INTERNATIONAL STANDARD

ISO 10303-46

First edition
1994-12-15

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

Part 46:
Integrated generic resources: Visual presentation

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

Partie 46: Ressources génériques intégrées: Présentation visuelle

Reference number
ISO 10303-46:1994(E)

Contents

International Organization for Standardization
Case Postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

Figures

Tables

Foreword

The International Organization for Standardization (ISO) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

International Standard ISO 10303–46 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Subcommittee SC4, *Industrial data and global manufacturing programming languages.*

ISO 10303 consists of the following parts under the general title *Industrial automation systems and integration – Product data representation and exchange:*

- Part 1, Overview and fundamental principles;
- Part 11, Description methods: The *EXPRESS* language reference manual;
- Part 21, Implementation methods: Clear text encoding of the exchange structure;
- Part 22, Implementation methods: Standard data access interface specification;
- Part 31, Conformance testing methodology and framework: General concepts;
- Part 32, Conformance testing methodology and framework: Requirements on testing laboratories and clients;
- Part 41, Integrated generic resources: Fundamentals of product description and support;
- Part 42, Integrated generic resources: Geometric and topological representation;
- Part 43, Integrated generic resources: Representation structures;
- Part 44, Integrated generic resources: Product structure configuration;
- Part 45, Integrated generic resources: Materials;
- Part 46, Integrated generic resources: Visual presentation;
- Part 47, Integrated generic resources: Shape variation tolerances;
- Part 49, Integrated generic resources: Process structure and properties;

- Part 101, Integrated application resources: Draughting;
- Part 104, Integrated application resources: Finite element analysis;
- Part 105, Integrated application resources: Kinematics;
- Part 201, Application protocol: Explicit draughting;
- Part 202, Application protocol: Associative draughting;
- Part 203, Application protocol: Configuration controlled design;
- Part 207, Application protocol: Sheet metal die planning and design;
- Part 210, Application protocol: Printed circuit assembly product design data;
- Part 213, Application protocol: Numerical control process plans for machined parts.

The structure of this International Standard is described in ISO 10303-1. The numbering of the parts of this International Standard reflects its structure:

- Part 11 specifies the description methods;
- Parts 21 and 22 specify the implementation methods;
- Parts 31 and 32 specify the conformance testing methodology and framework;
- Parts 41 to 49 specify the integrated generic resources;
- Parts 101 to 105 specify the integrated application resources;
- Parts 201 to 213 specify the application protocols.

Should further parts be published, they will follow the same numbering pattern.

Annexes A and B form an integral part of this part of ISO 10303. Annexes C, D, E and F are for information only.

Diskette

Users should note that this part of ISO 10303 comprises a diskette:
— the short names of entities given in annex A are also included on the diskette;
— the EXPRESS listings (annex C) are provided on the diskette only;
— a method to enable users to report errors in the documentation is given. Full details are provided in the file.

Introduction

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

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

- Presentation organization
- Presentation definition
- Presentation appearance
- Presentation resources

This part of ISO 10303 specifies the integrated resources for the visualization of displayable properties of products.

The information given in all four schemas of this part together is sufficient to describe in detail how product information shall be visualized by a receiving system. The presentation information contained in this part can be used only in conjunction with product information suitable for display. Presentation information as contained in this part cannot be displayed by itself without reference to product information.

The presentation organization schema describes the hierarchical and partially recursive structure of the presentation sets, areas and views in which images of the product information are displayed. It also explains how the components of the product information image and its annotation are organized as displayable objects and how these are placed into the context of presentations. This schema also accounts for the definition of the projective process for geometry by means of a camera model and for the specification of lighting and shading models.

The presentation definition schema serves to define how the individual geometric and non–geometric components of the product information are selected, assembled into presentation groups, and associated with presentation styles.

The presentation appearance schema defines the appearance attributes that can be chosen to describe the desired visual appearance of the displayable elements of the product information by enumerating the available graphical presentation styles.

The presentation resource schema provides basic graphical capabilities such as text font definition, symbol definition, and colour definition.

The visual presentation characteristics described in this part are often associated with information from other generic resource parts, especially with geometric and topological representations

(ISO 10303-42). The Application Protocols determine which resource parts are used together. Applications which make use of the generic resources in this part provide both the product information to be visually presented and the semantic meaning of the presentation. Possible applications include rendered views of product shape, results of scientific visualization, technical drawings, diagrams, charts, and graphics for technical publications.

Relation to graphics standards

The integrated resources specified in this part of ISO 10303 support the visual presentation of the properties of products. The generation of visual images using data specified by these integrated resources requires the use of an appropriate display system. This part of ISO 10303 specifies the input data to such systems, together with the necessary structures and constraints that relate presentation data to other aspects of product data.

