ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО 10303-513— 2009

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 513

Прикладные интерпретированные конструкции. Элементарное граничное представление

ISO 10303-513:2000
Industrial automation systems and integration — Product data representation and exchange — Part 513: Application interpreted construct: Elementary boundary representation
(IDT)

Издание официальное

БЗ 3—2009/108

Москва
Стандартинформ
2010

Предисловие

Цели и принципы стандартизации в Российской Федерации установлены Федеральным законом от 27 декабря 2002 г. № 184-ФЗ «О техническом регулировании», а правила применения национальных стандартов Российской Федерации — ГОСТ Р 1.0—2001 «Стандартизация в Российской Федерации. Основные положения»

Сведения о стандарте

1 ПОДГОТОВЛЕН Государственным научным учреждением «Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики» на основе собственного аутентичного перевода на русский язык стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 459 «Информационная поддержка жизненного цикла изделий»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 14 сентября 2009 г. № 367-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 10303-513:2000 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 513. Прикладные интерпретированные конструкции. Элементарное граничное представление» (ISO 10303-513:2000 «Industrial automation systems and integration — Product data representation and exchange — Part 513: Application interpreted construct: Elementary boundary representation»).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

Информация об изменениях к настоящему стандарту публикуется в ежегодно издаваемом информационном указателе «Национальные стандарты», а текст изменений и поправок — в ежемесячно издаваемых информационных указателях «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ежемесячно издаваемом информационном указателе «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет

© Стандартинформ, 2010

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения . . . 1
2 Нормативные ссылки . . . 1
3 Термины и определения . . . 2
3.1 Термины, определенные в ИСО 10303-1 . . . 2
3.2 Термины, определенные в ИСО 10303-42 . . . 3
3.3 Термин, определенный в ИСО 10303-202 . . . 3
3.4 Термин, определенный в ИСО 10303-514 . . . 3
3.5 Другие определения . . . 3
4 Сокращенный листинг на языке EXPRESS . . . 3
4.1 Основные понятия и допущения . . . 4
4.2 Определения объекта elementary_brep_shape_representation схемы aic_elementary_brep . . . 5
Приложениие А (обязательное) Сокращенное наименование объекта . . . 9
Приложение В (обязательное) Регистрация информационного объекта . . . 9
Приложение С (справочное) Машинно-интерпретируемые листинги . . . 10
Приложение D (справочное) EXPRESS-G диаграммы . . . 10
Приложение Е (справочное) Требования соответствия и цели тестирования ПИК . . . 15
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации . . . 39

Введение

Стандарты комплекса ИСО 10303 распространяются на компьютерное представление информации об изделиях и обмен данными об изделиях. Их целью является обеспечение нейтрального механизма, способного описывать изделия на всем протяжении их жизненного цикла. Этот механизм применим не только для обмена файлами в нейтральном формате, но является также основой для реализации и совместного доступа к базам данных об изделиях и организации архивирования.

Стандарты комплекса ИСО 10303 представляют собой набор отдельно издаваемых стандартов (частей). Стандарты данного комплекса относятся к одной из следующих тематических групп: «Методы описания», «Методы реализации», «Методология и основы аттестационного тестирования», «Интегрированные обобщенные ресурсы», «Интегрированные прикладные ресурсы», «Прикладные протоколы», «Комплекты абстрактных тестов», «Прикладные интерпретированные конструкции» и «Прикладные модули». Настоящий стандарт входит в группу «Прикладные интерпретированные конструкции».

Прикладная интерпретированная конструкция (ПИК) обеспечивает логическую группировку интерпретированных конструкций, поддерживающих конкретную функциональность для использования данных об изделии в разнообразных прикладных контекстах. Интерпретированная конструкция представляет собой обычную интерпретацию интегрированных ресурсов, поддерживающую требования совместного использования информации прикладными протоколами.

Настоящий стандарт определяет прикладную интерпретированную конструкцию для определения граничного представления твердого тела с элементарной геометрией и явной топологией.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 513

Прикладные интерпретированные конструкции. Элементарное граничное представление

Industrial automation systems and integration. Product data representation and exchange. Part 513. Application interpreted construct. Elementary boundary representation

Дата введения — 2010—07—01

1 Область применения

Настоящий стандарт определяет интерпретацию обобщенных ресурсов, обеспечивающую соответствие требованиям к определению модели элементарного граничного представления.

Требования настоящего стандарта распространяются на:

- определение объекта **elementary_brep_shape_representation**, являющегося представлением, образованным одним или несколькими объектами **manifold_solid_brep**, каждый из которых определен элементарной геометрией и полностью явной топологией;
- определение неограниченной геометрии кривых и поверхностей, используемых для определения граней B-rep модели;
- определение топологической структуры B-rep модели;
- трехмерную геометрию;
- B-rep модели (модели граничного представления);
- элементарные кривые, представляемые объектами **line** и **conic**;
- объекты **elementary_surface**;
- геометрические преобразования;
- объекты **polyline**;
- неограниченную геометрию;
- использование топологии для ограничения геометрических объектов.

Требования настоящего стандарта не распространяются на:

- двумерную геометрию;
- ограниченные кривые, кроме объектов **polyline**;
- ограниченные поверхности;
- смещенные кривые и поверхности.

ПИК, определенная настоящим стандартом, не зависит от какой-либо промышленной прикладной области.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие международные стандарты:

ИСО/МЭК 8824-1:1995 Информационные технологии. Взаимосвязь открытых систем. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации (ISO/IEC 8824-1:1995, Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation)

Издание официальное

ИСО 10303-1:1994 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы (ISO 10303-1:1994, Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles)

ИСО 10303-11:1994 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS (ISO 10303-11:2004, Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual)

ИСО/ТО 10303-12:1997 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 12. Методы описания. Справочное руководство по языку EXPRESS-I (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)

ИСО 10303-41:1994 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 41. Интегрированные обобщенные ресурсы. Основы описания и поддержки изделий (ISO 10303-41:1994, Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support)

ИСО 10303-42:1994 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 42. Интегрированные обобщенные ресурсы. Геометрическое и топологическое представление (ISO 10303-42:2003, Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation)

ИСО 10303-43:1994 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 43. Интегрированные обобщенные ресурсы. Структуры представлений (ISO 10303-43:1994, Industrial automation systems and integration — Product data representation and exchange — Part 43. Integrated generic resources: Representation structures)

ИСО 10303-202:1996 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 202. Прикладные протоколы. Ассоциативные чертежи (ISO 10303-202:1996, Industrial automation systems and integration — Product data representation and exchange — Part 202: Application protocol: Associative draughting)

ИСО 10303-514:1999 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 514. Прикладные интерпретированные конструкции. Расширенное граничное представление (ISO 10303-514:1999, Industrial automation systems and integration — Product data representation and exchange — Part 514: Application interpreted construct: Advanced boundary representation)

3 Термины и определения

3.1 Термины, определенные в ИСО 10303-1

В настоящем стандарте применены следующие термины:

- **приложение** (application);
- **прикладной контекст** (application context);
- **прикладной протокол**; ПП (application protocol; AP);
- **метод реализации** (implementation method);
- **интегрированный ресурс** (integrated resource);
- **интерпретация** (interpretation);
- **данные об изделии** (product data).

3.2 Термины, определенные в ИСО 10303-42

В настоящем стандарте применены следующие термины:

- **линейно связный** (arcwise connected);
- **граница** (boundary);
- **ограничения** (bounds);
- **координатное пространство** (coordinate space);
- **кривая** (curve);
- **незамкнутая кривая** (open curve);
- **ориентируемый** (orientable);
- **поверхность** (surface);

- **топологическое значение** (topological sense).

3.3 Термин, определенный в ИСО 10303-202

В настоящем стандарте применен следующий термин:

прикладная интерпретированная конструкция (ПИК) (application interpreted construct; AIC).

3.4 Термин, определенный в ИСО 10303-514

В настоящем стандарте применен следующий термин:

односвязное твердое тело (manifold solid).

3.5 Другие определения

В настоящем стандарте также применены следующие термины с соответствующими определениями:

3.5.1 **представление формы элементарными B-rep моделями** (elementary B-rep shape representation): Представление формы, состоящее из одной или более B-rep моделей односвязных твердых тел. Каждая составляющая B-rep модели должна иметь грани и ребра, определенные посредством элементарной геометрии.

3.5.2 **элементарная геометрия** (elementary geometry): Геометрия, образованная линиями, полилиниями, линиями второго порядка и элементарными поверхностями.

4 Сокращенный листинг на языке EXPRESS

В настоящем разделе определена EXPRESS-схема, в которой используются элементы интегрированных ресурсов и содержатся типы, конкретизации объектов и функции, относящиеся к настоящему стандарту.

Примечание — В интегрированных ресурсах допускается существование подтипов и элементов списков выбора, не импортированных в данную ПИК. Такие конструкции исключают из дерева подтипов или из списка выбора посредством правил неявного интерфейса, определенных в ИСО 10303-11. Ссылки на исключенные конструкции находятся вне области применения данной ПИК. В некоторых случаях исключаются все элементы списка выбора. Поскольку ПИК предназначены для реализации в контексте прикладного протокола, элементы списка выбора будут определяться областью применения прикладного протокола.

Данная прикладная интерпретированная конструкция предоставляет непротиворечивое множество геометрических и топологических объектов для определения моделей односвязных твердых тел с гранями, имеющими элементарную геометрию и явно определенными ребрами и вершинами. Грани B-rep моделей ограничены полилиниями, линиями или кривыми второго порядка.

Объектом самого верхнего уровня в данной ПИК является объект **elementary_brep_shape_representation**. Этот объект является конкретизацией объекта **shape_representation** (см. ИСО 10303-41), состоящей из объектов **manifold_solid_brep** и **mapped_item**, определенных как поступательно перемещенные или преобразованные копии объектов **manifold_solid_brep** с элементарной геометрией.

EXPRESS-спецификация

```
*)
SCHEMA aic_elementary_brep;
  USE FROM geometry_schema(axis2_placement_3d,
                           cartesian_point,
                           cartesian_transformation_operator_3d,
                           circle,
                           conical_surface,
                           cylindrical_surface,
                           degenerate_toroidal_surface,
                           direction,
                           ellipse,
                           hyperbola,
                           line,
                           parabola,
                           plane,
```

```
                                    polyline,
                                    spherical_surface,
                                    vector);
  USE FROM geometric_model_schema(manifold_solid_brep,
                                    brep_with_voids);

REFERENCE FROM geometric_model_schema(msb_shells);

USE FROM topology_schema(closed_shell,
                                    connected_face_set,
                                    edge_curve,
                                    edge_loop,
                                    face_bound,
                                    face_outer_bound,
                                    face_surface,
                                    oriented_closed_shell,
                                    vertex_loop,
                                    vertex_point);

  USE FROM representation_schema(mapped_item);

  USE FROM product_property_representation_schema(shape_representation);
(*
```

Примечания

1 Для объекта **connected_face_set** установлены явные интерфейсы (т.е. они включены в списки оператора USE FROM), чтобы разрешить правилам, определенным для объекта **elementary_brep_shape_representation**, доступ к атрибутам этого объекта. При использовании данной ПИК данный объект должен быть реализован в виде одного из своих подтипов.

2 Схемы, на которые выше даны ссылки, можно найти в следующих стандартах комплекса ИСО 10303:

geometry_schema	– ИСО 10303-42;
geometric_model_schema	– ИСО 10303-42;
representation_schema	– ИСО 10303-43;
product_property_representation_schema	– ИСО 10303-41.

4.1 Основные понятия и допущения

Для независимой реализации в схемах прикладных протоколов, использующих данную ПИК, предназначены следующие объекты:

- axis2_placement_3d;
- brep_with_voids;
- cartesian_point;
- cartesian_transformation_operator_3d;
- circle;
- closed_shell;
- conical_surface;
- cylindrical_surface;
- degenerate_toroidal_surface;
- direction;
- edge_curve;
- edge_loop;
- elementary_face;
- ellipse;
- face_bound;
- face_outer_bound;
- face_surface;
- hyperbola;
- line;
- manifold_solid_brep;
- mapped_item;
- oriented_closed_shell;

- parabola;
- plane;
- polyline;
- representation_map;
- spherical_surface;
- toroidal_surface;
- vector;
- vertex_loop;
- vertex_point.

Прикладной протокол, использующий данную ПИК, должен обеспечивать поддержку всех вышеперечисленных объектов.

Прикладной протокол, использующий данную ПИК, должен допускать реализацию объекта **shape_representation** как объекта **elementary_brep_shape_representation**.

4.2 Определения объекта elementary_brep_shape_representation схемы aic_elementary_brep

Объект **elementary_brep_shape_representation** является подтипом объекта **shape_representation**, в котором элементы представления являются конкретизациями объектов **manifold_solid_brep**. Эти конкретизации отличаются от более общей B-rep модели тем, что для представления граней и ребер используются только явные геометрические формы. Геометрия граней ограничена объектами **elementary_surface** и линиями ребер, которые могут быть представлены объектами **line**, **polyline** или **conic**.

