of the **faceted_brep**.

WR5: For each **manifold_solid_brep** in the **items** set the **outer** shell attribute shall not be of the oriented subtype.

WR6: If the **manifold_solid_brep** is also a **brep_with_voids** then each shell in the **voids** set shall be an **oriented_closed_shell** with orientation value FALSE.

WR7: If a **mapped_item** is included in the **items** set then the **mapped_representation** of the **mapping_source** attribute shall be a **faceted_brep_shape_representation**.

NOTE 2 If a **cartesian_transformation_operator_3d** is included as **mapped_item.mapping_target** with an **axis2_placement_3d** that corresponds to the original coordinate system as **mapped_representation.mapping_origin**, then the resulting **mapped_item** is a transformed copy of the **faceted_brep_shape_representation**. The precise definition of the transformation, including translation, rotation, scaling and, if appropriate, mirroring, is given by the transformation operator.

EXPRESS specification:

```
 *)
 END_SCHEMA; -- end AIC FACETED BREP SCHEMA
 (*
```

Annex A
(normative)

Short names of entities

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

Table A.1 – Short names of entities

Entity name	Short name
FACETED_BREP_SHAPE_REPRESENTATION	FBSR

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(512) 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_faceted_brep in an open information system, the object identifier

{ iso standard 10303 part(512) version(1) object(1) aic-faceted-brep(1) }

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

Annex C
(informative)

Computer-interpretable listings

This annex provides a listing of the EXPRESS entity names and corresponding short names as specified in this Part of ISO 10303 without comments or other explanatory text. This annex is available in computer-interpretable form and can be found at the following URLs:

Short names: http://www.mel.nist.gov/div826/subject/apde/snr/
EXPRESS: http://www.mel.nist.gov/step/parts/part512/

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC 184/SC4 Secretariat directly at: sc4sec@cme.nist.gov.

NOTE – The information provided in computer-interpretable form at the above URLs is informative. The information that is contained in the body of this part of ISO 10303 is normative.

Annex D
(informative)

EXPRESS-G diagrams

Figures D.1 through D.2 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 1 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 used by other entities in this part of ISO 10303.

— geometric_set_select;

— pcurve_or_surface;

— shell;

— trimming_select;

— vector_or_direction.

NOTE 2 The implicit interface rules of ISO 10303-11 also introduce some entities whose instantiation is prohibited by rules on the **faceted_brep_shape_representation**. These entities are marked " * " in the EXPRESS-G diagrams.

NOTE 3 To avoid congestion in the figures some derived attributes are omitted in Figures D.1 and D.2.