Many display systems conform to existing ISO standards for computer graphics, such as GKS-3D (ISO/IEC 8805) and PHIGS/PHIGS PLUS (ISO/IEC 9592). This part of ISO 10303 takes into account the concepts and terminology of these standards. Input data specified by this part of ISO 10303 is therefore intended to be suitable for further processing by displays conforming to graphics standards.

This page intentionally left blank

Industrial automation systems and integration — Product data representation and exchange — Part 46 : Integrated generic resources: Visual presentation

1 scope

This part of ISO 10303 specifies the integrated resources for the visualization of displayable product information . Presentation data as provided in this part are combined with product data and are exchanged together between systems with the aim that the receiving system can construct one or several pictures of the product information suitable for human perception.

This part specifies the generic resources required to describe the desired visual appearance of product information in its picture. The actual generation of the picture from the product information and its presentation data is left to the receiving system. The actual depiction may deviate from this target because of limitations in the capabilities of graphics systems.

Product information can be visualized in two ways, either by realistic, life–like images according to the rules of projective geometry and light propagation and reflection, or by symbolic presentations that conform with draughting standards and conventions. This part supports both types of presentations. The two types of visualization processes require different kinds of graphical transformations and these may be combined in the same picture.

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

- Association between product data defined by other parts of ISO 10303 and presentation data;
- Support of graphics functionality in compliance with current ISO graphics standards;
- Definition of presentation style attributes for realistic and symbolic visualizations of geometric and non–geometric displayable elements in the product information;
- Control of approximation tolerances for geometric presentation elements;
- Methods for defining the appearance of characters and symbols in fonts;
- Support of externally defined character fonts and symbols;
- Image control by a layer mechanism;
- Nesting of presentation areas.

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

– Definition of product information;

– Exchange of purely graphical information without any relationship to product information;

– Definition of the contents of character font and symbol libraries.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO 10303. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO 10303 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

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

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

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

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

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

ISO/IEC 8824-1:—[1)], *Information technology — Open systems interconnection — Abstract syntax notation one (ASN.1) — Part 1: Specification of basic notation.*

[1)] To be published.

3 Definitions and abbreviations

3.1 Terms defined in ISO 10303–1

This part of ISO 10303 makes use of the following terms defined in ISO 10303–1.

- application
- data
- data exchange
- generic resource
- information
- integrated resource
- presentation
- product
- product data
- product information
- structure

3.2 Terms defined in this part of ISO 10303

For the purpose of this part of ISO 10303, the following definitions apply.

3.2.1 annotation: text and/or symbology used for the purpose of communicating product information.

3.2.2 displayable product information: facts, concepts, or instructions about a product that are displayed through the visualization process.

EXAMPLE 1 – Displayable information of a product are properties such as shape, dimensions and tolerances, and material.

3.2.3 layer: a collection of displayable items for the purpose of controlling visibility and presentation style.

3.2.4 picture: a two–dimensional graphical presentation of product properties for human perception.

3.2.5 presentation information: the information necessary to create a presentation of product information through visualization. Presentation information is meaningful only if it is associated with product information.

3.2.6 realistic presentation of properties: a type of visualization that makes use of physical laws and mathematical principles to produce lifelike images. Realistic presentation makes use of perspective transformations, reflectance calculations, shading transparency, and colour definitions.

3.2.7 state variable: a variable which represents a quantity, e.g. temperature.

3.2.8 symbol: a mark or characters that are interpreted as the conventional sign of some object, idea, function, or process.

3.2.9 symbolic presentation of properties: a type of visualization that makes use of engineering conventions and practices to produce annotation.

3.2.10 synthetic camera model: a model which describes the process of mapping product shape to two dimensions. The model uses an abstraction of the process used by a camera to create a photo.

3.2.11 visualization: a process by which displayable product information and presentation information are used to produce a picture.

3.3 Abbreviations

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

CIE	Commission Internationale de l'Eclairage. Used to refer to the CIE universal colour definition system.
HLS	Hue, Lightness, Saturation colour space.
HSV	Hue, Saturation, Value colour space.
RGB	Red, Green, Blue colour space.