EXPRESS-спецификация

```
*)
ENTITY elementary_brep_shape_representation
SUBTYPE OF (shape_representation);
WHERE
    WR1 : SIZEOF (QUERY (it <* SELF.items |
            NOT (SIZEOF (['AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP',
                    'AIC_ELEMENTARY_BREP.FACETED_BREP',
                    'AIC_ELEMENTARY_BREP.MAPPED_ITEM',
                    'AIC_ELEMENTARY_BREP.AXIS2_PLACEMENT_3D'] *
                        TYPEOF(it)) = 1))) = 0;
    WR2 : SIZEOF (QUERY (it <* SELF.items |
          SIZEOF(['AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP',
          'AIC_ELEMENTARY_BREP.MAPPED_ITEM'] * TYPEOF(it)) =1 )) > 0;
    WR3 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
              'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
              NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
                NOT (SIZEOF (QUERY(fcs <* csh.cfs_faces |
                    NOT('AIC_ELEMENTARY_BREP.FACE_SURFACE' IN TYPEOF(fcs)))) = 0
                        ))) = 0
                          ))) = 0;
    WR4 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
              'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
              NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
                NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                    NOT(('AIC_ELEMENTARY_BREP.ELEMENTARY_SURFACE' IN
                        TYPEOF(fcs\face_surface.face_geometry))
                  ))) = 0
                    ))) = 0
                      ))) = 0;
    WR5 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
              'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
              NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
                NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                    NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
```

```
            'AIC_ELEMENTARY_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
                  NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                        NOT('AIC_ELEMENTARY_BREP.EDGE_CURVE' IN
               TYPEOF(oe.edge_element)))) = 0
                        ))) = 0
                     ))) = 0
                  ))) = 0
               ))) = 0;
   WR6 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
            'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
            NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
              NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                  NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
            'AIC_ELEMENTARY_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
                  NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                        NOT (SIZEOF (['AIC_ELEMENTARY_BREP.LINE',
                              'AIC_ELEMENTARY_BREP.CONIC',
                              'AIC_ELEMENTARY_BREP.POLYLINE'] *
            TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
                  )) = 0
                  ))) = 0
                     ))) = 0
                        ))) = 0
                           ))) = 0;
   WR7 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
            'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
            NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
                NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                     NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
            'AIC_ELEMENTARY_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
                     NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
             NOT(('AIC_ELEMENTARY_BREP.VERTEX_POINT' IN TYPEOF(oe.edge_start))
                  AND ('AIC_ELEMENTARY_BREP.VERTEX_POINT' IN
                    TYPEOF(oe.edge_end))
               ))) = 0
                  ))) = 0
                     ))) = 0
                        ))) = 0
                           ))) = 0;
   WR8 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
            'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
            NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
                NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                     NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
           'AIC_ELEMENTARY_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
                     NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                           ('AIC_ELEMENTARY_BREP.POLYLINE' IN
            TYPEOF(oe.edge_element\edge_curve.edge_geometry)) AND
            (NOT (SIZEOF (oe\oriented_edge.edge_element\
                  edge_curve.edge_geometry\polyline.points) >= 3))
               )) = 0
               ))) = 0
                  ))) = 0
                     ))) = 0
                        ))) = 0;
   WR9 : SIZEOF (QUERY (msb <* QUERY (it <* items |
            'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
            'AIC_ELEMENTARY_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF
```

```
                (msb\manifold_solid_brep.outer)))
            = 0;
    WR10 : SIZEOF (QUERY (brv <* QUERY (it <* items |
            'AIC_ELEMENTARY_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) |
            NOT (SIZEOF (QUERY (csh <* brv\brep_with_voids. voids |
              csh\oriented_closed_shell.orientation)) = 0))) = 0;
    WR11 : SIZEOF (QUERY (mi <* QUERY (it <* items |
                'AIC_ELEMENTARY_BREP.MAPPED_ITEM' IN TYPEOF(it)) |
          NOT ('AIC_ELEMENTARY_BREP.ELEMENTARY_BREP_SHAPE_REPRESENTATION' IN
            TYPEOF(mi\mapped_item.mapping_source.
                        mapped_representation)))) = 0;
    WR12 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
            'AIC_ELEMENTARY_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
            NOT (SIZEOF (QUERY (csh <* msb_shells(msb) |
              NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                NOT (SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* fcs.bounds |
          'AIC_ELEMENTARY_BREP.VERTEX_LOOP' IN TYPEOF(bnds.bound)) |
            NOT (('AIC_ELEMENTARY_BREP.VERTEX_POINT' IN
              TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
                ('AIC_ELEMENTARY_BREP.CARTESIAN_POINT' IN
                  TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.
                    loop_vertex\vertex_point.vertex_geometry))
            ))) = 0))) = 0))) = 0))) = 0;
END_ENTITY;
(*
```

Формальные утверждения

WR1 — атрибут **items**, супертипа **representation** должен содержать только объекты **manifold_solid_brep, mapped_item** и **axis2_placement_3d**. Согласно этому правилу, использование объектов **faceted_brep** недопустимо, поскольку экземпляр объекта **faceted_brep** также имел бы тип **manifold_solid_brep.**

WR2 — по крайней мере один элемент из множества элементов **items** должен быть объектом **manifold_solid_brep** или **mapped_item** (см. также WR11).

WR3 — Все грани, используемые для построения объекта **manifold_solid_brep** должны иметь тип **face_surface**.

Примечание — Вызов функции **msb_shells** в WR3 и последующих утверждениях является корректным, так как, хотя обобщенным типом аргумента 'msb' является **representation_item**, но оператором QUERY для него был определен тип **manifold_solid_brep**.

WR4 — для каждого объекта **manifold_solid_brep** из множества элементов **items**, ассоциированная поверхность для каждой грани должна быть объектом **elementary_surface**.

WR5 — для каждого объекта **manifold_solid_brep** из множества элементов **items**, ребра, используемые для определения границ, должны иметь тип **edge_curve**.

WR6 — для каждого объекта **manifold_solid_brep** из множества элементов **items**, каждая кривая, используемая для определения ограничений граней, должна быть объектом **conic**, **line** или **polyline**.

WR7 — для каждого объекта **manifold_solid_brep** из множества элементов **items**, все ребра, используемые для определения границ, должны быть обрезаны вершинами, имеющими тип **vertex_point**.

WR8 — для каждого объекта **manifold_solid_brep** из множества элементов **items**, каждый объект **polyline**, используемый для определения части ограничений грани, должен содержать три или более точек.

WR9 — для каждого объекта **manifold_solid_brep** из множества элементов **items**, атрибут внешней оболочки не должен иметь тип **oriented_closed_shell**.

WR10 — если объект **brep_with_voids** включены в множество элементов **items**, то каждая оболочка в множестве **voids** должна быть объектом **oriented_closed_shell** со значением ориентации FALSE.

WR11 — если объект **mapped_item** включен в множество элементов **items**, то объект **mapped_representation** атрибута **mapping_source** должен быть объектом **elementary_brep_shape_representation**.

Примечание — Если объект **cartesian_transformation_operator_3d** включен как объект **mapped_item.mapping_target** с объектом **axis2_placement_3d,** который соответствует в исходной системе координат объекту **mapped_representation.mapping_origin,** то результирующий объект **mapped_item** является преобразованной копией объекта **elementary_brep_shape_representation**. Точное определение данного преобразования, включая поступательное перемещение, вращение, масштабирование и, при необходимости, зеркалирование, задается оператором преобразования.

WR12 — для каждого объекта **manifold_solid_brep** из множества элементов **items** любой объект **vertex_loop,** используемый для определения ограничения грани, должен ссылаться на объект **vertex_point** с геометрией, определяемой объектом **cartesian_point**.

EXPRESS-спецификация

```
*)
  END_SCHEMA; -- конец схемы AIC_ELEMENTARY_BREP
(*
```

Приложение А
(обязательное)

Сокращенное наименование объекта

Сокращенное наименование объекта, установленного в настоящем стандарте, приведено в таблице А.1. Требования к использованию сокращенных наименований объектов содержатся в стандартах тематической группы «Методы реализации» комплекса ИСО 10303.

Таблица А.1 — Сокращенное наименование объекта

Полное наименование	Сокращенное наименование
ELEMENTARY_BREP_SHAPE_REPRESENTATION	EBSR

Приложение В
(обязательное)

Регистрация информационного объекта

В.1 Обозначение документа

Для обеспечения однозначного обозначения информационного объекта в открытой системе настоящему стандарту присвоен следующий идентификатор объекта:

{ iso standard 10303 part(513) version(1) }

Смысл данного обозначения установлен в ИСО/МЭК 8824-1 и описан в ИСО 10303-1.

В.2 Обозначение схемы

Для обеспечения однозначного обозначения в открытой информационной системе схеме aic_elementary_brep (см. раздел 4) присвоен следующий идентификатор объекта.

{ iso standard 10303 part(513) version(1) object(1) aic-elementary-brep(1) }

Смысл данного обозначения установлен в ИСО/МЭК 8824-1 и описан в ИСО 10303-1.

Приложение С
(справочное)

Машинно-интерпретируемые листинги

В данном приложении приведены ссылки на сайты, на которых находятся листинги наименований объектов на языке EXPRESS и соответствующих сокращенных наименований, установленных в настоящем стандарте. На этих же сайтах находятся листинги всех EXPRESS-схем, установленных или на которые даются ссылки в настоящем стандарте, без комментариев и другого поясняющего текста. Эти листинги доступны в машинно-интерпретируемой форме и могут быть получены по следующим адресам URL:

Сокращенные наименования: http://www.mel.nist.gov/div826/subject/apde/snr/

EXPRESS: http://www.mel.nist.gov/step/parts/part513/is/

При невозможности доступа к этим сайтам, необходимо обратиться в центральный секретариат ИСО или непосредственно в секретариат ИСО ТК184/ПК4 по адресу электронной почты: sc4sec@cme.nist.gov .

Примечание — Информация, представленная в машинно-интерпретируемой форме по указанным выше адресам URL, является справочной. Обязательным является текст настоящего стандарта.

Приложение D
(справочное)

EXPRESS-G диаграммы

Диаграммы, приведенные на рисунках D.1—D.4 получены из сокращенного листинга ПЭМ на языке EXPRESS, определенного в разделе 4. В диаграммах использована графическая нотация EXPRESS-G языка EXPRESS. Описание EXPRESS-G установлено в ИСО 10303-11, приложение D.

Примечания

1 Приведенные ниже выбранные типы импортируются в расширенный листинг ПИК в соответствии с правилами неявных интерфейсов по ИСО 10303-11. В настоящем стандарте эти выбранные типы в других объектах не используются:

- geometric_set_select;
- pcurve_or_surface;
- reversible_topology,
- shell;
- trimming_select;
- vector_or_direction.

2 Правила неявных интерфейсов, определенные в ИСО 10303-11, вводят также некоторые объекты, реализация которых запрещена правилами, относящимися к объекту **elementary_brep_shape_representation**. Эти объекты отмечены на EXPRESS-G диаграммах символом «*».

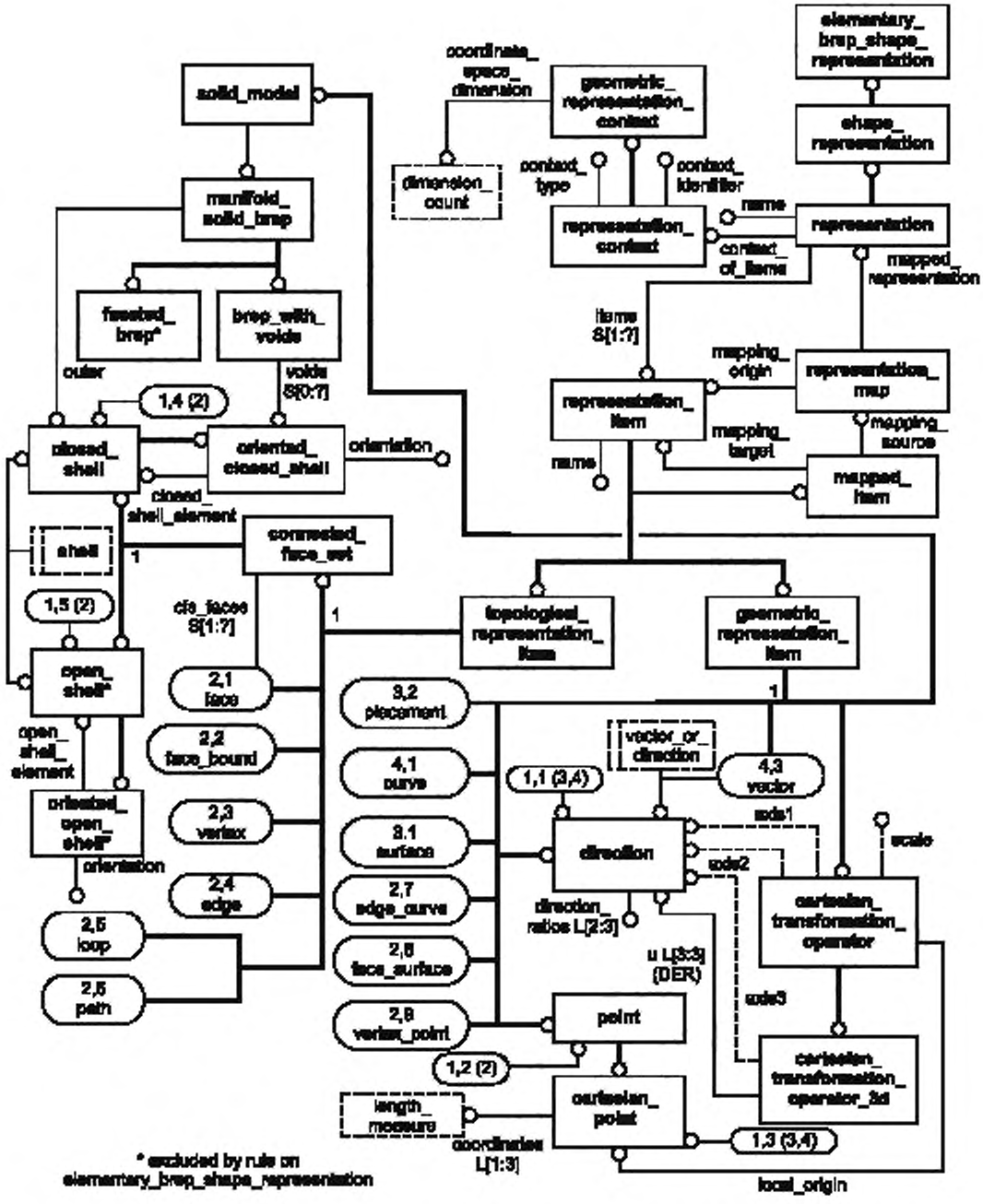

Рисунок D.1 — ПИК **elementary_boundary_representation** в формате EXPRESS-G (диаграмма 1 из 4)

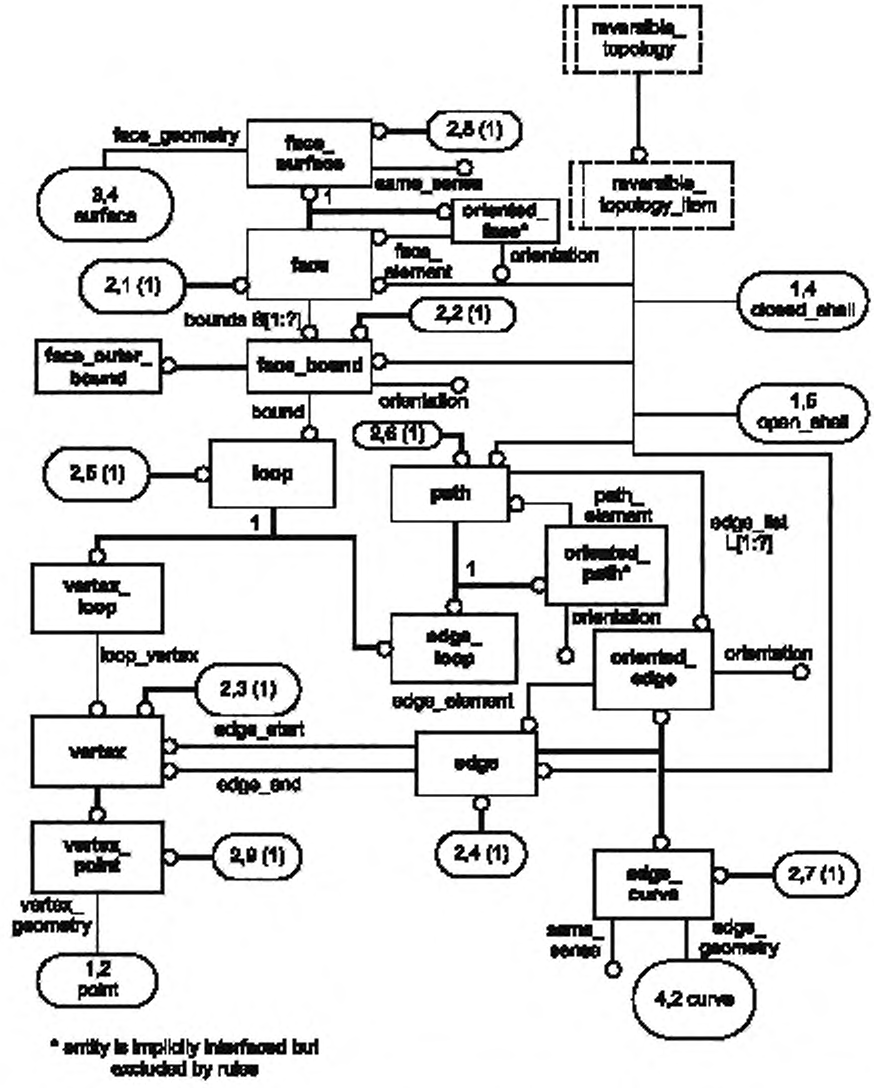

Рисунок D.2 — ПИК **elementary_boundary_representation** в формате EXPRESS-G (диаграмма 2 из 4)