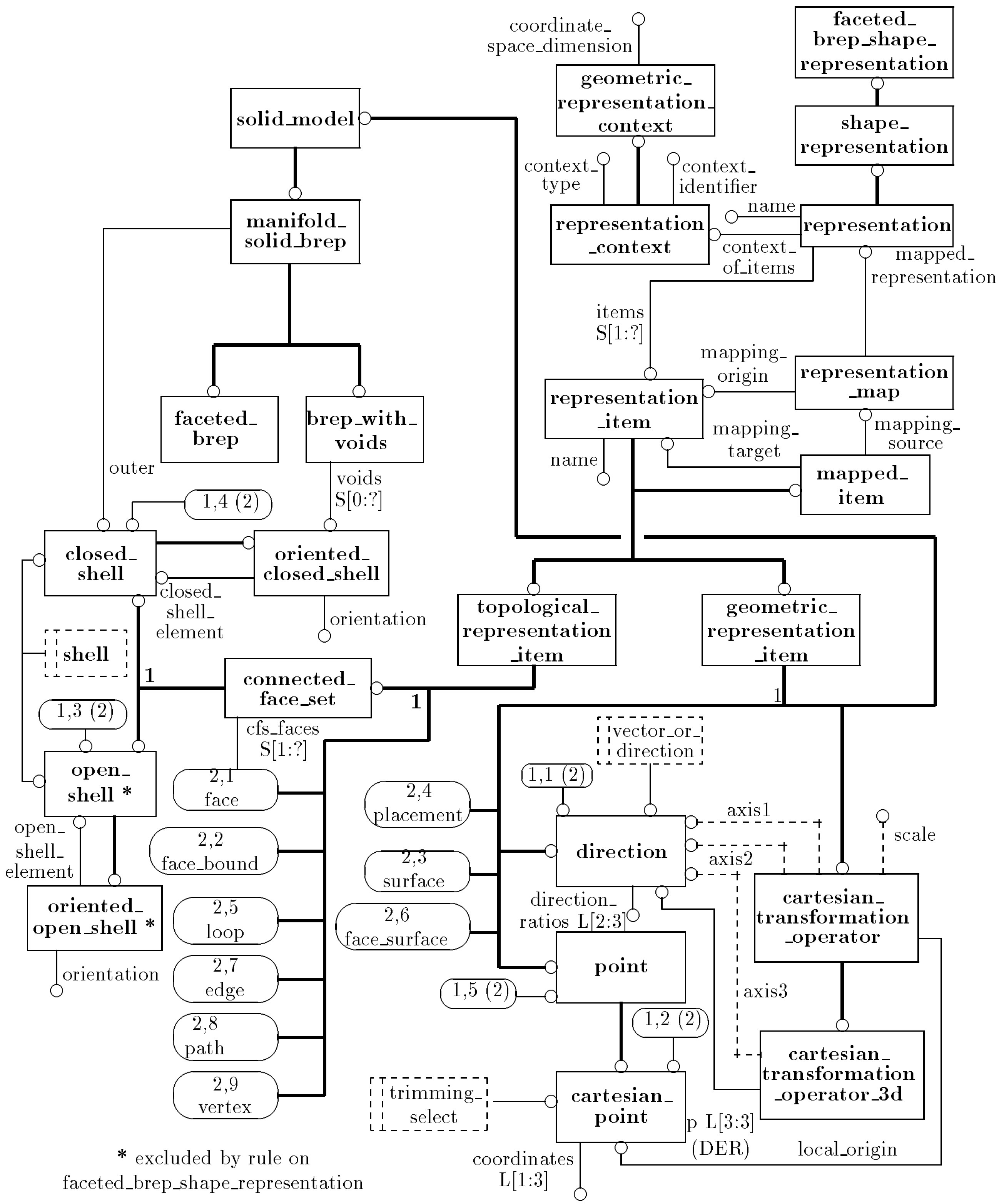

Figure D.1 – aic_faceted_brep, EXPRESS-G diagram 1 of 2

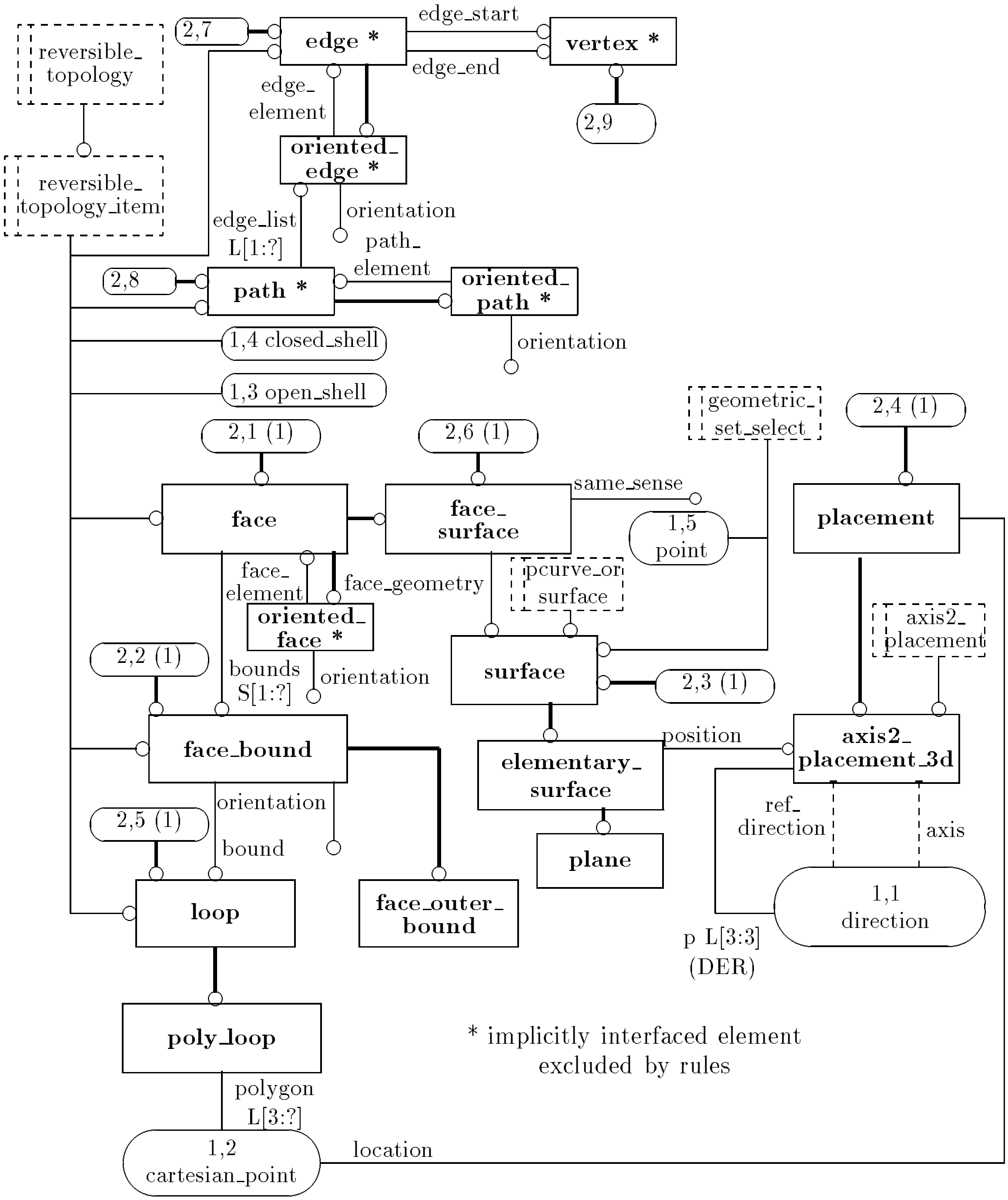

Figure D.2 – aic_faceted_brep, EXPRESS-G diagram 2 of 2

Annex E
(informative)

AIC conformance requirements and test purposes

E.1 AIC conformance requirements: faceted B-rep

Any application protocol that uses this AIC may require conformance to the AIC conformance requirements defined below when instantiating a **faceted_brep_shape_representation**.

Conformance to this AIC means that all the defined types and entity types defined in the AIC EXPRESS listing form are supported. The only legitimate use, within the context of this AIC, for a geometric or topological entity instance is for the purpose of defining a **faceted_brep_shape_representation**.

The following entities are instantiable as part of the definition of a **faceted_brep_shape_representation**:

— axis2_placement_3d;

— brep_with_voids;

— cartesian_point;

— cartesian_transformation_operator_3d;

— closed_shell;

— direction;

— face_bound;

— face_outer_bound;

— face_surface;

— faceted_brep;

— mapped_item;

— oriented_closed_shell;

— plane;

— poly_loop;

— representation_map.

E.2 Test purposes for faceted B-rep AIC

This section defines conformance test purposes which are appropriate for the faceted B-rep AIC. The criteria are based on the constructs found in clause 4 of this part of ISO 10303.

NOTE For any AP using this AIC further test purposes may be required for the 'top-level' entity which is a type of **shape_representation**. The AP will also require to establish a product context for these test purposes.