4 Presentation organization

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

EXPRESS specification:

```
*)
SCHEMA presentation_organization_schema;

REFERENCE FROM presentation_resource_schema
    (colour,
     planar_box,
     presentation_scaled_placement);

REFERENCE FROM geometry_schema
    (axis2_placement_2d,
     axis2_placement_3d,
     cartesian_point,
     curve,
     direction,
     dot_product,
     geometric_representation_context,
     geometric_representation_item,
     plane
     );

REFERENCE FROM representation_schema
    (item_defined_transformation,
     item_in_context,
     mapped_item,
     representation,
     representation_item,
     representation_map,
     representation_relationship,
     representation_relationship_with_transformation);

REFERENCE FROM measure_schema
    (length_measure,
     positive_plane_angle_measure);

REFERENCE FROM support_resource_schema
    (identifier,
     label,
     text,
     bag_to_set);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

presentation_resource_schema	Clause 7 of this part of ISO 10303
geometry_schema	ISO 10303-42
representation_schema	ISO 10303-43
measure_schema	ISO 10303-41
support_resource_schema	ISO 10303-41

2 – The EXPRESS-G diagrams for this schema may be found in Annex E of this part of ISO 10303.

4.1 Introduction

The **presentation_organization_schema** provides a structure for the management of a picture and the components of a picture. It also specifies the relationship between the properties of a product and their representation in a picture. The components of a picture may be related to each other, either as an association between two components that are otherwise independent, or, as one component is an element of the definition of another. These relationships allow complex structures of pictures and components to be created.

The components of a picture may be organized within a hierarchy built of presentation sets, presentation areas, area dependent annotation representations, presentation views, view dependent annotation representations, and product data representation views. The hierarchy consists of the following four levels:

Level 1:

presentation set: A presentation set is a collection of independent pictures that are concerned with the same subject. Examples of presentation sets are a collection of drawing sheets or a collection of the images of several display screens. A presentation set is composed of one or many presentation areas.

Level 2:

presentation area: A presentation area is a generalization of a display that represents a single area on a display screen or device. A presentation area contains any number of presentation areas, presentation views, and area–dependent annotation representations.

Level 3:

presentation view: A presentation view is a two dimensional presentation of product shape, which includes any annotation that is associated with that view. A presentation

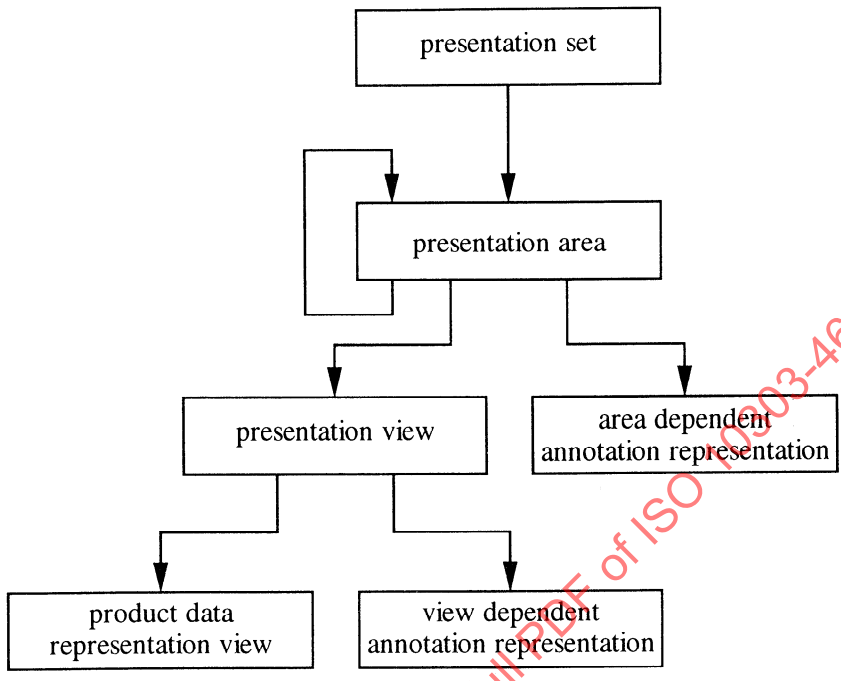


Figure 1 – Presentation hierarchy