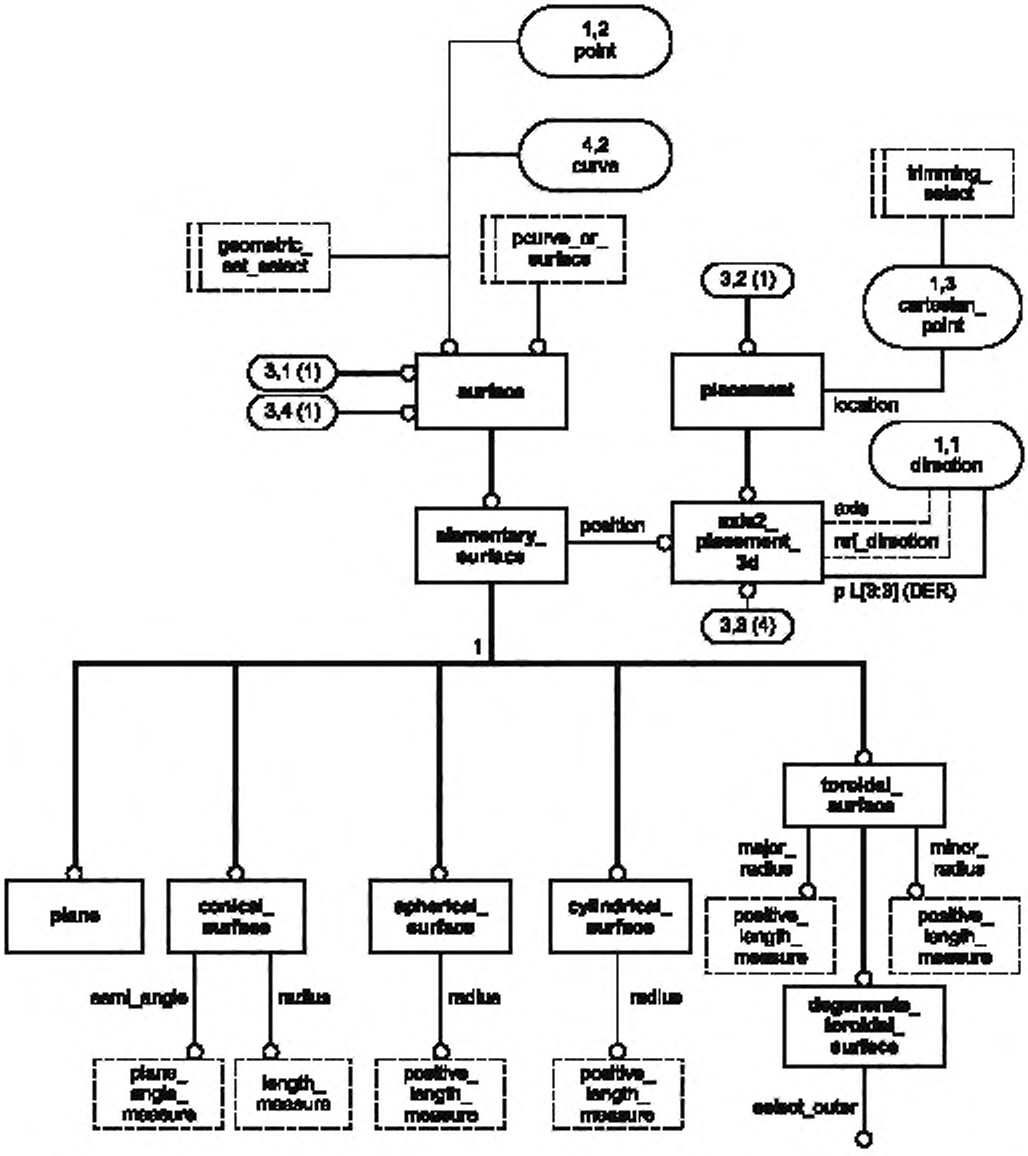

Рисунок D.3 — ПИК **elementary_boundary_representation** в формате EXPRESS-G (диаграмма 3 из 4)

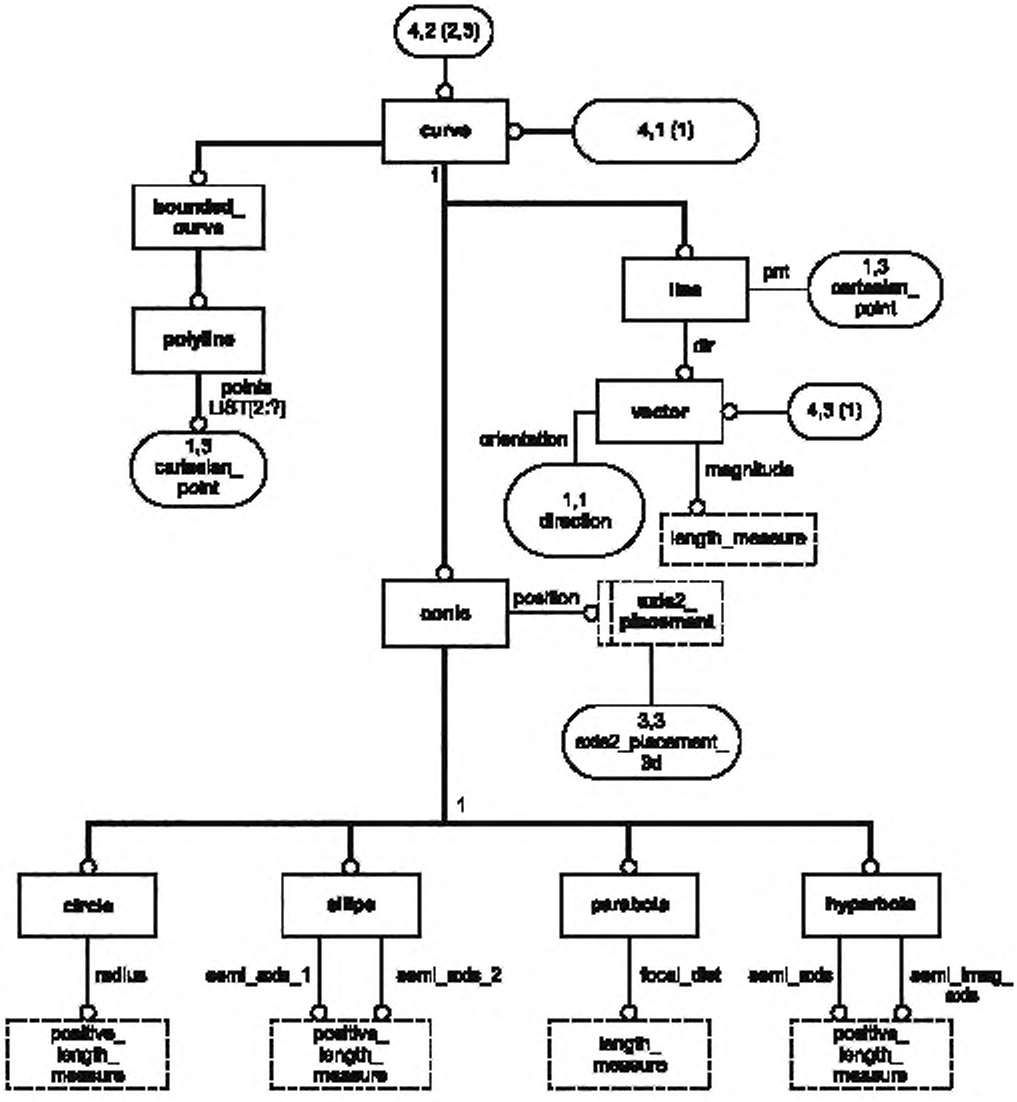

Рисунок D.4 — ПИК **elementary_boundary_representation** в формате EXPRESS-G (диаграмма 4 из 4)

Приложение Е
(справочное)

Требования соответствия и цели тестирования ПИК

Е.1 Требования соответствия ПИК: элементарная B-rep модель

Любой прикладной протокол, использующий данную ПИК, может потребовать согласования с определенными ниже требованиями соответствия ПИК при реализации объекта **elementary_brep_shape_representation**.

Соответствие данной ПИК означает, что должны поддерживаться все типы данных и типы объектов, определенные в листинге на языке EXPRESS. Единственным допустимым использованием экземпляра геометрического или топологического объекта в контексте данной ПИК является его использование для определения объекта **elementary_brep_shape_representation**.

Указанные ниже объекты могут быть реализованы как часть определения объекта **elementary_brep_shape_representation**:

- axis2_placement_3d;
- brep_with_voids;
- cartesian_point;
- cartesian_transformation_operator_3d;
- circle;
- closed_shell;
- conical_surface;
- cylindrical_surface;
- degenerate_toroidal_surface;
- direction;
- edge_curve;
- edge_loop,
- elementary_face;
- ellipse;
- face_bound;
- face_outer_bound;
- face_surface,
- hyperbola;
- line;
- manifold_solid_brep;
- mapped_item;
- oriented_closed_shell;
- parabola;
- plane;
- polyline;
- representation_map;
- spherical_surface;
- toroidal_surface;
- vector;
- vertex_loop;
- vertex_point.

Е.2 Цели тестирования ПИК элементарной B-rep модели

В настоящем разделе определены цели тестирования для ПИК элементарной B-rep модели. Цели тестирования основаны на конструкциях, определенных в разделе 4.

Е.2.1 Объект elementary_brep_shape_representation

На основании определения данного объекта сформулированы следующие цели тестирования:

EB1 — проверка объекта **representation** как объекта **shape_representation** и как объекта **elementary_brep_shape_representation** (см. Е.3.1).

EB2 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_context** с атрибутом **items**, представленным как объект **manifold_solid_brep** (см. Е.3.1).

EB3 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_context** с атрибутом **items**, представленным как объект **mapped_item** (см. Е.3.6).

EB4 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_context** с атрибутом **items**, представленным двумя или более элементами,представленными

как объекты **manifold_solid_brep**, или **mapped_item**, или **axis2_placement_3d**. При этом по меньшей мере один из них должен быть объектом **axis2_placement_3d** (см. Е.3.6).

Е.2.2 Объект manifold_solid_brep

На основании определения данного объекта сформулированы следующие цели тестирования:

EB5 — проверка объекта **manifold_solid_brep** с атрибутом **outer** (атрибут **voids** отсутствует), представленным как объект **closed_shell** (но не как подтип **oriented_closed_shell**) (см. Е.3.1).

EB6 — проверка объекта **manifold_solid_brep** как подтипа **brep_with_voids** с атрибутом **outer**, представленным как объект **closed_shell**, и атрибутом **voids**, представленным множеством, состоящим из одного объекта **oriented_closed_shell** (атрибут **voids** присутствует) (см. Е.3.2).

EB7 — проверка объекта **manifold_solid_brep** как подтипа **brep_with_voids** с атрибутом **outer**, представленным как объект **closed_shell**, и атрибутом **voids**, представленным множеством, состоящим из более чем одного объекта **oriented_closed_shell** (атрибут **voids** присутствует) (см. Е.3.2).

Е.2.3 Объект oriented_closed_shell

На основании определения данного объекта и ограничений, наложенных на объект **elementary_brep_shape_representation**, сформулирована следующая цель тестирования:

EB8 — проверка объекта **oriented_closed_shell** с атрибутом **orientation**, имеющим значение FALSE (см. Е.3.2).

Е.2.4 Объект closed_shell

На основании определения данного объекта и ограничений, наложенных на объект **elementary_brep_shape_representation**, сформулированы следующие цели тестирования:

EB9 — проверка объекта **closed_shell** с атрибутом **cfs_faces**, представленным множеством, состоящим из одного объекта **face_surface** (см. Е.3.2).

EB10 — проверка объекта **closed_shell** с атрибутом **cfs_faces**, представленным множеством, состоящим из более чем одного объекта **face_surface** (см. Е.3.1).

Е.2.5 Объект face

На основании определения данного объекта и ограничений, наложенных на объект **elementary_brep_shape_representation**, сформулированы следующие цели тестирования:

EB11 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим из одного объекта **face_bound**, представленного как объект **face_outer_bound** с атрибутом **orientation**, имеющим значение TRUE (см. Е.3.1).

EB12 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим из одного объекта **face_bound**, представленного как объект **face_outer_bound** с атрибутом **bound**, представленным объектом **edge_loop** (но не **oriented_path**) и атрибутом **orientation**, имеющим значение FALSE (см. Е.3.1).

EB13 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим, по меньшей мере, из двух объектов **face_bound** с атрибутом **bound**, представленным объектом **edge_loop** и атрибутом **orientation**, имеющим значение TRUE (см. Е.3.1).

EB14 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим, по меньшей мере, из двух объектов **face_bound** с атрибутом **bound**, представленным объектом **edge_loop** и атрибутом **orientation**, имеющим значение FALSE (см. Е.3.1).

EB15 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим, по меньшей мере, из двух объектов **face_bound** (включая один объект **vertex_loop**) (см. Е.3.5).

Е.2.6 Объект face_surface

На основании определения данного объекта и ограничений, наложенных на объект **elementary_brep_shape_representation**, сформулированы следующие цели тестирования:

EB16 — проверка объекта **face_surface** с атрибутом **face_geometry**, представленным как объект **surface** (см. Е.3.1).

EB17 — проверка объекта **face_surface** с атрибутом **same_sense**, имеющим значение TRUE (см. Е.3.1).

EB18 — проверка объекта **face_surface** с атрибутом **same_sense**, имеющим значение FALSE (см. Е.3.5).

Е.2.7 Объект surface

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулирована следующая цель тестирования:

EB19 — проверка объекта **surface** как объекта **elementary_surface** (см. Е.3.1).