E.2.1 faceted_brep_shape_representation

The following test purposes are derived from the definition of this entity:

FB1: representation as **shape_representation** as **faceted_brep_shape_representation**. (see E.3.1)

FB2: faceted_brep_shape_representation with context as **geometric_context** with **items** as **faceted_brep**. (see E.3.1)

FB3: faceted_brep_shape_representation with context as **geometric_context** with **items** as **mapped_item**. (see E.3.4)

FB4: faceted_brep_shape_representation with context as **geometric_context** with **items** as two or more items as **faceted_brep**, or **mapped_item**, or **axis2_placement_3d**, including at least one **axis2_placement_3d**. (see E.3.4)

E.2.2 faceted_brep

The following test purposes are derived from the definition of this entity:

FB5: faceted_brep with **outer** (voids absent) as **closed_shell**. (NOT **oriented_closed_shell** subtype.) (see E.3.1)

FB6: faceted_brep and **brep_with_voids** subtype with outer as **closed_shell** and voids as a SET of one **oriented_closed_shell**. (voids present) (see E.3.2)

FB7: faceted_brep and **brep_with_voids** subtype with outer as **closed_shell** and voids as a SET of more than one **oriented_closed_shell**. (voids present) (see E.3.2)

E.2.3 oriented_closed_shell

The following test purpose is derived from the definition of this entity and the constraints imposed on the **faceted_brep_shape_representation**:

FB8: oriented_closed_shell with **orientation** = FALSE. (see E.3.2)

E.2.4 closed_shell

The following test purpose is derived from the definition of this entity and the constraints imposed on the **faceted_brep_shape_representation**:

FB9: closed_shell with **cfs_faces** as a SET of more than one **face_surface**. (see E.3.1)

E.2.5 face_surface

The following test purposes are derived from the definition of this entity and the constraints imposed on the **faceted_brep_shape_representation**:

FB10: face_surface with **face_geometry** as **plane**. (see E.3.1)

FB11: face_surface with **same_sense** = TRUE. (see E.3.1)

FB12: face_surface with **same_sense** = FALSE. (see E.3.3)

E.2.6 face

The following test purposes are derived from the definition of this entity and the constraints imposed on the **faceted_brep_shape_representation**:

FB13: face as **face_surface** with **bounds** as SET of one **face_bound** as **face_outer_bound** with **orientation** = TRUE. (see E.3.1)

FB14: face as **face_surface** with **bounds** as SET of one **face_bound** as **face_outer_bound** with **bound** as **poly_loop** and **orientation** = FALSE. (see E.3.3)

FB15: face as **face_surface** with **bounds** as SET of at least two **face_bound**s (including one **face_outer_bound**) with **bound** as **poly_loop** and **orientation** = TRUE. (see E.3.3)

FB16: face as **face_surface** with **bounds** as SET of at least two **face_bound**s (including one **face_outer_bound**) with **bound** as **poly_loop** and **orientation** = FALSE. (see E.3.3)

E.2.7 surface

The following criteria are derived from the definition of this entity and the constraints imposed on the **faceted_brep_shape_representation**:

FB17: surface (as **elementary_surface**) as **plane** with **position** as **axis2_placement_3d** with **axis** present. (see E.3.1)

FB18: surface (as **elementary_surface**) as **plane** with **position** as **axis2_placement_-3d** with **axis** absent. (see E.3.3)

FB19: surface (as **elementary_surface**) as **plane** with **position** as **axis2_placement_-3d** with **ref_direction** present. (see E.3.1)

FB20: surface (as **elementary_surface**) as **plane** with **position** as **axis2_placement_-3d** with **ref_direction** absent. (see E.3.3)

E.2.8 cartesian_transformation_operator_3d

The following criteria are derived from the definitions of this entity, the **mapped_item** entity, and the constraints imposed on the **faceted_brep_shape_representation**:

FB21: mapped_item with **mapping_target** as **cartesian_transformation_operator_-3d**. (see E.3.5)

FB22: cartesian_transformation_operator as **cartesian_transformation_operator_-3d** with scale as REAL not equal to 1.0. (see E.3.5)

E.3 Test cases for faceted B-rep AIC

E.3.1 Test case fb1

Test case fb1 is the most basic test case consisting of a single solid tetrahedron with one vertex at the origin of the coordinate system. All geometry is explicitly defined with no defaults and no sense reversals required. The definition of the outer shell is provided by the **tetrashell_instance** context using the original parameters.

E.3.1.1 AIM test purposes covered

FB1: representation as shape_representation as faceted_brep_shape_representation;

FB2: faceted_brep_shape_representation with context as geometric_representation_context with items as faceted_brep;

FB5: faceted_brep with outer (voids absent) as closed_shell (NOT oriented_closed_shell subtype);

FB9: closed_shell with cfs_faces as a SET of more than one face_surface

FB10: face_surface with face_geometry as plane;

FB11: face_surface with same_sense = TRUE;

FB13: face as face_surface with bounds as SET of one face_bound as face_outer_bound with orientation = TRUE;

FB17: surface (as elementary_surface) as plane with position as axis2_placement_3d with axis present;

FB19: surface (as elementary_surface) as plane with position as axis2_placement_3d with ref_direction present.

E.3.1.2 Postprocessor input specification

The test case is a **faceted_brep_shape_representation** consisting of a single **faceted_brep**. The **faceted_brep** is in the form of a solid tetrahedron with one vertex at the origin and the adjacent edges along the coordinate axes. All faces are defined by **poly_loop**s and are of type **face_surface** with the surface geometry defined by a plane. A suitable set of dimensions is defined in the EXPRESS-I specification below that gives the complete specification.

E.3.1.2.1 EXPRESS-I specification: fbrep_1

```
*)
TEST_CASE example_fbrep_1; WITH aic_faceted_brep;

  REALIZATION

  LOCAL
    shell_object : closed_shell ;
    tetrahedron  : faceted_brep ;
    fbsr1 : faceted_brep_shape_representation ;
    its_units : named_unit ;
    prod1_context : representation_context ;
  END_LOCAL;

    CALL tetrashell_instance ; -- uses default values, so no WITH
      IMPORT (shell_object := @tetrashell; ) ;
    END_CALL;

      its_units := length_unit() || si_unit ('milli', 'metre') ;

      prod1_context := geometric_representation_context
                      ('context_1', 'context_for_tetrahedron', 3) ||
                      global_unit_assigned_context ( [its_units] ) ;

      tetrahedron := faceted_brep ('tetrahedron', shell_object) ;

      fbsr1 := faceted_brep_shape_representation
                ('fbsr1',  [tetrahedron], its_context );
```

```
  END_REALIZATION;
END_TEST_CASE;
(*
```

NOTE 1 This test case uses the most simple form of tetrashell_instance context with default values to define shape.