view contains any number of product data representation views and view–dependent annotation representations.

area dependent annotation representation: An area dependent annotation representation is all annotation associated with a presentation area.

Level 4:

product data representation view: A product data representation view is a two–dimensional presentation of product shape, which includes any annotation associated with the product shape.

view–dependent annotation representation: A view–dependent annotation representation is the annotation associated with a presentation view.

An actual presentation hierarchy may contain more levels than described above. A single presentation area may itself be composed of several presentation areas. A presentation hierarchy may also contain less levels than described above, because some of the components making up the hierarchy are not required.

NOTE – Figures 1 and 2 illustrate the relationships between the levels of the presentation hierarchy.

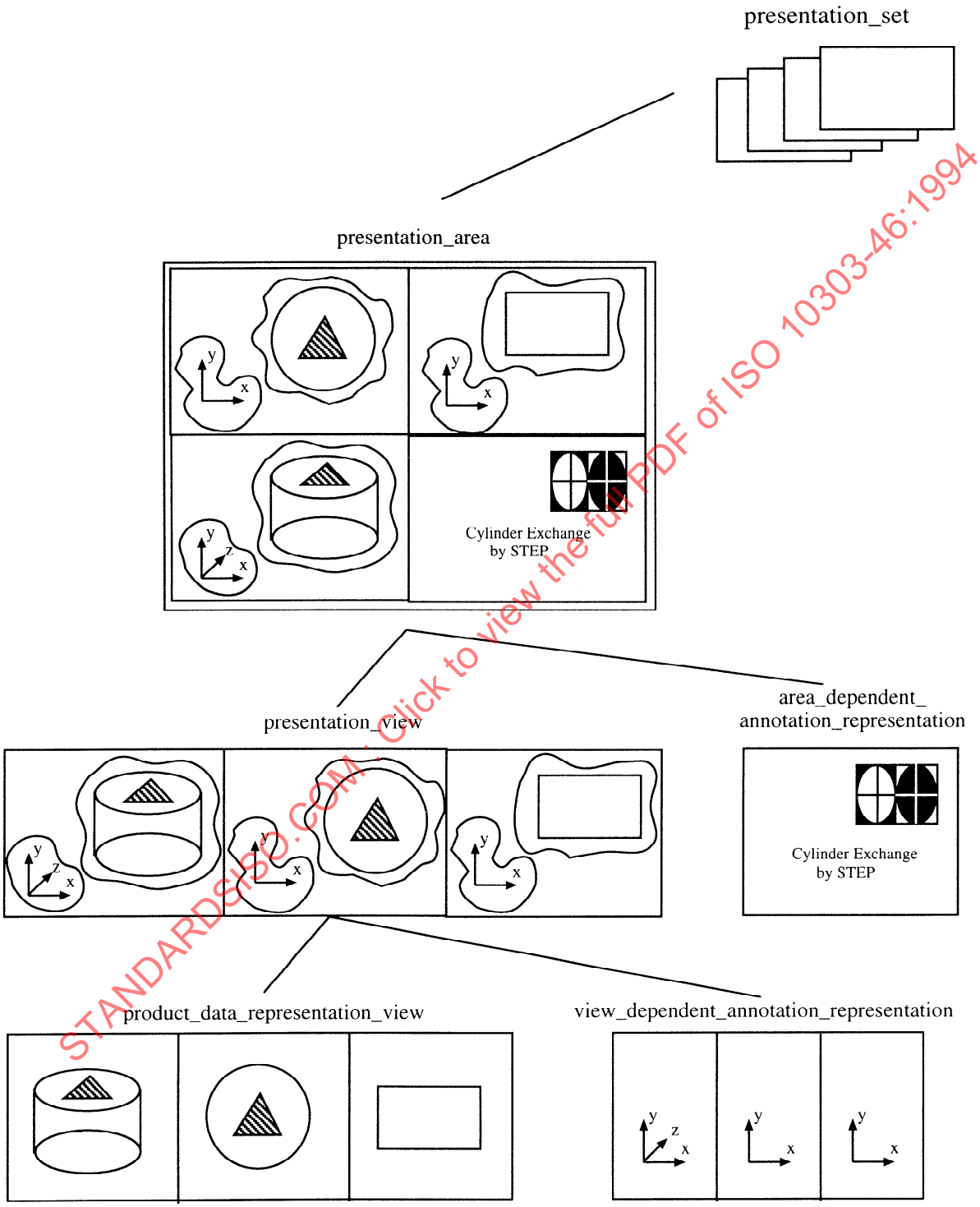


Figure 2 – Example of a presentation hierarchy

4.2 Fundamental concepts and assumptions

4.2.1 Presentation hierarchy

Within the presentation hierarchy, only a presentation area contains sufficient information to allow the unambiguous generation of a picture using a suitable output device such as a computer screen, printer, or plotter.

Other elements in the hierarchy do not provide the necessary placement information for proper positioning on a display device. Presentation views or other lower–level components are displayable only when related directly or indirectly to a presentation area.

The **presentation_organization_schema** describes the presentation hierarchy through representation of the different elements of the hierarchy and the relationships between them.

NOTE 1 – The concepts of representation and of relationships between representations are described in ISO 10303-43.

Each element of the presentation hierarchy is described using the **presentation_representation** entity, a subtype of **representation**. Subtypes of the **presentation_representation** entity describe different elements within the hierarchy.

NOTE 2 – The **representation** entity is defined in ISO 10303-43.

The context of each element of the presentation hierarchy is described using the **geometric_representation_context** entity; this is constrained to be two dimensional.

NOTE 3 – The **geometric_representation_context** entity is defined in ISO 10303-42.

The contents of each element of the presentation hierarchy are described by the set of **items** of each **presentation_representation**. The **items** are either two–dimensional geometry or annotation that are to be presented in the element, or the results of including other elements.

Some elements of the hierarchy are constrained with respect to their contents, or to the relationships that they may play with other elements. Specific semantics are associated with these constraints; for example, the **product_data_representation_view** entity describes an element of the hierarchy that only includes the results of projecting three–dimensional geometry or annotation.