Е.2.8 Объект elementary_surface

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулированы следующие цели тестирования:

EB20 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с присутствием атрибута **axis** (см. Е.3.1).

EB21 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с отсутствием атрибута **axis** (см. Е.3.4).

EB22 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с присутствием атрибута **ref_direction** (см. E.3.1).

EB23 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с отсутствием атрибута **ref_direction** (см. E.3.4).

EB24 — проверка объекта **elementary_surface** как объекта **plane** (см. E.3.1).

EB25 — проверка объекта **elementary_surface** как объекта **cylindrical_surface** (см. E.3.1).

EB26 — проверка объекта **elementary_surface** как объекта **conical_surface** (см. E.3.5).

EB27 — проверка объекта **elementary_surface** как объекта **spherical_surface** (см. E.3.1).

EB28 — проверка объекта **elementary_surface** как объекта **toroidal_surface** (см. E.3.3).

E.2.9 Объект loop

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулированы следующие цели тестирования:

EB29 — проверка объекта **loop** как объекта **edge_loop** (см. E.3.1).

EB30 — проверка объекта **loop** как объекта **vertex_loop** с атрибутом **loop_vertex**, представленным как объект **vertex_point** с атрибутом **vertex_geometry**, представленным как объект **cartesian_point** (см. E.3.2).

E.2.10 Объект edge

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулированы следующие цели тестирования:

EB31 — проверка объекта **edge** как объекта **edge_curve** с атрибутами **edge_start** и **edge_end**, представленными объектом **vertex_point** (см.E.3.1).

EB32 — проверка объекта **edge** как объекта **oriented_edge** с атрибутом **orientation**, имеющим значение TRUE (см. E.3.1).

EB33 — проверка объекта **edge** как объекта **oriented_edge** с атрибутом **orientation**, имеющим значение FALSE (см. E.3.3).

E.2.11 Объект edge_curve

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулированы следующие цели тестирования:

EB34 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **line** (см. E.3.3).

EB35 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **polyline** (см. E.3.4).

EB36 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **conic** (см. E.3.1).

EB37 — проверка объекта **edge_curve** с атрибутом **same_sense**, имеющим значение TRUE(см. E.3.1).

EB38 — проверка объекта **edge_curve** с атрибутом **same_sense**, имеющим значение FALSE (см. E.3.5).

E.2.12 Объект conic

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулированы следующие цели тестирования:

EB39 — проверка объекта **conic** как объекта **circle** (см. E.3.1).

EB40 — проверка объекта **conic** как объекта **ellipse** (см. E.3.1).

EB41 — проверка объекта **conic** как объекта **hyperbola** (см. E.3.5).

EB42 — проверка объекта **conic** как объекта **parabola** (см. E.3.5).

E.2.13 Объект polyline

На основании определения данного объекта и ограничений, наложенных на объект **face_surface**, сформулирована следующая цель тестирования:

EB43 — проверка объекта **polyline** с атрибутом **points**, представленным списком из трех или более объектов **cartesian_point** (см. E.3.4).

E.2.14 Объект cartesian_transformation_operator_3d

На основании определения данного объекта, объекта **mapped_item** и ограничений, наложенных на объект **elementary_brep_shape_representation**, сформулированы следующие цели тестирования:

EB44 — проверка объекта **mapped_item** с атрибутом **mapping_target**, представленным как объект **cartesian_transformation_operator_3d** (см. E.3.7).

EB45 — проверка объекта **cartesian_transformation_operator** как объекта **cartesian_transformation_operator_3d** с масштабом, представленным значением типа REAL, не равным 1,0 (см. E.3.7).

E.3 Абстрактные контрольные примеры для модели элементарного граничного представления

Абстрактные контрольные примеры для постпроцессора представлены в настоящем разделе на языке EXPRESS-I.

Для каждого контрольного примера для препроцессора приведено простое текстовое описание, чтобы обеспечить возможность создания модели, подобной той, которая представлена на языке EXPRESS-I для тестирования постпроцессора. Для каждого контрольного примера определен ряд относящихся к нему целей тестирования.

Примечание — Многие из целей тестирования применимы к нескольким контрольным примерам, но критерии определены только для первого из них. В частности, это относится ко многим целям, приведенным в контрольном примере eb1.

E.3.1 Контрольный пример eb1

Контрольный пример eb1 является наиболее фундаментальным контрольным примером, описывающим грани, необходимые для определения одиночного сплошного цилиндра с полусферическим основанием и эллиптической верхней гранью. Вся геометрия задана в явной форме без значений, принимаемых по умолчанию, и без необходимости изменения направлений на обратные. Определение граней обеспечивается контекстом объекта **cylinder_sphere_shell** с использованием исходных параметров.

E.3.1.1 Цели тестирования

Ниже перечислены цели тестирования прикладной интерпретированной модели, охватываемые данным контрольным примером.

EB1 — проверка объекта **representation** как объекта **shape_representation** и как объекта **elementary_brep_shape_representation**.

EB2 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_representation_context** с атрибутом **items**, представленным как объект **manifold_solid_brep**.

EB5 — проверка объекта **manifold_solid_brep** с атрибутом **outer** (атрибут **voids** отсутствует), представленным как объект **closed_shell** (но не как подтип **oriented_closed_shell**).

EB10 — проверка объекта **closed_shell** с атрибутом **cfs_faces**, представленным множеством, состоящим из более чем одного объекта **face_surface**.

EB11 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим из одного объекта **face_bound,** представленного как объект **face_outer_bound** с атрибутом **orientation**, имеющим значение TRUE.

EB12 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим из одного объекта **face_bound**, представленного как объект **face_outer_bound** с атрибутом **orientation**, имеющим значение FALSE.

EB13 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим, по меньшей мере, из двух объектов **face_bound** с атрибутом **bound**, представленным объектом **edge_loop** и атрибутом **orientation**, имеющим значение TRUE.

EB14 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим, по меньшей мере, из двух объектов **face_bound** с атрибутом **bound**, представленным объектом **edge_loop** и атрибутом **orientation**, имеющим значение FALSE.

EB16 — проверка объекта **face_surface** с атрибутом **face_geometry**, представленным как объект **surface**.

EB17 — проверка объекта **face_surface** с атрибутом **same_sense**, имеющим значение TRUE.

EB19 — проверка объекта **surface** как объекта **elementary_surface**.

EB20 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с присутствием атрибута **axis**.

EB22 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с присутствием атрибута **ref_direction**.

EB24 — проверка объекта **elementary_surface** как объекта **plane**.

EB25 — проверка объекта **elementary_surface** как объекта **cylindrical_surface**.

EB27 — проверка объекта **elementary_surface** как объекта **spherical_surface**.

EB29 — проверка объекта **loop** как объекта **edge_loop**.

EB31 — проверка объекта **edge** как объекта **edge_curve** с атрибутами **edge_start** и **edge_end**, представленными объектом **vertex_point**.

EB32 — проверка объекта **edge** как объекта **oriented_edge** с атрибутом **orientation**, имеющим значение TRUE.

EB36 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **conic**.

EB37 — проверка объекта **edge_curve** с атрибутом **same_sense**, имеющим значение TRUE.

EB39 — проверка объекта **conic** как объекта **circle**.

EB40 — проверка объекта **conic** как объекта **ellipse**.

E.3.1.2 Спецификация входа в препроцессор

Создается объект **elementary_brep_shape_representation**, состоящий из единственного объекта **manifold_solid_brep**. Объект **manifold_solid_brep** должен иметь форму сплошного цилиндра с полусферическим основанием и плоской наклонной верхней гранью. Центр полусферы находится в начале координат, а ось Z является осью цилиндра. B-rep модель представлена одной замкнутой оболочкой с тремя гранями. Соответствующее множество размеров определено в приведенной ниже спецификации на языке EXPRESS-I.

Примечание — Для определения граней B-rep модели использован контекст объекта **cylinder_sphere_shell** в своей простейшей форме с параметрами, заданными по умолчанию.

E.3.1.3 Спецификация входа в постпроцессор

```
*)
TEST_CASE example_ebrep_1; WITH aic_elementary_brep;

  REALIZATION

  LOCAL
    shell_object : closed_shell ;
    cysp_solid : manifold_solid_brep ;
    ebsr : elementary_brep_shape_representation ;
    its_units : named_unit ;
    its_context : representation_context ;
 END_LOCAL;

   CALL cylinder_sphere_shell ; -- используются значения по умолчанию
      IMPORT (shell_object := @cyspshell; ) ;
   END_CALL;
      cysp_solid := manifold_solid_brep ('cysp_solid', shell_object) ;
      its_units := length_unit() || si_unit ('milli', 'metre') ;

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

      ebsr := elementary_brep_shape_representation
                    ( 'ebsr', [cysp_solid], its_context );
 END_REALIZATION;
END_TEST_CASE;
(*
```

E.3.1.4 Критерии решения постпроцессора

EB1 — все формальные утверждения для объекта **elementary_brep_shape_representation** должны быть проверены.

EB2 — единицы длины должны быть правильно интерпретированы, воссозданная модель не должна содержать объекты **polyloop** или **vertex_loop**.

EB5 — перпендикуляры к оболочке должны быть направлены от нее.

EB10 — грани должны быть соединены вдоль ребер, другие пересечения граней недопустимы.

EB11 — геометрия граней должна быть правильно обрезана объектом **face_bound**.

EB12 —объекты **face_bound** с значением атрибута **orientation** FALSE должны быть правильно интерпретированы, чтобы определить правильную часть поверхности грани.

EB13 —Множество ограничений должно быть правильно обработано, чтобы обрезать грань.

EB14 — объекты **face_bound** с разными значениями атрибута **orientation** должны быть правильно интерпретированы.

EB16 — объекты **edge_curve** и вершины, ограничивающие объекты **edge_loop**, должны лежать на поверхности, определяющей объект **face_geometry**.

EB20 — объект **axis2_placement** с атрибутом **axis** должен быть правильно интерпретирован, чтобы определить положение поверхности.

EB22 — объект **axis2_placement** с атрибутом **ref_direction** должен быть правильно интерпретирован, чтобы определить положение поверхности.

EB24 — ограничивающие контуры грани с объектом **face_geometry**, представленным как объект **plane**, должны быть копланарны.

EB25 — неограниченный объект **cylindrical_surface** должен быть ограничен объектами **edge_loop**.

EB27 — правильная часть объекта **spherical_surface** должна быть определена объектами **edge_loop**.

EB31 — все объекты **vertex_point** должны принадлежать объектам **edge_curve**.

EB36 — объект **edge** с идентичными начальной и конечной вершинами и объект **edge_geometry**, представленный эллипсом, должны быть правильно интерпретированы как замкнутый эллипс.

EB39 — объект **edge** с идентичными начальной и конечной вершинами и объект **edge_geometry**, представленный окружностью, должны быть правильно интерпретированы как замкнутая окружность.

EB40 — подтип **ellipse** объекта **conic** должен быть правильно интерпретирован.

E.3.2 Контрольный пример eb2

Контрольный пример eb2 разработан для проверки определения элементарного граничного представления, содержащего одну или несколько пустот. Контекст объекта **cylinder_sphere_shell** используется с различными параметрами для определения внешней оболочки и оболочек пустот. Результатом является полый сплошной цилиндр с расположенной внутри пустотой (или пустотами) той же или сферической формы.

Примечание — При необходимости данный тест может быть легко модифицирован для проверки геометрической точности путем изменения параметров, определяющих пустоты, расположенных очень близко друг к другу или к внешней оболочке. В текущей версии данного контрольного примера возможность такого вмешательства не предусмотрена.

E.3.2.1 Цели тестирования

Ниже перечислены цели тестирования, охватываемые данным контрольным примером.

EB6 - проверка объекта **manifold_solid_brep** как подтипа **brep_with_voids** с атрибутом **outer**, представленным как объект **closed_shell**, и атрибутом **voids**, представленным множеством, состоящим из одного объекта **oriented_closed_shell** (атрибут **voids** присутствует).

EB7 — проверка объекта **manifold_solid_brep** как подтипа **brep_with_voids** с атрибутом **outer**, представленным как объект **closed_shell**, и атрибутом **voids**, представленным множеством, состоящим из более чем одного объекта **oriented_closed_shell**.

EB8 — проверка объекта **oriented_closed_shell** с атрибутом **orientation**, имеющим значение FALSE.

EB9 — проверка объекта **closed_shell** с атрибутом **cfs_faces**, представленным множеством, состоящим из одного объекта **face_surface**.

EB30 — проверка объекта **loop** как объекта **vertex_loop**.

E.3.2.2 Спецификация входа в препроцессор

Создается объект **elementary_brep_shape_representation**, состоящий из единственного объекта **manifold_solid_brep**. Объект **manifold_solid_brep** должен иметь форму сплошного цилиндра с полусферическим основанием и плоской наклонной верхней гранью. Одно такое граничное представление должно содержать пустоту такой же формы и ориентации. Второй пример должен содержать две непересекающиеся пустоты: одну — такой же формы, вторую — сферической формы. Пустота сферической формы должна быть определена одним объектом **face_surface** с использованием объекта **vertex_loop**. Центр полусферы для внешней оболочки находится в начале координат, а ось Z является осью цилиндра. Каждая оболочка определена как единая замкнутая оболочка с тремя гранями. Соответствующее множество размеров определено в приведенной ниже спецификации на языке EXPRESS-I.

E.3.2.3 Спецификация входа в постпроцессор

Примечания

1 Контекст объекта **cylinder_sphere_shell** с соответствующими параметрами используется для определения граней внешней оболочки B-rep модели и для определения оболочек пустот.