NOTE 2 Global length units assigned as mm in global_units_assigned_context.

E.3.1.3 Postprocessor verdict criteria

FB1: All WRs on **faceted_brep_shape_representation** shall be verified.

FB2: model created shall contain no vertices,
model shall contain no edges,
model shall contain no curves,
length units correctly interpreted,
global_unit_assigned_context correctly interpreted.

FB5: **faceted_brep** entity shall be correctly interpreted.

FB9: **cfs_faces** attribute shall be a SET of more than one face_surface these faces shall intersect on lines joining points.

FB10: **plane** defining **face_geometry** for each **face_surface** shall pass through all points defining bounding **poly_loop**.

FB11: Normal to each **plane** shall point out of solid.

FB19: Set **P** of 3 axes for each **axis2_placement_3d** shall be correctly derived.

E.3.2 Test case fb2

Test case fb2 is designed to test the definition of a faceted B-rep containing one or more voids. The **tetrashell_instance** context is used with different parameters to define the outer shell and the void shells. The result is a hollow tetrahedral solid with void(s) of a similar shape.

NOTE If required this test can easily be modified to test geometric precision by varying the parameters to define voids that are very close to each other or to the outer shell. As defined in the current version of this test case there should be no possibility of such interference.

E.3.2.1 AIM test purposes covered

FB6: faceted_brep and brep_with_voids subtype with outer as closed_shell and voids as a SET of one oriented_closed_shell (voids present);

FB7: faceted_brep and brep_with_voids subtype with outer as closed_shell and voids as a SET of more than one oriented_closed_shell;

FB8: oriented_closed_shell with orientation as FALSE.

E.3.2.2 Postprocessor input specification

The test model is a **faceted_brep_shape_representation** consisting of a single **faceted_brep**. The **faceted_brep** is in the form of a hollow tetrahedron with one vertex at the origin and the adjacent edges along the coordinate axes. Two separate instances shall be created, one with a single inner void, the other with 2 voids. Each void shell is of a similar shape to the outer shell and located inside the body. The voids shall not intersect each other or the outer shell. A suitable set of dimensions is defined in the EXPRESS-I specification below that gives the complete specification.

E.3.2.2.1 EXPRESS-I specification: fbrep_2

```
*)
TEST_CASE example_fbrep_2; WITH faceted_brep_aic;

  REALIZATION

  LOCAL
    shell_object, hollow1, hollow2 : closed_shell ;
    void1, void2 : oriented_closed_shell;
    tetra_with_void : manifold_solid_brep ;
    tetra_with_voids : manifold_solid_brep ;
    fbsr1, fbsr2 : faceted_brep_shape_representation ;
    its_context : representation_context ;
    its_units : named_unit ;
  END_LOCAL;

    CALL tetrashell_instance ; -- uses default values, so no WITH
      IMPORT (shell_object := @tetrashell; );
    END_CALL;

    CALL tetrashell_instance ; -- parameters re-set for dimensions (large void)
      IMPORT (hollow1 := @tetrashell; );
      WITH (orc := 20; lx := 50; ly := 50; lz := 50;);
    END_CALL;

    CALL tetrashell_instance ; -- parameters re-set for dimensions (small void)
      IMPORT (hollow2 := @tetrashell; );
      WITH (orc := 5; lx := 20; ly := 20; lz := 20;);
    END_CALL;
```

```
    CALL basic_product_structure ;  -- parameters for second product.
      IMPORT (shape_2_def := @prod_def_shape; );
      WITH   (prod_name := @prod2_name;
              pdef_desc := 'test product definition 2';
              propd_desc := 'shape of test product 2';
              prod_name :=  'second test product';
              prod_id   := 'P02' ;
              pdf_id    := 'PDF02' ; );
    END_CALL;

    void1 := oriented_closed_shell ('void1', hollow1, FALSE) ;
    void2 := oriented_closed_shell ('void2', hollow2, FALSE) ;

    tetra_with_void := faceted_brep ('tetra_with_void', shell_object) ||
                       brep_with_voids ([void1]) ;

    tetra_with_voids := faceted_brep ('tetra_with_voids', shell_object) ||
                        brep_with_voids ([void1, void2]) ;

    its_units := length_unit() || si_unit ('milli', 'metre') ;

    its_context := geometric_representation_context
                   ('context_1', 'context_for_tetrahedron', 3) ||
                   global_unit_assigned_context ( [its_units] ) ;


    fbsr1 := faceted_brep_shape_representation
             ( 'fbsr1', [tetra_with_void], its_context ) ;

    fbsr2 := faceted_brep_shape_representation
             ('fbsr2',  [tetra_with_voids], its_context ) ;


  END_REALIZATION;
END_TEST_CASE;
(*
```

NOTE 1 All criteria for fb1, except those relating to FB5, also apply.

NOTE 2 Test case uses oriented_closed_shell to define voids, orientation must be FALSE.

NOTE 3 Context is re-used with different parameters to define void shells.

E.3.2.3 Postprocessor verdict criteria

FB6: Complex subtype faceted_brep and brep_with_voids shall be correctly interpreted.

FB7: faceted_brep with more than one void correctly interpreted, void shells shall not intersect each other, or outer.

FB8: Normal to each void shell shall point into void.

E.3.3 Test case fb3

Test case fb3 is designed to test the abilities to define faces with inner loops and also to test the use of defaults and sense reversals. The test object is in the form of a rectangular block with a triangular through hole and a triangular depression in the top face. All geometry and topology is defined in the text case.