The topmost level of the presentation hierarchy is represented by the entities **presentation_set**, **presentation_area**, and **area_in_set**. The **area_in_set** entity supports many–to–many relationships between presentation sets and presentation areas.

Other elements of the presentation hierarchy are related to each other using either the **presentation_representation_relationship** entity, or the **mapped_item** and **representation_map** entities.

NOTE 4 – The **mapped_item** and **representation_map** entities are defined in ISO 10303-43.

An association between two elements of the hierarchy which are independently defined is described using the **presentation_representation_relationship** entity. This describes the relationship between two instances of the **presentation_representation** entity; in this relationship, one **presentation_representation** is denoted as the parent, the other as the child. The description of a transformation is included in the relationship; this transformation is the geometric

relationship between the **items** of the parent **presentation_representation** and those of the child **presentation_representation**.

EXAMPLE 2 – To define a hierarchy which consists of a single **presentation_area**, two **presentation_views** and a single **area_dependent_annotation_representation**, three instances of **presentation_representation_relationship** are required as shown in figure 3.

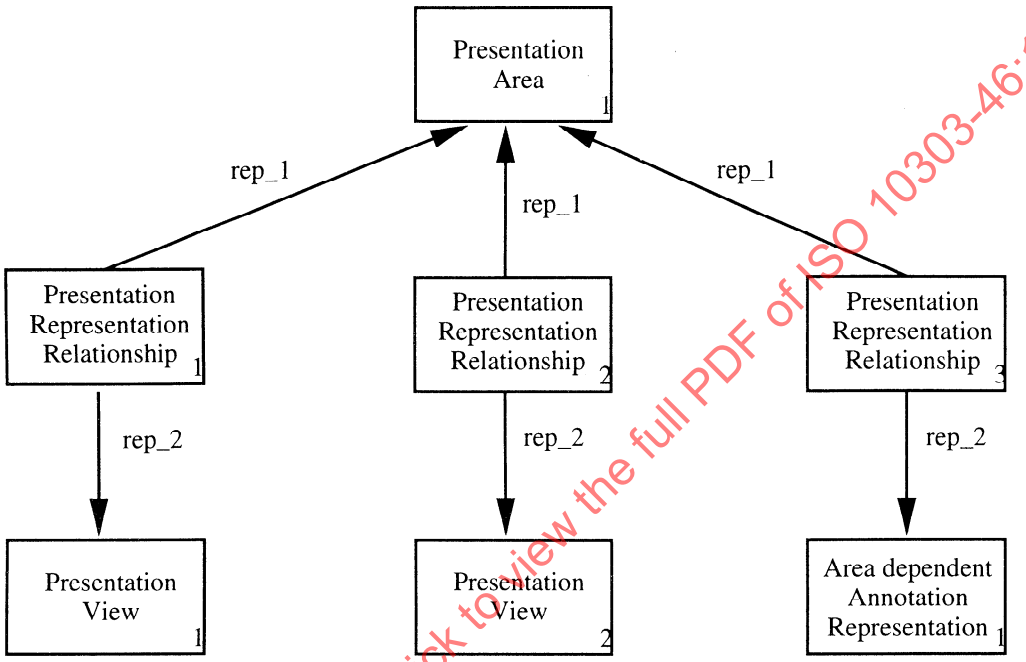


Figure 3 – Mapping the presentation hierarchy to instances of entities

An association between two elements of the hierarchy, where one participates in the definition of the other, is described using the **mapped_item** and **representation_map** entities.

Such an association is described by an instance of the **mapped_item** entity. This instance is included as one of the **items** of the **presentation_representation** that contains the other. The second **presentation_representation** is referenced as the **mapped_representation** of a **representation_map**, that is specified as the **mapping_source** of the **mapped_item**. The transformation that describes the geometric relationship between the **items** of the two **presentation_representations** is described by the **mapping_target** of the **mapped_item** and the **mapping_origin** of the **representation_map**.

EXAMPLE 3 To include a **presentation_view** within a **presentation_area** requires one instance of **mapped_item**, one instance of **representation_map**, and two instances of **axis2_placement_2d** that act as the origin and the target for the mapping, as shown in figure 4.

NOTE 5 – The **axis2_placement_2d** entity is defined in ISO 10303-42.

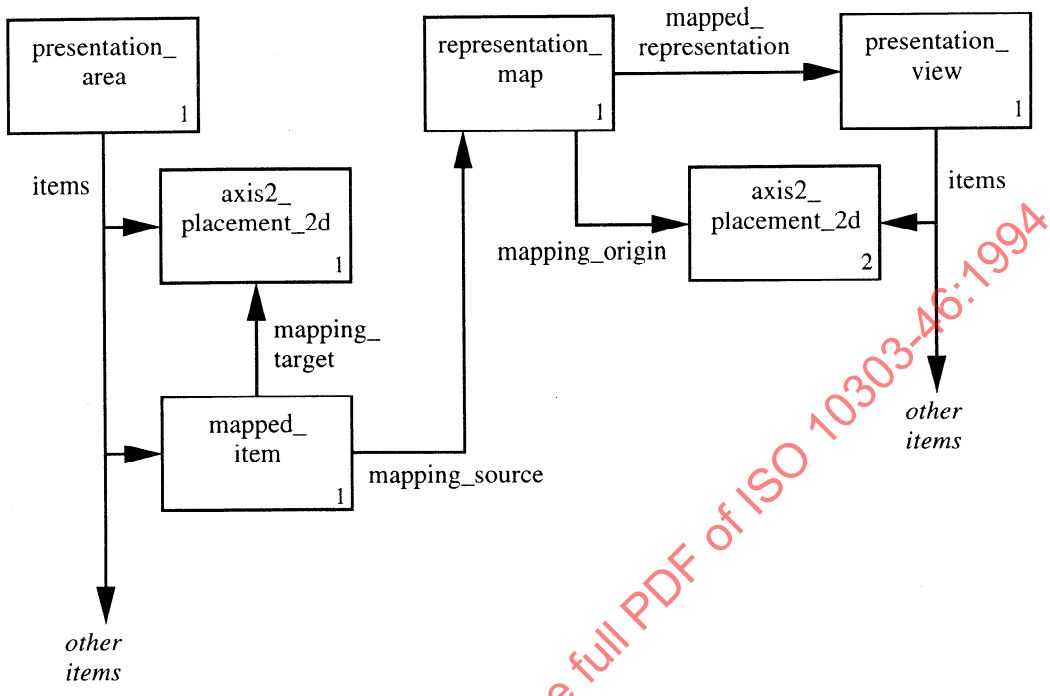


Figure 4 – Association of presentation_view and presentation_area using mapped_item

4.2.2 Camera model and projection