2 Внешняя оболочка объекта **brep_with_voids** является объектом **closed_shell**, но не объектом **oriented_closed_shell**. Объект **oriented_closed_shell** используется для определения оболочек пустот, при этом атрибут **orientation** должно иметь значение FALSE.

```
*)
TEST_CASE example_ebrep_2;WITHaic_elementary_brep;

  REALIZATION

  LOCAL
    shell_object, hollow1, hollow2 : closed_shell ;
     void1, void2 . oriented_closed_shell;
    cylsp_with_void : brep_with_voids ;
    cylsp_with_voids : brep_with_voids ;
    ebsr1, ebsr2 : elementary_brep_shape_representation ;
    its_units . named_unit ;
    context1, context2 : representation_context ,
    sph2 : spherical_surface .
    l1, l2 : length_measure;
    top_pt . cartesian_point;
    top_vert : vertex_point;
     v_loop : vertex_loop;
    s_bound : face_outer_bound ;
    sp_face : face_surface ,
    sp_shell : closed_shell ;
  END_LOCAL;
```

```
    CALL cylinder_sphere_shell ; -- используются значения по умолчанию
       IMPORT (shell_object := @cyspshell; );
    END_CALL;

    CALL cylinder_sphere_shell; -- переустанавливаются параметры размеров
        IMPORT (hollow1 := @cyspshell; ); -- большая пустота
         WITH (orc := 10; rad := 12; ht := 50; );
    END_CALL;

    CALL cylinder_sphere_shell; -- переустанавливаются параметры размеров
          (сфера для сферической пустоты)
       IMPORT (sph2 := @sphere,
                    l1 := @orc;
                    l2 := @rad ;);
      WITH (orc := -5; rad := 10 ,);
    END_CALL;

    void1 := oriented_closed_shell ('void1', hollow1, FALSE) ;
    top_pt := cartesian_point ('top_pt', [l1, l1, (l1 + l2) ]) ;
    top_v := vertex_point ('top_v', top_pt ) ;
    v_loop := vertex_loop ('v_loop', top_v ) ;
    s_bound := face_outer_bound ('s_bound', v_loop, TRUE ) ;
    sp_face := face_surface ('sp_face', [s_bound], sph2, TRUE );
    sp_shell := closed_shell ('sp_shell', [sp_face] );
    void2 := oriented_closed_shell ('void2', sp_shell, FALSE) ;

    cylsp_with_void :=
          manifold_solid_brep ('cylsp_w_v', shell_object) ||
                        brep_with_voids ([void1]) ;

    cylsp_with_voids :=
          manifold_solid_brep ('cylsp_w_vs', shell_object)||
                        brep_with_voids ([void1, void2]) ;

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

    context1 := geometric_representation_context ,
                        ('context_1', 'context_for_cylsp_with_void', 3) ||
                            global_unit_assigned_context ( [its_units] ) ;

    context2 := geometric_representation_context ,
                        ('context_1', 'context_for_cylsp_with_voids', 3) ||
                            global_unit_assigned_context ( [its_units] ) ;

    ebsr1 := elementary_brep_shape_representation
                ( 'ebsr1', [cylsp_with_void], context1 ) ,

    ebsr2 := elementary_brep_shape_representation
              ('ebsr2', [cylsp_with_voids], context2 ) ;

  END_REALIZATION,
END_TEST_CASE;
(*
```

E.3.2.4 Критерии решения постпроцессора

EB6 — оболочка пустоты не должна пересекаться с внешней оболочкой; оболочка пустоты должна полностью находиться внутри внешней оболочки.

EB7 — оболочки пустот не должны пересекаться с внешней оболочкой или друг с другом; каждая оболочка пустоты должна полностью находиться внутри внешней оболочки, два объекта **elementary_brep_shape_representation** ebsr1 и ebsr2 не должны быть пространственно связаны.

EB8 — перпендикуляры к оболочкам пустот должны быть направлены внутрь пустот.

EB9 — объект **closed_shell** с единственной гранью, геометрия которой определена объектом **spherical_surface**, должен быть правильно обработан.

EB30 — объект **vertex_loop** должен быть корректно обработан как объект **face_bound**, чтобы определить грань как полностью сферическую поверхность.

E.3.3 Контрольный пример eb3

Контрольный пример eb3 является простым контрольным примером, определяющим грани, необходимые для определения одиночного сплошного сегмента тора, ограниченного плоскостями. Одно из пересечений плоскости с тором представлено плоской полилинией. Определение оболочки обеспечивается контекстом объекта **toroidal_segment** с использованием исходных параметров.

E.3.3.1 Цели тестирования

Ниже перечислены цели тестирования, охватываемые данным контрольным примером.

EB28 — проверка объекта **elementary_surface** как объекта **toroidal_surface**.

EB33 — проверка объекта **edge** как объекта **oriented_edge** с атрибутом **orientation**, имеющим значение FALSE.

EB34 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **line**.

EB35 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **polyline**.

E.3.3.2 Спецификация входа в препроцессор

Создается объект **elementary_brep_shape_representation**, состоящей из единственного объекта **manifold_solid_brep**. Объект **manifold_solid_brep** должен иметь форму тороидального сегмента с центром, расположенным в начале координат, и центральной осью, направленной по оси Z. Сегмент создан сечением тора тремя плоскостями, одна из которых (z = 0) проходит через центр и перпендикулярна к центральной оси. Две другие плоскости параллельны друг другу, причем одна из них (x = 0) проходит через центр. Линии пересечений являются дугами окружностей или полилинией. B-rep модель определена одной замкнутой оболочкой с четырьмя гранями. Соответствующее множество размеров определено в приведенной ниже спецификации на языке EXPRESS-I.

E.3.3.3 Спецификация входа в постпроцессор

Примечание — Для определения граней и всей геометрии и топологии B-rep модели используется контекст объекта **toroidal_segment** с параметрами, заданными по умолчанию.

```
*)
TEST_CASE example_ebrep_3; WITH aic_elementary_brep;

  REALISATION

  LOCAL
    shell_object : closed_shell ;
    torus_solid : manifold_solid_brep ;
    ebsr : elementary_brep_shape_representation ;
    its_units . named_unit ;
    its_context : representation_context ;
  END_LOCAL;

    CALL toroidal_segment , -- используются значения по умолчанию
      IMPORT (shell_object := @torshell, ) ;
    END_CALL;
      torus_solid := manifold_solid_brep ('torus_solid', shell_object) ;

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

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

    ebsr := elementary_brep_shape_representation
                ('ebsr', [torus_solid], its_context );
  END_REALIZATION;
END_TEST_CASE;
(*
```

E.3.3.4 Критерии решения постпроцессора

EB28 — грань объекта **toroidal_surface** должна быть обработана и ограничена правильно.

EB33 — если объект **edge** с атрибутом **orientation**, имеющим значение FALSE, используется повторно, он должен быть интерпретирован правильно.

EB34 — объект **line** должен быть правильно обрезан объектами **vertice**.

EB35 — все точки объекта **polyline** должны принадлежать объекту **toroidal_surface** с допустимым отклонением менее чем 0,0000001. Точки полилинии должны быть копланарны.

E.3.4 Контрольный пример eb4

В контрольном примере eb4 определены грани, необходимые для определения одиночного сплошного тела, полученного путем объединения двух цилиндров разных радиусов с ортогональными осями. Линия пересечения является замкнутой трехмерной кривой, представленной полилинией. Определение оболочки обеспечивается контекстом объекта **cylinder_union_polyline** с использованием исходных параметров.

E.3.4.1 Цели тестирования

Ниже перечислены цели тестирования, охватываемые данным контрольным примером:

EB21 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с отсутствием атрибута **axis**.

EB23 — проверка объекта **elementary_surface** с атрибутом **position**, представленным как объект **axis2_placement_3d** с отсутствием атрибута **ref_direction**.

EB35 — проверка объекта **edge_curve** с атрибутом **edge_geometry**, представленным как объект **polyline** (замкнутая трехмерная полилиния).

E.3.4.2 Спецификация входа в препроцессор

Создается объект **elementary_brep_shape_representation**, состоящий из единственного объекта **manifold_solid_brep**. Объект **manifold_solid_brep** должен иметь форму двух перпендикулярных пересекающихся цилиндров. Для определения положения одного из этих цилиндров при определении объекта **axis2_placement_3d** должны быть использованы параметры, заданные по умолчанию. Линия пересечения должна быть представлена полилинией. Соответствующее множество размеров определено в приведенной ниже спецификации на языке EXPRESS-I.

E.3.4.3 Спецификация входа в постпроцессор

Примечание — Для определения граней и всей геометрии и топологии B-rep модели используется контекст объекта **cylinder_union_polyline** с параметрами, заданными по умолчанию.

```
*)
TEST_CASE example_ebrep_4; WITH aic_elementary_brep;

   OBJECTIVE

   REALIZATION

   LOCAL
      shell_object : closed_shell ;
      cylxcyl_solid : manifold_solid_brep ;
      ebsr : elementary_brep_shape_representation ;
      its_units : named_unit ;
      its_context : geometric_representation_context ;
   END_LOCAL;

      CALL cylinder_union_polyline ; -- используются значения по умолчанию
         IMPORT (shell_object := @cxcshell ;) ;
      END_CALL;
         cylxcyl_solid := manifold_solid_brep ('cylxcyl_solid',
                                               shell_object) ;
         its_units := length_unit() || si_unit ('milli', 'metre') ;

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

         ebsr := elementary_brep_shape_representation
                 ('ebsr', [cylxcyl_solid], its_context );
   END_REALIZATION;
END_TEST_CASE;
(*
```

E.3.4.4 Критерии решения постпроцессора

EB21 — значения по умолчанию атрибута **axis** должны быть заданы правильно.

EB23 — значения по умолчанию атрибута **ref_direction** должны быть заданы правильно.

EB35 — все точки полилинии должны принадлежать обоим объектам **cylindrical_surface** с допустимым отклонением менее чем 0,000001.

E.3.5 Контрольный пример eb5

В контрольном примере eb5 определены грани, необходимые для определения сплошных тел, полученных в результате пересечения наклонных плоскостей с конусом. Кривые, ограничивающие грани, могут быть эллипсами, гиперболами, параболами, дугами окружностей и сегментами прямых линий. Определение оболочек обеспечивается контекстом объекта **cone_faces** с использованием исходных параметров.

E.3.5.1 Цели тестирования

Ниже перечислены цели тестирования, охватываемые данным контрольным примером.

EB2 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_context** с атрибутом **items**, представленным множеством, состоящим более чем из одного объекта **manifold_solid_brep**.

EB15 — проверка объекта **face** как объекта **face_surface** с атрибутом **bounds**, представленным множеством, состоящим, по меньшей мере, из двух объектов **face_bound**, включая один объект **vertex_loop**.

EB26 — проверка объекта **elementary_surface** как объекта **conical_surface**.

EB40 — проверка объекта **conic** как объекта **ellipse**.

EB41 — проверка объекта **conic** как объекта **hyperbola**.

EB42 — проверка объекта **conic** как объекта **parabola**.

E.3.5.2 Спецификация входа в препроцессор

Создается объект **elementary_brep_shape_representation**, состоящий из двух объектов **manifold_solid_brep**. Объекты **manifold_solid_brep** должны иметь форму конусов, ограниченных наклонными плоскостями. Плоскости выбираются так, чтобы линии пересечения имели форму эллипса, параболы, гиперболы и дуг окружности. Первый конус имеет вершину, образованную объектом **vertex_loop**, и эллиптическое основание. У второго конуса вершина образована эллиптической кривой, как и его основание. Соответствующее множество размеров определено в приведенной ниже спецификации на языке EXPRESS-I.

E.3.5.3 Спецификация входа в постпроцессор

Примечание — Для определения граней и всей геометрии и топологии B-rep модели используется контекст объекта **cone_shell** с параметрами, заданными по умолчанию.

```
*)
TEST_CASE example_ebrep_5; WITH aic_elementary_brep;

  REALIZATION

  LOCAL
    shell1, shell2 : closed_shell ;
    cone1, cone2 : manifold_solid_brep ;
    ebsr : elementary_brep_shape_representation ;
    angle_u, len_u, angle_c_u : named_unit ;
    ang_m_wu : plane_angle_measure_with_unit ;
    its_context : geometric_representation_context ;
  END_LOCAL;

    CALL cone_shell ; -- используются значения по умолчанию
      IMPORT(shell1 := @vconeshell ,
             shell2 := @con4fshell ; ) ;
    END_CALL;
      cone1 := manifold_solid_brep ('cone1', shell1) ;
      cone2 := manifold_solid_brep ('cone2', shell2) ;

      angle_c_u := plane_angle_unit() || si_unit ( , 'radian' ) ;
      ang_m_wu := plane_angle_measure_with_unit(.017453293, angle_c_u),
      angle_u := plane_angle_unit() ||
                   conversion_based_unit('degree', ang_m_wu ) ;
      len_u := length_unit() || si_unit ('milli', 'metre') ;

      its_context := geometric_representation_context
                       ('context_1', 'context_for_cones', 3) ||
                     global_unit_assigned_context ([len_u, angle_u]) ;
```

```
        ebsr := elementary_brep_shape_representation
                ('ebsr',[cone1, cone2], its_context );
    END_REALIZATION;
END_TEST_CASE;
(*
```

E.3.5.4 Критерии решения постпроцессора

EB2 — две B-rep модели должны точно соприкасаться поверхностью общей грани, две B-rep модели не должны пересекаться.

EB15 — объект **face** с атрибутом **face_bound**, представленным как объект **vertex_loop**, должен быть правильно обработан.

EB26 — объекты **conical_surface** должны быть правильно обрезаны ограничивающими ребрами.

EB40 – все точки на эллипсе должны быть расположены точно на обоих объектах **conical_surface** и на объекте **plane**, пересекающем их.

EB41 — все точки гиперболических ребер должны одновременно принадлежать объектам **conical_surface** и пересекающему объекту **plane**, а объект **hyperbola** должен быть правильно обрезан посредством объектов **vertex_point**.

EB42 — все точки параболических ребер должны одновременно принадлежать объектам **conical_surface** и пересекающему объекту **plane**, а объект **parabola** должен быть правильно обрезан посредством объектов **vertex_point**.

E.3.6 Контрольный пример eb6

Контрольный пример eb6 разработан для проверки использования объектов **mapped_item** при создании простой сборки из элементарных B-rep моделей. Он также обеспечивает проверку непротиворечивости объектов **geometric_representation_context** для разных координатных пространств. Данный тест использует контекст объекта **cylinder_union_polyline** для определения геометрии и топологии.

E.3.6.1 Область охвата целей тестирования

Ниже перечислены цели тестирования, охватываемые данным контрольным примером:

EB3 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_ representation_context** с атрибутом **items**, представленным как объект **mapped_item**.

EB4 — проверка объекта **elementary_brep_shape_representation** с атрибутом **context**, представленным как объект **geometric_representation_context** с двумя или более элементами, представленными как объекты **manifold_solid_brep**, или **mapped_item**, или **axis2_placement_3d**. При этом по меньшей мере один из них должен быть объектом **axis2_placement_3d**.

E.3.6.2 Спецификация входа в препроцессор

Создается основной объект **elementary_brep_shape_representation**, состоящий из объекта **manifold_solid_brep** в форме двух пересекающихся цилиндров, определенных в контрольном примере eb4. Затем определяется объект **mapped_item** как поступательно перемещенная и повернутая копия данного представления. После этого определяются два следующих объекта **elementary_brep_shape_representation**; один, определенный в том же контексте, состоит только из объекта **mapped_item**, а другой, определенный в другом контексте, содержит исходные объекты **manifold_solid_brep**, **mapped_item** и **axis2_placement_3d**. Полные сведения о размерах и отображении определены в приведенной ниже спецификации на языке EXPRESS-I.

E.3.6.3 Спецификация входа в постпроцессор

Примечания

1 В приведенной ниже спецификации два разных объекта **geometric_representation_context** созданы для геометрических определений.

2 Для определения граней и всей геометрии и топологии B-rep модели используется контекст объекта **cylinder_union_polyline** с параметрами, заданными по умолчанию.

3 Объект **ebsr1** должен быть повернутой и поступательно перемещенной копией объекта **ebsr**.