E.3.3.1 AIM test purposes covered

FB12: face_surface with same_sense = FALSE;

FB14: face as face_surface with bounds as SET of one face_bound as face_outer_bound with bound as poly_loop and orientation = FALSE;

FB15: face as face_surface with bounds as SET of at least two face_bounds (including one face_outer_bound) with bound as poly_loop and orientation = TRUE;

FB16: face as face_surface with bounds as SET of at least two face_bounds (including one face_outer_bound) with bound as poly_loop and orientation = FALSE;

FB18: surface (as elementary_surface) as plane with position as axis2_placement_3d with axis absent;

FB20: surface (as elementary_surface) as plane with position as axis2_placement_3d with ref_direction absent.

E.3.3.2 Postprocessor input specification

. The model is a **faceted_brep_shape_representation** consisting of a single **faceted_brep**. The **faceted_brep** is in the form of a rectangular block with edges parallel to the coordinate axes. The block has a triangular through hole and a triangular depression in one face. Some loops are created with orientation false. For some faces the sense of the **face_geometry** is false. Default values are used for some attributes of **axis2_placement_3d** when creating the exchange model. A full specification is provided in the EXPRESS-I definition.

E.3.3.2.1 EXPRESS-I specification: fbrep_3

```
*)
TEST_CASE example_fbrep_3; WITH faceted_brep_aic;
```

```
REALIZATION

LOCAL
  origin, px, py, pxy, pz, pxz, pyz, pzyz   : cartesian_point;
  q, qx, qy, qz, qxz, qyz                   : cartesian_point;
  r, rx, ry, rz, rxz, ryz                   : cartesian_point;
  neg_x, neg_y, neg_z, slope, pos_x, pos_y  : direction;
  loopb, loopt, loopf, loopbk, loopl, loopr : poly_loop;
  loopbi, loopmid, loopt1, loopt2, loopqf   : poly_loop;
  loopql, loopqs, looprb, looprr, looprs    : poly_loop;
  a1, a2, a3, ap3, ar3, ap1, ap2, ar1, ar2  : axis2_placement_3d;
  ar4, aq1, aq2, aq4                        : axis2_placement_3d;
  pa1, pa2, pa3, pap1, pap2, pap3, paq1     : plane;
  paq2, paq4, par1, par2, par3, par4        : plane;
  bottom, top, front, back, left, right     : face_outer_bound;
  qfbd, qlbd, qsbd, rbbd, rrbd, rsbd, midbd : face_outer_bound;
  bibd, t1bd, t2bd                          : face_bound;
  fs1, fs2, fs3, fsp1, fsp2, fsp3, fsq1     : face_surface;
  fsq2, fsq4, fsr1, fsr2, fsr3, fsr4        : face_surface;
  blockshell                                : closed_shell ;
  block                                     : faceted_brep ;
  fbsr                              : faceted_brep_shape_representation ;
  its_context : representation_context ;
  its_units : named_unit ;
END_LOCAL;

(* Cartesian_points on boundary: *)
origin := cartesian_point ('origin', [0, 0, 0]) ;
px := cartesian_point ('px', [50, 0, 0]) ;
py := cartesian_point ('py', [0, 50, 0]) ;
pxy := cartesian_point ('pxy', [50, 50, 0]) ;
pz := cartesian_point ('pz', [0, 0, 100]) ;
pxz := cartesian_point ('pxz', [50, 0, 100]) ;
pyz := cartesian_point ('pyz', [0, 50, 100]) ;
pxyz := cartesian_point ('pxyz', [0, 0, 100]) ;

q   := cartesian_point ('q', [10,10, 0]) ;
qx := cartesian_point ('qx', [25,10, 0]) ;
qy := cartesian_point ('qy', [10, 25, 0]) ;
qz := cartesian_point ('qz', [10, 10, 100]) ;
qxz := cartesian_point ('qxz', [25, 10, 100]) ;
qyz := cartesian_point ('qyz', [10, 25, 100]) ;

r   := cartesian_point ('r', [45, 45, 50]) ;
rx := cartesian_point ('rx', [35, 45, 50]) ;
ry := cartesian_point ('ry', [45, 35, 50]) ;
```

```
  rz := cartesian_point ('rz', [45, 45, 100]) ;
  rxz := cartesian_point ('rxz', [35, 45, 100]) ;
  ryz := cartesian_point ('ryz', [45, 35, 100]) ;

  neg_x := direction ('neg_x', [-1, 0, 0]) ;
  neg_y := direction ('neg_y', [0, -1, 0]) ;
  neg_z := direction ('neg_z', [0, 0, -1]) ;
  slope := direction ('slope', [1, 1, 0 ]) ;
  pos_x := direction ('pos_x', [1,  0, 0]) ;
  pos_y := direction ('pos_y',[0, 1, 0 ]) ;

(*  outer loops:      *)
  loopb := poly_loop ('loopb', [origin, px, pxy, py]) ;
  loopt := poly_loop ('loopt',[origin, pxz, pxyz, pyz]) ;
  loopf := poly_loop ('loopf', [origin, px, pxz, pz]) ;
  loopbk := poly_loop ('loopbk',[py, pxy, pxyz, pyz]) ;
  loopl  := poly_loop ('loopl ', [origin, pz, pyz, py]) ;