4 Объект **ebsrass** должен быть эквивалентен двум копиям объекта **ebsr**, «склеенным» вместе.

```
*)

TEST_CASE example_ebrep_6; WITH aic_elementary_brep;

  REALIZATION

  LOCAL
    Origin : cartesian_point ;
    pos_z, neg_y : direction ;
    refaxes, topaxes, baseaxes : axis_placement_3d;
```

```
      shell_object : closed_shell ,
      cylxcyl : manifold_solid_brep ,
      ebsr, ebsr1, ebsrass : elementary_brep_shape_representation ;
      grc1, grc2 : geometric_representation_context ;
      transrot1, trans2 : mapped_item ;
      mapping1, mapping2 :representation_map ;
   END_LOCAL;

      CALL cylinder_union_polyline ; -- используются значения по умолчанию
         IMPORT (shell_object := @cycshell;
                  origin := @origin;
                  baseaxes := @ ab; refaxes := @ar; topaxes := @at, );
      END_CALL;

cylxcyl := manifold_solid_brep ('cylxcyl', shell_object) ;

grc1 = geometric_representation_context ('ctx1',
                                 'context' for cylinder union', 3) ;

grc2 = geometric_representation_context ('ctx2',
                           'context for rotated cylinder union', 3) ,

ebsr := elementary_brep_shape_representation ('ebsr', [cylxcyl], grc1);

mapping1 := representation_map (baseaxes, ebsr );
transrot1 := mapped_item  ('transrot1', mapping1, refaxes ).

(* Определить представление только с использованием transrot1 *)
ebsr1 := elementary_brep_shape_representation ('ebsr1',
                                                [transrot1], grc1) ;
(* Определить представление, являющееся сборкой пересекающихся цилиндров и отображенной (перемещенной)
копии.
*)

mapping2 := representation_map (baseaxes, ebsr );
trans2 := mapped_item ('trans2', mapping2, topaxes );

ebsrass := elementary_brep_shape_representation
          ('ebsrass', [ebsr1, ebsrot2, baseaxes], grc2 ) ;

   END_REALIZATION;
END_TEST_CASE;
(*
```

E.3.6.4 Критерии решения постпроцессора

EB3 — после обработки объект **mapped_item** должен быть правильно интерпретирован, результатом является повернутая и поступательно перемещенная копия исходной B-rep модели.

EB4 — объект **elementary_brep_shape_representation**, содержащий объект **mapped_item**, B-rep модель и объект **axis2_placement_3d** должны быть правильно интерпретированы, результатом является исходная B-rep модель и повернутая и поступательно перемещенная копия исходной B-rep модели, которая касается грани.

EB3 и EB4 — два разных объекта **elementary_brep_shape_representation** не должны быть пространственно связаны.

E.3.7 Контрольный пример eb7

Контрольный пример eb7 разработан для проверки использования объектов **mapped_item** совместно с объектом **cartesian_transformation_operator** при создании простой сборки из многогранных B-rep моделей. Проверяется использование коэффициента масштабирования. В данном тесте используется контекст объекта **cylinder_sphere_shell** для определения геометрии и топологии.

E.3.7.1 Цели тестирования

Ниже перечислены цели тестирования, охватываемые данным контрольным примером:

EB44 — проверка объекта **mapped_item** с атрибутом **mapping_target**, представленным как объект **cartesian_transformation_operator_3d**.

EB45 — проверка объекта **cartesian_transformation_operator** как объекта **cartesian_transformation_operator_3d** с масштабом, представленным значением типа REAL, не равным 1,0.

E.3.7.2 Спецификация входа в препроцессор

Создается объект **elementary_brep_shape_representation**, состоящий из единственного объекта **manifold_solid_brep**. B-rep модель должна иметь форму сплошного цилиндра с полусферическим основанием и плоским наклонным верхом. Центр полусферы находится в начале координат, а ось цилиндра должна располагаться вдоль оси Z. Затем данное представление используется совместно с объектами **mapped_item** и **cartesian_transformation_operator** для создания в том же контексте объекта **representation_context** представления, состоящего из повернутой, поступательно перемещенной и масштабированной (но не с коэффициентом 1,0) копии исходного представления и исходной B-rep модели. Поступательное перемещение и значение коэффициента масштабирования выбираются так, чтобы цилиндры из двух B-rep моделей соприкасались, но не пересекались.

E.3.7.3 Спецификация входа в постпроцессор

Примечание — Для определения граней и всей геометрии и топологии B-rep модели используется контекст объекта **cylinder_sphere_shell** с параметрами, заданными по умолчанию. Объект **csrans** должен быть повернутой, масштабированной и поступательно перемещенной копией объекта **cysp_solid**.

```
*)

TEST_CASE example_ebrep_7; WITH elementary_brep_aic;

REALIZATION

   LOCAL
      radius : length_measure ;
      origin, neworigin : cartesian_point ;
      pos_x, pos_y, pos_z, neg_z . direction ;
      oldaxes : axis_placement_3d;
      transform . cartesian_transformation_operator_3d;
      shell_object : closed_shell ;
      cysp_solid : manifold_solid_brep ;
      ebsr, ebsrass : elementary_brep_shape_representation ;
      its_units : named_unit ;
      grc1, grc2 : representation_context ;
      csrans . mapped_item ;
      mapping1 : representation_map ;
   END_LOCAL;
    CALL cylinder_sphere_shell ; -- используются значения по умолчанию
        IMPORT (shell_object : = @cyspshell;
                        pos_x : = @pos_x ;
                        pos_y : = @pos_y ;
                        pos_z : = @pos_z ;
                        origin : = @origin ;
                        radius : = @rad , ) ,
   END_CALL;

cysp_solid := manifold_solid_brep ('cysp_solid', shell_object) ,

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

grc1 = geometric_representation_context ('ctx1',
                                'context for cysp_solid', 3) ||
    global_unit_assigned_context ( [its_units] ) ;

grc2 = geometric_representation_context ('ctx2',
       'context for assembly', 3) ||
       global_unit_assigned_context ( [its_units] ) ;

(* Определяются axis_placement и cartesian_transformation_operator
 для использования при отображении *)
neworigin := cartesian_point ([1.8*radius, 0.0, 0.0] );
```

```
neg_z := direction ([0.0, 0.0, -1.0] );

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

transform := cartesian_transformation_operator_3d ('transform',
                                    pos_x, neg_z, neworigin, 0.8, pos_y );

ebsr := elementary_brep_shape_representation ('ebsr',
                                        [cysp_solid, oldaxes], grc1 ) ;

mapping1 := representation_map (oldaxes, ebsr ),
(* Объект cysptrans является 80-процентной копией исходного объекта,
    повернутой вокруг оси X и смещенной вдоль этой оси *)
cstrans := mapped_item ('cstrans', mapping1, transform ),

(* Определяется представление, являющееся сборкой исходной B-rep
    модели и ее преобразованной копии (масштабированной, смещенной и
    повернутой). *)

ebsrass := elementary_brep_shape_representation
        ('ebsrass', [cysp_solid, cstrans], grc2 ) ,

END_REALIZATION;
END_TEST_CASE;
(*
```

E.3.7.4 Критерии решения постпроцессора

EB44 — после обработки, сплошная B-rep модель, определенная посредством объекта **mapped_item**, должна быть корректно масштабирована и размещена.

EB45 — после обработки объект **absrass** должен состоять из двух сплошных B-rep моделей цилиндрической формы с полусферическим основанием, которые касаются друг друга в плоскости x = 25. Одна из них является копией масштаба 4/5 другой модели, полученной после поступательного перемещения и поворота.

E.4 Контексты, определенные для контрольных примеров элементарных B-rep моделей

Приведенные ниже контексты на языке EXPRESS-I использованы в контрольных примерах раздела E.3.

E.4.1 Контекст объекта cylinder_sphere_shell

Данный контекст описывает грани, необходимые для определения простого объекта **closed_shell** цилиндрической формы с полусферическим основанием, имеющим центр, расположенный в точке (orc,orc,orc), радиус rad и осевую высоту к наклонной грани h.

Все границы определены посредством объектов **edge_loop** и **conic**.

```
*)
CONTEXT cylinder_sphere_shell ;
   WITH aic_elementary_brep;
PARAMETER
orc              : length_measure := 0.0;
h                : length_measure := 100.0;
rad              . length_measure := 25.0,
majrad           : length_measure := rad*rt2;
origin : cartesian_point := cartesian_point ('origin', [orc, orc, orc]),
ctop   : cartesian_point := cartesian_point ('ctop', [orc, orc,
                                                    orc+h]);
pos_x  : direction := direction ('pos_x', [1, 0, 0]);
pos_y  : direction := direction ('pos_y', [0, 1, 0]);
pos_z  : direction := direction ('pos_z', [0, 0, 1]);
dslope : direction := direction ('dslope', [1, 0, -1]);
dperp  : direction := direction ('dperp', [1, 0, 1]);

a1     : axis2_placement_3d := axis2_placement_3d ('a1', origin,
                                                    pos_z, pos_x) ;
a2     : axis2_placement_3d := axis2_placement_3d ('a2', ctop,
                                                    dperp, dslope) ,
```

```
pl      : plane := plane ('pl', a2) ;
cyl     : cylindrical_surface := cylindrical_surface ('cyl', a1, rad);
sphere : spherical_surface := spherical_surface ('sphere', a1, rad);
circ    : circle := circle ('circ', a1, rad);
elli    : ellipse := ellipse('elli', a2, majrad, rad);

cpoint  : cartesian_point := cartesian_point ('cpoint', [(orc + rad),
                                                         orc, orc]);
epoint  : cartesian_point :=
            cartesian_point ('epoint', [(orc + rad), orc,
                                        (orc + h - rad)]);
vertc   : vertex_point := vertex_point ('vertc', cpoint);
verte   : vertex_point := vertex_point ('verte', epoint);

edge1  : edge_curve := edge_curve ('edge1', vertc, vertc, circ, TRUE);
edge2  : edge_curve := edge_curve ('edge2', verte, verte, elli, TRUE);
oe1    : oriented_edge := oriented_edge ('oe1', edge1, TRUE);
oe2    : oriented_edge := oriented_edge ('oe2', edge2, TRUE);
loopc  : edge_loop := edge_loop ('loopc', ['oe1']);
loope  : edge_loop := edge_loop ('loope', [oe2]);

bc      : face_outer_bound := face_outer_bound ('bc', loopc, FALSE) ;
be      : face_outer_bound := face_outer_bound ('be', loope, TRUE) ;
bcylbot : face_bound := face_bound ('bcylbot', loopc, TRUE);
bcyltop : face_bound := face_bound ('bcyltop', loope, FALSE);

curved_face : face_surface := face_surface('curved_face', [bcylbot,
                                        bcyltop], cyl, TRUE);
top_face : face_surface := face_surface ('top_face', [be], pl, TRUE);
bottom_face : face_surface := face_surface ('bottom_face', [bc],
                                                    sphere, TRUE);
END_PARAMETER;

SCHEMA_DATAcyl_sph_shell_ctxt;
CONSTANT
    rt2 == sqrt(2.0);
    rt3 == sqrt(3.0);
END_CONSTANT;
cfs = connected_face_set {SUBOF(@tri);
                cfs_faces -> {@curved_face, @top_face, @bottom_face);
                                SUPOF(@cyspshell);} ;

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

ri = representation_item {name -> 'cyspshell', SUPOF(@tri), };
cyspshell = closed_shell {SUBOF(@cfs); };

END_SCHEMA_DATA;

END_CONTEXT ;
(*
```

E.4.2 Контекст объекта cone_shell

Данный контекст описывает грани, необходимые для определения замкнутых оболочек конической формы с круглыми, эллиптическими, гиперболическими или параболическими гранями. Конус имеет вершину с координатами (orc,orc,orc), половинный угол при вершине 30° и ось, параллельную оси z. Перпендикуляр к каждой плоской грани ортогонален к направлению оси y и расположен под фиксированным углом.

Примечание — Объекты **plane_angle_unit** должны быть определены в градусах.

Размеры результирующей оболочки должны регулироваться изменением значений расстояний dc, de, dh, dp от вершины конуса до точек пересечения с осью конуса плоскостей окружности, гиперболы, параболы соответственно.

Две смежные оболочки могут быть определены простой конической оболочкой с эллиптическим основанием и более сложной конической формой с плоскими гранями эллиптической вершины, круглого основания и гиперболической и параболической сторонами. У основания есть два прямых ребра, параллельных оси y.

Все границы определены посредством объектов **edge_loop** с использованием объектов **line** или **conic** или посредством объекта **vertex_loop**.

```
*)
CONTEXT cone_shell;
    WITH aic_elementary_brep;

PARAMETER
  orc           : length_measure := 0.0;
  dc            : length_measure := 200.0;
  de            : length_measure := 20.0;
(* Примечание. dp и dh должны быть больше чем расстояние dc
   до базовой окружности. Во избежание пересечений с верхним эллипсом,
   должны выполняться соотношения dh > 7.47 de, и dp > 1.27 de *)
 dh             : length_measure := 300.0;
 dp             : length_measure := 250.0;
 emaj           : length_measure := de*(rt3/rt2);
 emin           : length_measure := de*(rt2/2.0);
 haxis          : length_measure := 0.5*dh*rt3/rt2;
 himag          : length_measure := dh*sqrt((rt3 - 1.0)/8.0);
 yh             : length_measure := sqrt((rt3 - 1.0)*((2.5-1.5*rt3)*dh*dh +
                      dc*dh*(3.0*rt3 - 5.0) + 4.0*dc*dc*(2.0 - rt3)/3.0));
(* размещение указано для конуса, базовой окружности, эллипса, параболы и гиперболы, соответственно:
*)
origin          : cartesian_point := cartesian_point('origin',[orc, orc, orc]);
  cbase         : cartesian_point := cartesian_point ('cbase',
                                                [orc, orc, orc - dc]);
  ecent         : cartesian_point := cartesian_point('ecent',
                                        [orc+0.5*de, orc, orc-1.5*de]);
  hcent         : cartesian_point := cartesian_point('hcent',
                  [orc+dh*(rt3 + 1.0)/8.0, orc, orc + dh*0.375*(rt3 - 1.0)]);
  ppoint        : cartesian_point := cartesian_point('ppoint',
                                    [(orc - 0.5*dp/rt3), orc,(orc - 0.5*dp)]),
  epoint        : cartesian_point := cartesian_point('epoint',
                    [orc + 0.5*de*(rt3 + 1.0), orc, orc - 0.5*de*(3.0 + rt3)]);
(* точки пересечения кривых второго порядка с плоскостью основания: *)
  ppb1          : cartesian_point := cartesian_point('ppb1', [orc +
      (dc - dp)/rt3, orc - (0.5*dp/rt3)*sqrt(8.0*dc/dp - dp), orc - dc]);
  ppb2          : cartesian_point := cartesian_point('ppb2',[orc + (dc - dp)/rt3,
                    orc + (0.5*dp/rt3)*sqrt(8.0*dc/dp - dp), orc - dc]);
  phb1          : cartesian_point := cartesian_point('phb1',
                        [orc + (dp - dc)*(2.0 - rt3), orc - yh, orc - dc]),
  phb2          : cartesian_point := cartesian_point('phb2',
                        [orc + (dp - dc)*(2.0 - rt3), orc + yh, orc - dc]);

  pos_x         : direction := direction ('pos_x', [1, 0, 0]);
  pos_y         : direction := direction ('pos_y', [0, 1, 0]);
  vec_y         : vector := vector ('vec_y', pos_y, 1.0);
  pos_z         : direction := direction ('pos_z', [0, 0, 1]);
  denorm        : direction := direction ('denorm', [1.0, 0, 1.0]);
  dhnorm        : direction := direction ('dhnorm',
                            [(rt3 + 1.0) , 0, -(rt3 - 1.0)]);
  dpnorm        : direction := direction ('dpnorm', [rt3, 0, 1]);
(* плоскости эллипса, параболы, гиперболы расположены под углами 45°,
   30° и 15° относительно оси конуса *)
  dir_e         : direction := direction ('dir_e', [1.0, 0, -1.0]);
  dir_h         : direction := direction ('dir_h',
                                        [-(rt3 - 1.0) , 0, -(rt3 + 1.0)]);
  dir_p         : direction := direction ('dir_p', [1, 0, -rt3]);
```

```
a1          : axis2_placement_3d := axis2_placement_3d ('a1', origin,
                                                        pos_z, pos_x);
ac          : axis2_placement_3d := axis2_placement_3d ('ac', cbase, pos_z,
                                                        pos_x);
ae          : axis2_placement_3d := axis2_placement_3d ('ae', ecent, denorm,
                                                        dir_e);
ah          : axis2_placement_3d := axis2_placement_3d ('ah', hcent, dhnorm,
                                                        dir_h);
ap          : axis2_placement_3d := axis2_placement_3d ('ap', ppoint, dpnorm,
                                                        dir_p);
a2          : axis2_placement_3d := axis2_placement_3d ('a2', cbase, pos_z,
                                                        pos_x);

ple         : plane := plane ('ple', ae);
plc         : plane := plane ('plc', ac);
plh         : plane := plane ('plh', ah);
plp         : plane := plane ('plp', ap);
cone        : conical_surface := conical_surface ('cone', a1, 0.0, 30.0);

circ        : circle := circle ('circ', ac, (dc/rt3);
elli        : ellipse := ellipse('elli', ae, emaj, emin);
hyp         : hyperbola := hyperbola('hyp', ah, haxis, himag);
parab       : parabola := parabola('parab', ap, 0.25*dp/rt3);
linpb       : line := line('linpb', ppb1, vec_y);
linph       : line := line('linph', phb1, vec_y);

vertorc       : vertex_point := vertex_point ('vertorc', origin);
verte         : vertex_point := vertex_point ('verte', epoint);
vertpb1       : vertex_point := vertex_point ('vertpb1', ppb1);
vertpb2       : vertex_point := vertex_point ('vertpb', ppb2);
verthb1       : vertex_point := vertex_point ('verthb1', phb1);
verthb2       : vertex_point := vertex_point ('verthb2', phb2);

edge1    : edge_curve := edge_curve ('edge1', verte, verte, elli, TRUE);
edge2    : edge_curve := edge_curve ('edge2', vertpb1, vertpb2,
                                                        parab, TRUE);
edge3    : edge_curve := edge_curve ('edge3', verthb1, verthb2, hyp,TRUE);
edgeb1   : edge_curve := edge_curve ('edgeb1', vertpb1, verthb1,
                                                        circ, TRUE);
edgeb2   : edge_curve :=edge_curve ('edgeb', verthb1, verthb2,
                                                        linph, TRUE);
edgeb3   : edge_curve := edge_curve ('edgeb3', verthb2, vertpb2,
                                                        circ, TRUE);
edgeb4   : edge_curve := edge_curve ('edgeb4', vertpb2, vertpb1,
                                                        linpb, FALSE);
oe1         : oriented_edge := oriented_edge ('oe1', edge1, TRUE);
oe2t        : oriented_edge := oriented_edge ('oe2t', edge2, TRUE);
oe2f        : oriented_edge := oriented_edge ('oe2f', edge2, FALSE);
oe3t        : oriented_edge := oriented_edge ('oe3t', edge3, TRUE);
oe3f        : oriented_edge := oriented_edge ('oe3f', edge3, FALSE);
oeb1t       : oriented_edge := oriented_edge ('oeb1t', edgeb1, TRUE);
oeb1f       : oriented_edge := oriented_edge ('oeb1f', edgeb1, FALSE);
oeb2t       : oriented_edge := oriented_edge ('oeb2t', edgeb2, TRUE);
oeb2f       : oriented_edge := oriented_edge ('oeb2f', edgeb2, FALSE);
oeb3t       : oriented_edge := oriented_edge ('oeb3t', edgeb3, TRUE);
oeb3f       : oriented_edge := oriented_edge ('oeb3f', edgeb3, FALSE);
oeb4t       : oriented_edge := oriented_edge ('oeb4t', edgeb4, TRUE);
oeb4f       : oriented_edge := oriented_edge ('oeb4f', edgeb4, FALSE);

loope       : edge_loop := edge_loop ('loope', [oe1]);
looppar     : edge_loop := edge_loop ('looppar', [oeb4t, oe2t]);
loophyp     : edge_loop := edge_loop ('loophyp', [oeb2t, oe3f]);
```

```
  loopbase    : edge_loop := edge_loop ('loopbase', [oeb4f, oeb3f,
                                                     oeb2f, oeb1f]);
  loopcone    : edge_loop := edge_loop ('loopcone', [oe2f, oeb1t,
                                                     oe3t, oeb3t]);
  apexloop    : vertex_loop := vertex_loop ('apexloop', vertorc) ;

  bc1       : face_bound := face_bound ('bc1', loopcone, TRUE) ;
  bc2       : face_bound := face_bound ('bc2', loope, FALSE) ;
  be_t      : face_outer_bound := face_outer_bound ('be_t', loope, TRUE);
  be_f      : face_bound := face_bound ('be_f', loope, FALSE) ;
  bpar      : face_outer_bound := face_outer_bound ('bpar', looppar, TRUE),
  bhyp      : face_outer_bound := face_outer_bound ('bhyp', loophyp, TRUE);
  bbase     : face_outer_bound := face_outer_bound('bbase', loopbase, TRUE),
  bcone     : face_bound := face_bound ('bcone', loopcone, TRUE) ;
  vbound : face_bound := face_bound ('vbound', apexloop, TRUE);
  (* Четыре плоские грани конуса *)
  curved_face : face_surface := face_surface ('curved_face', [bcone,
                                    be_f], cone, TRUE);
  tope_face     : face_surface := face_surface ('tope_face', [be_t],
                                                    ple, TRUE),
  bottomc_face : face_surface := face_surface
                        ('bottomc_face', [bbase], plc, FALSE);
  par_face      : face_surface :=
                        face_surface ('par_face', [bpar], plp, TRUE);
  hyp_face      : face_surface :=
                        face_surface ('hyp_face', [bhyp], plh, TRUE);

(* Грани конуса с эллиптическим основанием и верхним контуром *)
  top_face      : face_surface := face_surface
                        ('top_face', [be_t, vbound], cone, TRUE);
 bottome_face : face_surface := face_surface ('bottome_face',
                                                [be_f], ple, FALSE);
END_PARAMETER;

SCHEMA_DATA cone_shell_ctxt;

CONSTANT
    rt2 == sqrt(2.0);
    rt3 == sqrt(3.0);
END_CONSTANT;

ri1 = representation_item (name -> 'vconeshell'; SUPOF(@tri1),) ;

tri1 = topological_representation_item (SUBOF(@ri1); SUPOF(@cfs1);) ;

cfs1 = connected_face_set (SUBOF(@tri);
                        cfs_faces -> (@top_face, @bottome_face);
                        SUPOF(@vconeshell); ) ;

ri2 = representation_item (name -> 'con4fshell', SUPOF(@tri2),) ;

tri2 = topological_representation_item (SUBOF(@ri2); SUPOF(@cfs2);) ;

cfs2 = connected_face_set (SUBOF(@tri2);
        cfs_faces -> (@tope_face, @bottomc_face, curved_face, par_face,
                hyp_face); SUPOF(@con4fshell); );

vconeshell = closed_shell (SUBOF(@cfs1); );

con4fshell = closed_shell (SUBOF(@cfs2); ),

END_SCHEMA_DATA;

END_CONTEXT;

(*
```

E.4.3 Контекст объекта toroidal_segment

Данный контекст описывает грани, необходимые для определения сегмента тора, ограниченного плоскостями. Объектами **edge_curve** являются объекты **line**, **circular arcpolyline**.

Центр тора расположен в начале координат, а его центральной осью является ось z; максимальный и минимальный радиусы тора равны 100 и 20. Ограничивающие плоскости заданы координатами z = 0, x = 0 и x = 50. Все точки полилиний расположены на поверхности тора с допустимым отклонением менее чем 10E(-6).

Все границы определены посредством объектов **edge_loop**. Основные размеры не должны изменяться.

```
*)
CONTEXT toroidal_segment;
    WITH aic_elementary_brep;
PARAMETER
  orc      : length_measure := 0.0;
  rad1     : length_measure := 100.0;
  rad2     : length_measure := 20.0;
  rci      : length_measure := 80.0;
  rco      : length_measure := 120.0;
  origin   : cartesian_point := cartesian_point ('origin', [orc, orc,
                                                    orc]);
  p1       : cartesian_point := cartesian_point ('p1', [50.0, 62.44998,
                                                     0.0]);
  pcleft   : cartesian_point := cartesian_point ('pcleft', [0.0, 100.0,
                                                    0.0]);
  pbleft   : cartesian_point := cartesian_point ('pbleft', [0.0, 80.0,
                                                    0.0]);
  ptleft   : cartesian_point := cartesian_point ('ptleft', [0.0, 120.0,
                                                    0.0]);

  pos_x  : direction := direction ('pos_x', [1, 0, 0]);
  pos_y  : direction := direction ('pos_y', [0, 1, 0]);
  pos_z  : direction := direction ('pos_z', [0, 0, 1]);
  neg_x  : direction := direction ('neg_x', [-1, 0, 0]);
  vec_y  : vector := vector ('vec_y', pos_y, 1.0);

  a1       : axis2_placement_3d := axis2_placement_3d ('a1', origin,
                                                    pos_z, pos_x);
  a2       : axis2_placement_3d := axis2_placement_3d ('a2', pcleft,
                                                    neg_x, pos_y);
  a3       : axis2_placement_3d := axis2_placement_3d ('a3', p1,
                                                    pos_x, pos_z);

  base   : plane := plane ('base', a1);
  pleft  : plane := plane ('pleft', a2);
  pright : plane := plane ('pright', a3);
  torus  : toroidal_surface := toroidal_surface ('torus', a1, rad1,
                                                    rad2);

  circin   : circle := circle ('circin', a1, rci);
  circout  : circle := circle ('circout', a1, rco);
  circleft : circle := circle ('circleft', a2, rad2);

  p2       : cartesian_point :=
                cartesian_point ('p2', [50.0, 62.633918, 2.392932]);
  p3       : cartesian_point :=
                cartesian_point ('p3', [50.0, 64.92632, 8.609]);
  p4       : cartesian_point :=
                cartesian_point ('p4', [50.0, 67.325057, 11.8123625]);
  p5       : cartesian_point :=
                cartesian_point ('p5', [50.0, 69.839261, 14.1766914]);
  p6       : cartesian_point :=
                cartesian_point ('p6', [50.0, 72.479126, 16.03916]);
  p7       : cartesian_point :=
                cartesian_point ('p7', [50.0, 75.25605, 17.518988]);
```

```
p8     : cartesian_point :=
            cartesian_point ('p8', [50.0, 78.182821, 18.660529]),
p9     : cartesian_point :=
            cartesian_point ('p9', [50.0, 81.27384, 19.469096]);
p10    : cartesian_point :=
            cartesian_point ('p10', [50.0, 84.5453828, 19.920975]);
p11    : cartesian_point :=
            cartesian_point ('p11', [50.0, 88.0159173, 19.9623578]);
p12    : cartesian_point :=
            cartesian_point ('p12', [50.0, 91.706487, 19.498351]);
p13    : cartesian_point :=
            cartesian_point ('p13', [50.0, 95.6411789, 18.343994]);
p14    : cartesian_point :=
            cartesian_point ('p14', [50.0, 99.8476928, 16.24428]);
p15    : cartesian_point :=
            cartesian_point ('p15', [50.0, 104.3580503, 12.3673625]);
p16    : cartesian_point :=
            cartesian_point ('p16', [50.0, 107.225346, 8.046179]);
p17    : cartesian_point :=
            cartesian_point ('p17', [50.0, 109.008344, 1.692583]);
p18    : cartesian_point :=
            cartesian_point ('p18', [50.0, 109.0871212, 0.0]);

poly   : polyline := polyline('poly', [p1, p2, p3,
                                       p4, p5, p6, p7, p8, p9,
                  p10, p11, p12, p13, p14, p15, p16, p17, p18]),
l1     : line := line ('l1', p1, vec_y);
l2     : line := line ('l2', pcleft, vec_y),

v1     : vertex_point := vertex_point ('v1', p1);
v2     : vertex_point := vertex_point ('v2', p18);
v3     : vertex_point := vertex_point ('v3', pbleft);
v4     : vertex_point := vertex_point ('v4', ptleft);

edgeb1 : edge_curve := edge_curve ('edgeb1', v1, v2, l1, TRUE);
edget1 : edge_curve := edge_curve ('edget1', v1, v2, poly, TRUE);
edgeb2 : edge_curve := edge_curve ('edgeb2', v1, v3, circin, TRUE);
edgeb3 : edge_curve := edge_curve ('edgeb3', v2, v4, circout, TRUE);
edgeb4 : edge_curve := edge_curve ('edgeb4', v3, v4, l2, TRUE);
edget2 : edge_curve := edge_curve ('edget2', v3, v4, circleft, TRUE);
oeb1t  : oriented_edge := oriented_edge ('oeb1t', edgeb1, TRUE);
oeb1f  : oriented_edge := oriented_edge ('oeb1f', edgeb1, FALSE);
oeb2t  : oriented_edge := oriented_edge ('oeb2t', edgeb2, TRUE);
oeb2f  : oriented_edge := oriented_edge ('oeb2f', edgeb2, FALSE);
oeb3t  : oriented_edge := oriented_edge ('oeb3t', edgeb3, TRUE);
oeb3f  : oriented_edge := oriented_edge ('oeb3f', edgeb3, FALSE);
oeb4t  : oriented_edge := oriented_edge ('oeb4t', edgeb4, TRUE);
oeb4f  : oriented_edge := oriented_edge ('oeb4f', edgeb4, FALSE);
oet1t  : oriented_edge := oriented_edge ('oet1t', edget1, TRUE),
oet1f  : oriented_edge := oriented_edge ('oet1f', edget1, FALSE);
oet2t  : oriented_edge := oriented_edge ('oet2t', edget2, TRUE);
oet2f  : oriented_edge := oriented_edge ('oet2f', edget2, FALSE);

loopb     : edge_loop := edge_loop ('loopb', [oeb4t, oeb3f, oeb1f,
                                              oeb2t]);
loopt     : edge_loop := edge_loop ('loopt', [oeb2f, oet1t, oeb3t,
                                              oet2f]),
loopleft  : edge_loop := edge_loop ('loopleft', [oeb4f, oet2t]);
loopright : edge_loop := edge_loop ('loopright', [oeb1t, oet1f]),

bbase : face_outer_bound := face_outer_bound ('bbase', loopb, TRUE) ;
btop  : face_outer_bound := face_outer_bound ('btop', loopt, TRUE) ;
```

```
  bleft   : face_outer_bound := face_outer_bound ('bleft', loopleft,
                                                  TRUE);
  Bright : face_outer_bound := face_outer_bound ('bright', loopright,
                                                  TRUE);

  curved_face : face_surface := face_surface ('curved_face', [btop],
                                                  torus, TRUE);
  base_face   : face_surface :=
                    face_surface ('base_face', [bbase], base, FALSE);
  left_face    : face_surface :=
                     face_surface ('left_face', [bleft], pleft, TRUE);
  right_face   : face_surface :=
                   face_surface ('right_face', [bright], pright, TRUE);
END_PARAMETER;

SCHEMA_DATA tor_shell_ctxt;

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

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

cfs = connected_face_set {SUBOF(@tri);
    cfs_faces -> (@curved_face, @base_face, @left_face, @right_face);
                        SUPOF(@torshell); },

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

END_SCHEMA_DATA;

END_CONTEXT ;

(*
```

E.4.4 Контекст объекта cylinder_union_polyline

Данный контекст описывает грани, необходимые для определения граней объединения двух цилиндров разных радиусов.