  loopr  := poly_loop ('loopr ', [px, pxz, pxyz, pxy]) ;

(* inner loops (triangular):    *)
  loopbi  := poly_loop ('loopbi ', [q, qx, qy]) ;
  loopmid := poly_loop ('loopmid', [r, rx, ry]) ;
  loopt1  := poly_loop ('loopt1', [qz, qxz, qyz]) ;
  loopt2  := poly_loop ('loopt2', [rz, rxz, ryz]) ;

(*   inside loops (rectangular):     *)
  loopqf  :=  poly_loop ('loopqf', [q, qx, qxz, qz]) ;
  loopql  :=  poly_loop ('loopql', [q, qz, qyz, qy]) ;
  loopqs  := poly_loop ('loopqs', [qx, qxz, qyz, qy]) ;
  looprb  :=  poly_loop ('looprb', [r, rx, rxz, rz]) ;
  looprr  :=  poly_loop ('looprr', [r, rz, ryz, ry]) ;
  looprs  := poly_loop ('looprs', [rx, rxz, ryz, ry]) ;

(* axis_placements {note: number determines if axis parallel to x, y or z}  *)
   a1 := axis2_placement_3d ('a1', origin, neg_x, neg_y) ;
   a2 := axis2_placement_3d ('a2', origin, neg_y, neg_x) ;
   a3 := axis2_placement_3d ('a3', origin, neg_z, ?) ;
   ap3 := axis2_placement_3d ('ap3', pz,  ?, ?) ;
   ar3 := axis2_placement_3d ('ar3', r,  ?,  ?) ;
   ap1 := axis2_placement_3d ('ap1', px, pos_x, ?) ;
   ap2 := axis2_placement_3d ('ap2', py, pos_y, ?) ;
   ar1 := axis2_placement_3d ('ar1', r, pos_x, ?) ;
   ar2 := axis2_placement_3d ('ar2', r, pos_y, ?) ;
   ar4 := axis2_placement_3d ('ar4', r, slope, ?) ;
   aq1 := axis2_placement_3d ('aq1', q, pos_x, ?) ;
   aq2 := axis2_placement_3d ('aq2', q, pos_y, ?) ;
```

```
   aq4 := axis2_placement_3d ('aq4', r, slope, ?) ;

(* plane defined for each axis placement: *)
  pa1 := plane ('pa1', a1) ;
  pa2 := plane ('pa2', a2) ;
  pa3 := plane ('pa3', a3) ;
  pap1 := plane ('pap1', ap1) ;
  pap2 := plane ('pap2', ap2) ;
  pap3 := plane ('pap3', ap3) ;
  paq1 := plane ('paq1', aq1) ;
  paq2 := plane ('paq2', aq2) ;
  paq4 := plane ('paq4', aq4) ;
  par1 := plane ('par1', ar1) ;
  par2 := plane ('par2', ar2) ;
  par3 := plane ('par3', ar3) ;
  par4 := plane ('par4', ar4) ;

(*   block outer bounds  *)
  bottom := face_outer_bound ('bottom', loopb,  FALSE) ;
  top    := face_outer_bound ('top', loopt,  TRUE) ;
  front  := face_outer_bound ('front', loopf,  TRUE) ;
  back   := face_outer_bound ('back', loopbk,  FALSE) ;
  left   := face_outer_bound ('left', loopl,  FALSE) ;
  right  := face_outer_bound ('right', loopr,  TRUE) ;

(*   inner face outer bounds  *)
  qfbd  := face_outer_bound ('qfbd', loopqf,  FALSE) ;
  qlbd   := face_outer_bound ('qlbd', loopql,  TRUE) ;
  qsbd  := face_outer_bound ('qsbd', loopqs,  FALSE) ;
  rbbd   := face_outer_bound ('rbbd', looprb,  FALSE) ;
  rrbd  := face_outer_bound ('rrbd', looprr,  TRUE) ;
  rsbd   := face_outer_bound ('rsbd', looprs,  TRUE) ;
  midbd := face_outer_bound ('midbd', loopmid,  TRUE) ;

(*   inner bounds    *)
  bibd  := face_bound ('bibd', loopbd,  TRUE) ;
  t1bd  := face_bound ('t1bd', loopt1,  FALSE) ;
  t2bd  := face_bound ('t2bd', loopt2,  FALSE) ;

(*  outer faces of block *)
  fs1 := face_surface ('fs1', [left], pa1,  TRUE) ;
  fs2 := face_surface ('fs2', [front],  pa2,  TRUE) ;
  fs3 := face_surface ('fs3', [bottom, bibd],  pa3,  TRUE) ;
  fsp1 := face_surface ('fsp1', [right], pap1, TRUE) ;
  fsp2 := face_surface ('fsp2', [back], pap2, TRUE) ;
  fsp3 := face_surface ('fsp3', [top, t1bd, t2bd], pap3, TRUE) ;
```

```
(*   inner faces
      hole:    *)
  fsq1 := face_surface ('fsq1', [qlbd], paq1, TRUE) ;
  fsq2 := face_surface ('fsq2', [qfbd], paq2,  TRUE) ;
  fsq4 := face_surface ('fsq4', [qsbd], paq4,  FALSE) ;
(*  depression:  *)
  fsr1 := face_surface ('fsr1', [rrbd], par1,  FALSE) ;
  fsr2 := face_surface ('fsr2', [rbbd], par2,  FALSE) ;
  fsr3 := face_surface ('fsr3', [midbd], par3,  TRUE) ;
  fsr4 := face_surface ('fsr4', [rsbd], par4,  TRUE) ;

blockshell := closed_shell ('blockshell', [fs1, fs2, fs3, fsp1, fsp2, fsp3,
                             fsq1, fsq2, fsq4, fsr1, fsr2, fsr3, fsr4]) ;

block := faceted_brep ('block', blockshell) ;



    its_units := length_unit() || si_unit ('milli', 'metre') ;

    its_context := geometric_representation_context
                     ('its_context', 'context_for_block', 3) ||
                     global_unit_assigned_context ( [its_units] ) ;


      fbsr := faceted_brep_shape_representation
       ('fbsr',  [block], its_context ) ;


  END_REALIZATION;
END_TEST_CASE;
(*
```