Данный контекст дает пример неплоского объекта **polyline** и объекта **face** с тремя ограничивающими контурами.

Все границы определены посредством объектов **edge_loop**. Основные размеры не должны изменяться.

```
*)
CONTEXT cylinder_union_polyline ;
    WITH aic_elementary_brep;
PARAMETER
  orc   : length_measure := 0.0;
  rad1 : length_measure := 50.0;
  rad2 : length_measure := 20.0;
  l1    : length_measure := 80.0;
  l2    : length_measure := -80.0;

origin : cartesian_point := cartesian_point ('origin',[orc, orc, orc]);
ptop   : cartesian_point := cartesian_point ('ptop', [orc, orc, l1]);
pbase : cartesian_point := cartesian_point ('pbase', [orc, orc, l2]);
pright : cartesian_point := cartesian_point ('pright', [orc, l1, orc]);
pte    : cartesian_point := cartesian_point ('pte', [rad1, orc, l1]);
pbe    : cartesian_point := cartesian_point ('pbe', [rad1, orc, l2]);
pre    : cartesian_point := cartesian_point ('pre', [rad2 , l1, orc]);

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

```
a1     : axis2_placement_3d := axis2_placement_3d ('a1', origin, pos_z,
                                                   pos_x) ;
a2     : axis2_placement_3d := axis2_placement_3d ('a2', origin, pos_y,
                                                   pos_x) ;
at     . axis2_placement_3d := axis2_placement_3d ('at', ptop, ?, ? );
ab     : axis2_placement_3d := axis2_placement_3d ('ab', pbase, ?, ?);
ar     : axis2_placement_3d .= axis2_placement_3d ('ar', pright,
                                                  pos_y, pos_x) .

base   : plane := plane ('base', ab) ;
top    . plane := plane ('top', at) ;
plright : plane .= plane ('plright', ar) ;
cyl1   : cylindrical_surface := cylindrical_surface ('cyl1', a1, rad1);
cyl2   : cylindrical_surface := cylindrical_surface ('cyl2', a2, rad2);

circtop    : circle := circle ('circtop', at , rad1);
circbase   : circle := circle ('circbase', ab , rad1);
circright  : circle := circle ('circright', ar , rad2);

p1     : cartesian_point :=
           cartesian_point ('p1', [0.0, 50.0, 20.0]);
p2     : cartesian_point :=
           cartesian_point ('p2', [3.4729636, 49.8792394, 19.6961551]);
p3     : cartesian_point :=
           cartesian_point ('p3', [6.8404029, 49.529879, 18.793852]);
p4     : cartesian_point :=
           cartesian_point ('p4', [10.0, 48.9897949, 17.3205081]);
p5     : cartesian_point :=
           cartesian_point ('p5', [12.8557522, 48.31904, 15.320889]);
p6     : cartesian_point :=
           cartesian_point ('p6', [15.3208889, 47.5948565, 12.8557522]);
p7     : cartesian_point :=
           cartesian_point ('p7', [17.3205081, 46.904158, 10.0]);
p8     : cartesian_point :=
           cartesian_point ('p8', [18.7938524, 46.3334772, 6.84040287]);
p9     : cartesian_point :=
           cartesian_point ('p9', [19.6961551, 45.95717, 3.4729635]);
p10    : cartesian_point .=
           cartesian_point ('p10', [20.0, 45.8257569, 0.0]);
p11    : cartesian_point .=
           cartesian_point ('p11', [19.6961551, 45.95717, –3.4729635]).
p12    : cartesian_point .=
           cartesian_point ('p12', [18.7938524, 46.3334772, –6.84040287]),
p13    : cartesian_point :=
           cartesian_point ('p13', [17.3205081, 46.904158, –10.0]);
p14    : cartesian_point .=
           cartesian_point ('p14', [15.3208889, 47.5948565, –12.8557522]);
p15    : cartesian_point .=
           cartesian_point ('p15', [12.8557522, 48.31904, –15.320889]),
p16    : cartesian_point .=
           cartesian_point ('p16', [10.0, 48.9897949, –17.3205081]);
p17    : cartesian_point :=
           cartesian_point ('p17', [6.8404029, 49.529879, –18.793852]),
p18    : cartesian_point :=
           cartesian_point ('p18', [3.4729636, 49.8792394, –19.6961551]);
p19    : cartesian_point :=
          cartesian_point ('p19', [0.0, 50.0, –20.0]);
p20    : cartesian_point :=
           cartesian_point ('p20', [–3.4729636, 49.8792394, –19.6961551]);
p21    : cartesian_point :=
           cartesian_point ('p21', [–6.8404029, 49.529879, –18.793852]);
p22    : cartesian_point .=
           cartesian_point ('p22', [–10.0, 48.9897949, –17.3205081]);
```

```
p23  : cartesian_point :=
          cartesian_point ('p23', [–12.8557522, 48.31904, –15.320889]);
p24  : cartesian_point :=
          cartesian_point ('p24', [–15.3208889, 47.5948565, –12.8557522]),
p25  : cartesian_point :=
          cartesian_point ('p25', [–17.3205081, 46.904158, –10.0]);
p26  : cartesian_point :=
          cartesian_point ('p26', [–18.7938524, 46.3334772, –6.84040287]);
p27  : cartesian_point :=
          cartesian_point ('p27', [–19.6961551, 45.95717, –3.4729635]);
p28  : cartesian_point :=
          cartesian_point ('p28', [–20.0, 45.8257569, 0.0]);
p29  : cartesian_point :=
          cartesian_point ('p29', [–19.6961551, 45.95717, 3.4729635]);
p30  : cartesian_point :=
          cartesian_point ('p30', [–18.7938524, 46.3334772, 6.84040287]);
p31  : cartesian_point :=
          cartesian_point ('p31', [–17.3205081, 46.904158, 10.0]);
p32  : cartesian_point :=
          cartesian_point ('p32', [–15.3208889, 47.5948565, 12.8557522]);
p33  : cartesian_point :=
          cartesian_point ('p33', [–12.8557522, 48.31904, 15.320889]);
p34  : cartesian_point :=
          cartesian_point ('p34', [–10.0, 48.9897949, 17.3205081]);
p35  : cartesian_point :=
          cartesian_point ('p35', [–6.8404029, 49.529879, 18.793852]);
p36  : cartesian_point :=
          cartesian_point ('p36', [–3.4729636, 49.8792394, 19.6961551]);

poly  : polyline := polyline ('poly', [p1, p2, p3, p4, p5, p6,
              p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17,
              p18, p19, p20, p21, p22, p23, p24, p25, p26, p27,
              p28, p29, p30, p31, p32, p33, p34, p35, p36, p1]);

v1    : vertex_point := vertex_point ('v1', p1);
v2    : vertex_point := vertex_point ('v2', pte);
v3    : vertex_point := vertex_point ('v3', pbe);
v4    : vertex_point := vertex_point ('v4', pre);

edgem0  : edge_curve := edge_curve ('edgem0', v1, v1, poly, TRUE);
edget1   . edge_curve := edge_curve ('edget1', v2, v2, circtop, TRUE),
edgeb2   : edge_curve := edge_curve ('edgeb2', v3, v3, circbase, TRUE);
edger3   : edge_curve := edge_curve ('edger3', v4, v4, circright, TRUE);

oem0t    : oriented_edge := oriented_edge ('oem0t', edgem0, TRUE);
oem0f    : oriented_edge := oriented_edge ('oem0f', edgem0, FALSE),
oet1t    : oriented_edge := oriented_edge ('oet1t', edget1, TRUE);
oet1f    : oriented_edge . oriented_edge ('oet1f', edget1, FALSE);
oeb2t    . oriented_edge := oriented_edge ('oeb2t', edgeb2, TRUE);
oeb2f    : oriented_edge := oriented_edge ('oeb2f', edgeb2, FALSE);
oer3t    : oriented_edge := oriented_edge ('oer3t', edger3, TRUE);
oer3f    : oriented_edge := oriented_edge ('oer3f', edger3, FALSE);

loopb  . edge_loop := edge_loop ('loopb', [oeb2f]);
loopt  : edge_loop := edge_loop ('loopt', [oet1t]),
loopmidt  : edge_loop := edge_loop ('loopmidt', [oem0t]),
looprt    : edge_loop := edge_loop ('looprt', [oer3t]),
loopbt    : edge_loop := edge_loop ('loopbt', [oeb2t]);
looptf    : edge_loop := edge_loop ('looptf', [oet1f]);
loopmidf  : edge_loop := edge_loop ('loopmidf', [oem0f]),
looprf    : edge_loop := edge_loop ('looprf', [oer3f]);
```

```
bbase    : face_outer_bound := face_outer_bound ('bbase', loopb, TRUE),
btop     : face_outer_bound := face_outer_bound ('btop', loopt, TRUE) ;
bright   : face_outer_bound := face_outer_bound ('bright', looprt, TRUE);
bmidt    : face_bound := face_bound ('bmidt', loopmidt, TRUE) ;
bmidf    : face_bound := face_bound ('bmidf', loopmidf, TRUE) ;
bcy1top : face_bound := face_bound ('bcy1top', looptf, TRUE) ;
bcy1b    : face_bound := face_bound ('bcy1b', loopbt, TRUE) ,
bcy1m    : face_bound := face_bound ('bcy1m', loopmidt, TRUE) ;
bcy2m    : face_bound := face_bound ('bcy2m', loopmidf, TRUE) ;
bcy2r    : face_bound := face_bound ('bcy2r', looprf, TRUE) ;

cyl_face1   : face_surface := face_surface ('cyl_face1',
                                  [bcy1top, bcy1m, bcy1b], cyl1, TRUE);
cyl_face2   : face_surface := face_surface ('cyl_face2',
                                  [bcy2m, bcy2r], cyl2, TRUE);
base_face : face_surface :=
                    face_surface ('base_face', [bbase], base, FALSE);
top_face    : face_surface :=
                face_surface ('top_face', 'top_face', [btop], top, TRUE);
right_face : face_surface :=
                  face_surface ('right_face', [bright], plright, TRUE);
END_PARAMETER,

SCHEMA_DATA cyl_un_poly_ctxt;

ricx = representation_item {name -> 'cxcshell', SUPOF(@tricx);} ;

tricx = topological_representation_item {SUBOF(@ricx); SUPOF(@cfscx);};

cfscx = connected_face_set {SUBOF(@tricx);
        cfs_faces -> (@cyl_face1, @cyl_face2, @base_face, @top_face,
                                  @right_face), SUPOF(@csxshell); } ;

cxcshell = closed_shell {SUBOF(@cfscx);} ;
END_SCHEMA_DATA;

END_CONTEXT,

(*
```

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК 8824-1:1995	IDT	ГОСТ Р ИСО/МЭК 8824-1—2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации
ИСО 10303-1:1994	IDT	ГОСТ Р ИСО 10303-1—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы
ИСО 10303-11:1994	IDT	ГОСТ Р ИСО 10303-11—2000 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS
ИСО/ТО 10303-12:1997	IDT	ГОСТ Р ИСО/ТО 10303-12—2000 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 12. Методы описания. Справочное руководство по языку EXPRESS-I
ИСО 10303-41:1994	IDT	ГОСТ Р ИСО 10303-41—99 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 41. Интегрированные обобщенные ресурсы. Основы описания и поддержки изделий
ИСО 10303-42:1994	—	*
ИСО 10303-43:1994	IDT	ГОСТ Р ИСО 10303-43—2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 43. Интегрированные обобщенные ресурсы. Структуры представлений
ИСО 10303-202:1996	—	*
ИСО 10303-514: 1999	IDT	ГОСТ Р ИСО 10303-514—2007 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 514. Прикладные интерпретированные конструкции. Расширенное граничное представление.

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:
- IDT — идентичные стандарты.

УДК 656.072:681.3:006.354 ОКС 25.040.40 П87 ОКСТУ 4002

Ключевые слова: автоматизация производства, средства автоматизации, интеграция систем автоматизации, промышленные изделия, представление данных, обмен данными, прикладные интерпретированные конструкции, поддержка жизненного цикла изделий, граничное представление

Редактор *В.Н. Копысов*
Технический редактор *Н.С. Гришанова*
Корректор *В.Г. Грошунина*
Компьютерная верстка *И.А. Налейкиной*

Сдано в набор 04.08.2010. Подписано в печать 04.10.2010. Формат 60 × 84 1/8. Бумага офсетная. Гарнитура Ариал.
Печать офсетная. Усл. печ. л. 5,12. Уч.-изд. л. 4,90. Тираж 96 экз. Зак. 780.

ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
www.gostinfo.ru info@gostinfo.ru
Набрано во ФГУП «СТАНДАРТИНФОРМ» на ПЭВМ.
Отпечатано в филиале ФГУП «СТАНДАРТИНФОРМ» — тип. «Московский печатник», 105062 Москва, Лялин пер., 6.