NOTE 1 Inner loops are used to define holes and depressions.

NOTE 2 Sense reversal of loops and surface geometry is tested.

NOTE 3 Defaults used in defining axis2_placement_3d.

E.3.3.3 Postprocessor verdict criteria

FB12: face_surface with orientation FALSE shall be correctly processed, the **closed_shell** defined shall have all face normals pointing out of solid material.

FB14: face_bound with orientation FALSE shall be correctly processed, final orientation shall always be correct with face to left.

FB15: faces with inner bound shall be correctly processed, bounds shall be checked for intersections.

FB16: loop with orientation FALSE shall be correctly interpreted.

FB18: axis shall be correctly defaulted when processing **axis2_placement_3d**.

FB19: ref_direction shall be correctly defaulted when processing **axis2_placement_3d**.

E.3.4 Test case fb4

Test case fb4 is designed to test the use of **mapped_item**s in the creation of a simple assembly of faceted B-reps. It also provides a test of the consistent behaviour of **geometric_-representation_context**s in distinguishing between coordinate spaces. This test makes use of the **tetrashell_instance** context to define the geometry and topology.

E.3.4.1 AIM test purposes covered

FB3: faceted_brep_shape_representation with context as geometric_representation_context with items as mapped_item;

FB4: faceted_brep_shape_representation with context as geometric_representation_context with two or more items as faceted_brep, or mapped_item, or axis2_placement_3d, including at least one axis2_placement_3d.

E.3.4.2 Postprocessor input specification

The model is defined with a **faceted_brep_shape_representation** consisting of a single **faceted_brep**. The **faceted_brep** is in the form of a solid tetrahedron with one vertex at the origin and the adjacent edges along the coordinate axes. This representation is then used in conjunction with the **mapped_item** entity to create, in the same **representation_context**, a representation consisting of a rotated copy of the original representation. In a separate **representation_context** a representation is created consisting of the original **faceted_brep** and a mapped copy of the original representation. A full specification is provided in the EXPRESS-I definition.

E.3.4.2.1 EXPRESS-I specification: fbrep_4

```
  *)

TEST_CASE example_fbrep_4; WITH faceted_brep_aic;

  REALIZATION
```

```
  LOCAL
    origin : cartesian_point ;
    pos_z, neg_y  : direction ;
    refaxes, oldaxes, newaxes : axis_placement_3d;
    shell_object  : closed_shell ;
    tetrahedron : faceted_brep ;
    fbsr, fbsr1, fbsrass : faceted_brep_shape_representation ;
    grc1, grc2 : representation_context ;
    its_units : named_unit ;
    tetrarot1, tetrarot2  : mapped_item ;
    mapping1, mapping2   : representation_map ;
  END_LOCAL;



   CALL tetrashell_instance ; -- uses default values, so no WITH
      IMPORT (shell_object := @tetrashell;
              origin := @origin;
              neg_y := @ neg_y; refaxes := @a1;);
    END_CALL;


tetrahedron := faceted_brep ('tetrahedron', shell_object) ;
its_units := length_unit() || si_unit ('milli', 'metre') ;

grc1 := geometric_representation_context ('grc1',
    'context for tetrahedron', 3) ||
  global_unit_assigned_context ( [its_units] ) ;

grc2 := geometric_representation_context ('grc2',
      'context for rotated tetrahedron', 3) ||
      global_unit_assigned_context ( [its_units] ) ;

fbsr := faceted_brep_shape_representation ('fbsr', [tetrahedron], grc1 ) ;

(*  Define axis_placements for use in mapping *)
oldaxes := axis2_placement_3d ('oldaxes', origin, ?,  ?) ;

pos_z := direction ('pos_z', [0, 0, 1]) ;

newaxes := axis2_placement_3d ('newaxes', origin, pos_z, neg_y) ;

mapping1 := representation_map (refaxes, fbsr );
tetrarot1 := mapped_item ('tetrarot1', mapping1, newaxes );
```

```
(* Define representation using tetrarot1 only  *)
fbsr1 := faceted_brep_shape_representation ('fbsr1', [tetrarot1], grc1 ) ;

(* Define representation that is an assembly of tetrahedron + mapped copy.*)

  mapping2 := representation_map (oldaxes, fbsr );
  tetrarot2 := mapped_item ('tetrarot2', mapping2, newaxes );

  fbsrass := faceted_brep_shape_representation
          ('fbsrass', [tetrahedron, tetrarot2, oldaxes], grc2 ) ;

  END_REALIZATION;
END_TEST_CASE;
(*
```

NOTE 1 Test case of mapped_item and 'assembly' using simple solid tetrahedron.

NOTE 2 Tetrahedron shell is created by using context with default parameters.

NOTE 3 Outer shell of faceted_brep is a closed_shell and not an oriented_closed_shell.

NOTE 4 fbsr1 should be a rotated copy of fbsr.

NOTE 5 fbsrass should be equivalent to 2 copies of fbsr 'glued' together.

E.3.4.2.2 Postprocessor verdict criteria

FB3: After processing the solid tetrahedron defined by **mapped_item** shall be correctly defined and positioned.

FB4: Two separate geometric representation contexts are created, fbsr and fbsr1 are spatially related and should touch at one point only, fbsr1 and fbsrass are not spatially related, the shells in fbsrass should not intersect but should coincide over a common face which is implicitly shared.

E.3.5 Test case fb5

Test case fb5 is designed to test the use of **mapped_item**s in conjunction with a **cartesian_transformation_operator** in the creation of a simple assembly of faceted B-reps. The use of a scaling factor is tested. This test makes use of the **tetrashell_instance** context to define the geometry and topology.

E.3.5.1 AIM test purposes covered

FB21: mapped_item with mapping_target as cartesian_transformation_operator_3d;

FB22: cartesian_transformation_operator as cartesian_transformation_operator_3d with scale as REAL not equal to 1.0.

E.3.5.2 Postprocessor input specification

The model is defined by a **faceted_brep_shape_representation** consisting of a single **faceted_brep**. The **faceted_brep** is in the form of a solid tetrahedron with one vertex at the origin and the adjacent edges along the coordinate axes. This representation is then used in conjunction with the **mapped_item** entity and a cartesian_transformation_operator, to create, in the same **representation_context**, a representation consisting of a rotated and scaled (not by 1.0) copy of the original representation and the original **faceted_brep**. A full specification is provided in the EXPRESS-I definition.

E.3.5.2.1 EXPRESS-I specification: fbrep_5

```
  *)

TEST_CASE example_fbrep_5; WITH faceted_brep_aic;

  REALIZATION

  LOCAL
    origin : cartesian_point ;
    pos_z, neg_y, pos_x  : direction ;
    oldaxes : axis_placement_3d;
    transform : cartesian_transformation_operator_3d;
    shell_object  : closed_shell ;
    tetrahedron : faceted_brep ;
    fbsr, fbsrass : faceted_brep_shape_representation ;
    its_units : named_unit ;
    grc1, grc2 : representation_context ;
    tetratrans  : mapped_item ;
    mapping1  : representation_map ;
  END_LOCAL;

    CALL tetrashell_instance ; -- uses default values, so no WITH
      IMPORT (shell_object := @tetrashell;
              origin := @origin;
              neg_y := @ neg_y; );
    END_CALL;


tetrahedron := faceted_brep ('tetrahedron', shell_object) ;

its_units := length_unit() || si_unit ('milli', 'metre') ;
```

```
grc1 := geometric_representation_context ('grc1',
    'context for tetrahedron', 3) ||
  global_unit_assigned_context ( [its_units] ) ;

grc2 := geometric_representation_context ('grc2',
      'context for assembly', 3) ||
      global_unit_assigned_context ( [its_units] ) ;

(*  Define axis_placement and cartesian_transformation_operator for use in
mapping *)

pos_x := direction ('pos_x', [1, 0, 0]) ;

pos_z := direction ('pos_z', [0, 0, 1]) ;

oldaxes := axis2_placement_3d ('oldaxes', origin, pos_z, pos_x) ;

transform := cartesian_transformation_operator_3d ('transform',
                                    pos_x, neg_y, origin, 0.75, pos_z );

fbsr := faceted_brep_shape_representation ('fbsr',
                                 [tetrahedron, oldaxes], grc1 ) ;

mapping1 := representation_map (oldaxes, fbsr );
(* tetratrans is a 75% scaled copy of original reflected in ZX plane *)
tetratrans := mapped_item ('tetrarot1', mapping1, transform );

(* Define representation that is an assembly of tetrahedron +
transformed (scaled and reflected) copy.*)


fbsrass := faceted_brep_shape_representation
         ('fbsrass', [tetrahedron, tetratrans], grc2 ) ;

  END_REALIZATION;
END_TEST_CASE;
(*
```

NOTE 1 Test case of mapped_item and 'assembly' using simple solid tetrahedron.

NOTE 2 Tetrahedron shell is created by using context with default parameters.

NOTE 3 **tetratrans** should be a scaled copy of **fbsr** after reflection in OZX plane.

E.3.5.3 Postprocessor verdict criteria

FB21: After processing the solid tetrahedron defined by **mapped_item** shall be correctly defined and positioned.

FB22: **fbrass** should consist of two solid tetrahedrons that are in contact in the ZX plane. One is a 3/4 size copy of the other after reflection in this plane.

E.4 Contexts defined for test cases of faceted B-rep

The EXPRESS-I context below is used in the test cases in clause E.3.

Tetrashell_instance context

This context provides the facility to define a simple **closed_shell** of tetrahedral shape with vertices at (orc,orc,orc), (lx,orc,orc), (orc,ly,orc) and (orc,orc,lz). All bounds are defined by **poly_loop**s.

```
*)
CONTEXT tetrashell_instance ;

PARAMETER
  orc        : length_measure := 0;
  lx         : length_measure := 100;
  ly         : length_measure := 100;
  lz         : length_measure := 100;

  origin     : cartesian_point := cartesian_point ('origin', [orc, orc, orc]);
  p_x        : cartesian_point := cartesian_point ('p_x', [lx, orc, orc]);
  p_y        : cartesian_point := cartesian_point ('p_y', [orc, ly, orc]);
  p_z        : cartesian_point := cartesian_point ('p_z', [orc, orc, lz]);

  neg_x      : direction := direction ('neg_x', [-1, 0, 0]);
  neg_y      : direction := direction ('neg_y', [0, -1, 0]);
  neg_z      : direction := direction ('neg_z', [0, 0, -1]);
  dslope     : direction := direction ('dslope', [1, 1, 1]);
  dperp      : direction := direction ('dperp', [1, -1, 0]);

  loop_x     : poly_loop := poly_loop ('loop_x', [origin, p_z, p_y]) ;
  loop_y     : poly_loop := poly_loop ('loop_y', [origin, p_x, p_z]) ;
  loop_z     : poly_loop := poly_loop ('loop_z', [origin, p_y, p_x]) ;
 loop_slope : poly_loop := poly_loop ('loop_slope', [p_z, p_x, p_y]) ;

  a1         : axis2_placement_3d := axis2_placement_3d ('a1', origin,
                                                        neg_x, neg_y) ;
  a2         : axis2_placement_3d := axis2_placement_3d ('a2', origin,
```

```
                                                          neg_y, neg_x) ;
  a3        : axis2_placement_3d := axis2_placement_3d ('a3', origin,
                                                            neg_z, neg_y) ;
  a4        : axis2_placement_3d := axis2_placement_3d ('a4', p_x, dslope,
                                                                 dperp) ;

  p1        : plane := plane ('p1', a1) ;
  p2        : plane := plane ('p2', a2) ;
  p3        : plane := plane ('p3', a3) ;
  p4        : plane := plane ('p4', a4) ;

  b1        : face_outer_bound := face_outer_bound ('b1', loop_x, TRUE) ;
  b2        : face_outer_bound := face_outer_bound ('b2', loop_y, TRUE) ;
  b3        : face_outer_bound := face_outer_bound ('b3', loop_z, TRUE) ;
  b4        : face_outer_bound := face_outer_bound ('b4', loop_slope, TRUE) ;

  fs1 : face_surface := face_surface ('fs1', [b1], p1, TRUE) ;
  fs2 : face_surface := face_surface ('fs2', [b2], p2, TRUE) ;
  fs3 : face_surface := face_surface ('fs3', [b3], p3, TRUE) ;
  fs4 : face_surface := face_surface ('fs4', [b4], p4, TRUE) ;
END_PARAMETER;

SCHEMA_DATA tetra_ctxt;

cfs = connected_face_set {SUBOF(@tri); cfs_faces ->
                              ([@fs1, @fs2, @fs3, @fs4]);
                          SUPOF(@tetrashell);} ;

tri = topological_representation_item {SUBOF(@ri); SUPOF(@cfs);} ;

ri = representation_item {name -> 'tetrashell';
                                        SUPOF(@tri);} ;

tetrashell = closed_shell {SUBOF(@cfs); };

END_SCHEMA_DATA;

END_CONTEXT ;
(*
```

Index

ICS 25.040.40

Price based on 35 pages