Each level of the presentation hierarchy consists only of two–dimensional geometry or annotation. A three–dimensional synthetic camera model must be specified to associate a presentation with a three–dimensional product shape or planar annotation in three–dimensional space. This model specifies how the projection of three–dimensional geometry and annotation to two dimensions shall be executed by the displaying system. The three–dimensional synthetic camera model is defined in analogy to the graphics standards GKS–3D and PHIGS. More details about the synthetic camera model can be found in [6], [7] and in computer graphics literature, e.g., in [11] or [12]. For draughting requirements, a two–dimensional camera model is also supported. This two–dimensional model performs scaling and translation in two–dimensional space.

The **product_data_representation_view** is the only component of the presentation hierarchy which may consist of two–dimensional projections of associated three–dimensional product shape or of annotation in three–dimensional space. Because this part of ISO 10303 does not define the projected picture, but all information necessary to compute the projection, only a placeholder for that picture is part of the **product_data_representation_view**. This placeholder is called **camera_image** and refers to the camera model and the two– or three–dimensional product shape elements or annotation.

To allow more realistic presentations of three–dimensional objects, hidden line and hidden surface removal, as well as light sources, can be specified for a three–dimensional camera model.

4.2.3 Layers

A layer is a collection of product shape elements, annotation elements, or components of the presentation hierarchy grouped for the purpose of controlling visibility and style. The set of all items associated with a layer can be defined as visible or invisible within a component of the presentation hierarchy. A single item can be associated with several layers and a single layer can be used in several components of the presentation hierarchy. For an item which is associated with several layers, a different presentation style can be assigned to each layer. A layer is defined by a **presentation_layer_assignment**. The visibility and style assignment of a given layer within a component of the presentation hierarchy is specified by a **presentation_layer_usage**.

4.2.4 Association of presentation with a product model

This part of ISO 10303 allows the association of presentation information with the product information which is presented. This association is achieved through the use of a **presented_item_representation**. This entity relates the item which is presented with the presentation of that item. The **presented_item** shall be specialized by Application Protocols.

EXAMPLE 4 – Presented items are the body design of a car, the layout of an electronic chip, or the architecture of a building.

4.3 Presentation organization schema type definitions

4.3.1 presentation_size_assignment_select

The **presentation_size_assignment_select** type specifies the objects to which a size may be assigned.

EXPRESS specification:

```
*)
TYPE presentation_size_assignment_select = SELECT
  (presentation_view,
   presentation_area,
   area_in_set);
END_TYPE;
(*
```

4.3.2 area_or_view

The **area_or_view** type indicates the things that may be assigned a background colour.

EXPRESS specification:

```
*)
TYPE area_or_view = SELECT
  (presentation_area,
   presentation_view);
END_TYPE;
(*
```

4.3.3 central_or_parallel

A **central_or_parallel** indicates the type of perspective transformation that is used by the **camera_model**. The types are either a parallel projection onto the viewing plane or a central projection from the **projection_point** to the viewing plane. See figures 7 and 8.

EXPRESS specification:

```
*)
TYPE central_or_parallel = ENUMERATION OF
  (central,
   parallel);
END_TYPE;
(*
```

Enumerated item definitions:

central: the projection is made along lines emanating from a **projection_point** and intersecting with the **view_window**.

parallel: the projection is made parallel to the line from a **projection_point** to the geometric centre of the **view_window**.

4.3.4 layered_item

The **layered_item** specifies items to be grouped into layers by a **presentation_layer_assigment**.

EXPRESS specification:

```
*)
TYPE layered_item = SELECT
  (presentation_representation,
   representation_item);
END_TYPE;
(*
```

Informal propositions:

IP1: A **representation_item** that is assigned to a layer shall be a **styled_item**, or shall be defined by one or more **styled_item**.

4.3.5 presentation_representation_select

A **presentation_representation_select** is used to allow **presented_item_representation** to associate a presentation with the item being presented.

EXPRESS specification:

```
*)
TYPE presentation_representation_select = SELECT
  (presentation_representation,
   presentation_set);
END_TYPE;
(*
```

4.4 Presentation organization schema entity definitions: presentation hierarchy

4.4.1 presentation_set

The **presentation_set** is a collection of **presentation_areas**.

EXPRESS specification:

```
*)
ENTITY presentation_set;
INVERSE
  areas : SET [1:?] OF area_in_set FOR in_set;
END_ENTITY;
(*
```

Attribute definitions:

areas: the set of **presentation_area** entities which make up the **presentation_set**.

4.4.2 presentation_representation

A **presentation_representation** represents the definition of the picture to be generated by a system which displays the presentation information. The picture refers to an object which may consist of two–dimensional geometry, three–dimensional geometry, and annotation. The reference to geometry and annotation is made indirectly by building a hierarchy of **presentation_representations**. Lower levels of this hierarchy are constrained to contain only geometry or annotation.

NOTE 1 – For the description of the presentation hierarchy see 4.1.

EXPRESS specification:

```
*)
ENTITY presentation_representation
  SUBTYPE OF (representation);
WHERE
  WR1: SELF\representation.
            context_of_items\geometric_representation_context.
            coordinate_space_dimension = 2;
  WR2: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT'
       IN TYPEOF (SELF\representation.context_of_items);
END_ENTITY;
(*
```

Formal propositions:

WR1: The picture shall have a dimensionality of 2.

WR2: A **presentation_representation** shall have a geometric context.

NOTES

2 – When presenting a three–dimensional object, a two–dimensional projection of that three–dimensional object is used. A **presentation_representation** is a picture that represents things after they have been projected.

3 – The actual projected geometry does not exist in this part of ISO 10303. Instead, the **presentation_representation** stands for the projected geometry, and the information needed to create the projection is also included in the model.

4.4.3 presentation_area

The **presentation_area** represents a picture that can contain other pictures and can itself be contained in another picture which is also a **presentation_area**.

The size of the **presentation_area** is defined by the **presentation_size** entity. Every **presentation_area** is either given a size directly by its use in a **presentation_size** entity, or indirectly by its use in **area_in_set** entities. Clipping of a picture based on the **presentation_area** size is executed by the system that creates an actual display based on the presentation information. If a **presentation_area** is contained within another **presentation_area**, then the boundaries of the containing **presentation_area** are also used to clip the contained **presentation_area**.

A background colour may be specified through the use of a **presentation_area** in the **background_colour** entity. Only one **background_colour** entity shall use any **presentation_area**.

EXPRESS specification:

```
*)
ENTITY presentation_area
  SUBTYPE OF (presentation_representation);
WHERE
  WR1: ((SIZEOF (QUERY (ais <* USEDIN (SELF, 'PRESENTATION_ORGANIZATION_SCHEMA.' +
                                          'AREA_IN_SET.AREA') |
          SIZEOF (USEDIN (ais, 'PRESENTATION_ORGANIZATION_SCHEMA.' +
                               'PRESENTATION_SIZE.UNIT')) =1)) > 0) OR
        (SIZEOF (USEDIN (SELF, 'PRESENTATION_ORGANIZATION_SCHEMA.' +
                               'PRESENTATION_SIZE.UNIT')) =1));
END_ENTITY;
(*
```

Formal propositions:

WR1: A **presentation_area** shall be contained in a **presentation_set** through participation in **area_in_set.area**, where the **area_in_set** participates in **presentation_size.unit**, or, non–exclusively, a **presentation_area** shall get its size directly from a **presentation_size**.

Informal propositions:

IP1: Any **presentation_representation** shall be presented in the context of a tree which has as its root a **presentation_area**.

IP2: The size of a **presentation_area** shall not be specified more than once.

NOTE – Constraints on the usage of this entity are found in the rules of **presentation_representation_relationship**.

4.4.4 area_in_set

An **area_in_set** specifies the participation of a **presentation_area** in a **presentation_set**. A **presentation_area** may participate in many **presentation_sets**; a **presentation_set** includes at least one **presentation_area**.

EXPRESS specification:

```
*)
ENTITY area_in_set;
  area   : presentation_area;
  in_set : presentation_set;
END_ENTITY;
(*
```

Attribute definitions:

area: the **presentation_area** that participates in the specified **presentation_set**.

in_set: the **presentation_set** in which the specified **presentation_area** participates.

4.4.5 presentation_view

A **presentation_view** is a picture that may contain other pictures and may itself be contained in a picture. It does not represent a complete picture and shall not be displayed without being placed in a **presentation_area**.

The size of a **presentation_view** may be defined by its reference from a **presentation_size** entity. Clipping based on the **presentation_view** size is executed by the system that creates an actual display based on the presentation information. If a **presentation_view** size is not defined, then clipping is executed solely based on the size of the **presentation_area** in which the **presentation_view** is contained.

A background colour may be specified through the use of the **background_colour** entity.

EXPRESS specification:

```
*)
ENTITY presentation_view
  SUBTYPE OF (presentation_representation);
END_ENTITY;
(*
```

NOTE – Constraints on the usage of this entity are found in the rules of **presentation_representation_relationship**.

4.4.6 area_dependent_annotation_representation

An **area_dependent_annotation_representation** is a picture that may be contained in the picture of a **presentation_area**. The picture it represents is made up of those of its items which

are **annotation_occurrence** entities. An **area_dependent_annotation_representation** may be related only to a **presentation_area**.

EXPRESS specification:

```
*)
ENTITY area_dependent_annotation_representation
  SUBTYPE OF (presentation_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF\representation.items |
       NOT (SIZEOF (['PRESENTATION_DEFINITION_SCHEMA.' +
                     'ANNOTATION_OCCURRENCE',
                     'GEOMETRY_SCHEMA.AXIS2_PLACEMENT'] *
            TYPEOF(item)) = 1
       ))) = 0;
  WR2: SIZEOF (QUERY (item <* SELF\representation.items |
       ('PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_OCCURRENCE' IN
         TYPEOF (item))
       )) >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: The only kinds of **representation_items** that shall be in the set of **items** in an **area_dependent_annotation_representation** are **annotation_occurrence** or **axis2_placement** entities.

WR2: At least one of the items in an **area_dependent_annotation_representation** shall be an **annotation_occurrence**.

Informal propositions:

IP1: When an **area_dependent_annotation_representation** is presented, it shall be contained in a **presentation_area**.

NOTE – Constraints on the usage of this entity are found in the rules of **presentation_representation_relationship**.

4.4.7 product_data_representation_view

A **product_data_representation_view** is a picture that consists of two–dimensional projections of geometry, annotation, or both. It may contain other pictures, and may be contained in a picture. It does not represent a complete picture and shall not be displayed without being placed in a **presentation_view**.

EXPRESS specification:

```
*)
ENTITY product_data_representation_view
  SUBTYPE OF (presentation_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF\representation.items |
       NOT (SIZEOF (['PRESENTATION_ORGANIZATION_SCHEMA.CAMERA_IMAGE',
                     'GEOMETRY_SCHEMA.AXIS2_PLACEMENT'] *
```

```
                    TYPEOF (item)) = 1
      ))) = 0;
  WR2: SIZEOF (QUERY (item <* SELF\representation.items |
       ('PRESENTATION_ORGANIZATION_SCHEMA.CAMERA_IMAGE' IN
        TYPEOF (item))
       )) >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: The items in a **product_data_representation_view** shall be either **camera_images** or **axis2_placements**.

WR2: At least one of the items in a **product_data_representation_view** shall be a **camera_image**.

Informal propositions:

IP1: When a **product_data_representation_view** is presented, it shall be contained in a **presentation_view**.

NOTE – Constraints on the usage of this entity are found in the rules of **presentation_representation_relationship**.

4.4.8 view_dependent_annotation_representation

A **view_dependent_annotation_representation** is a picture that may be contained in the picture of a **presentation_view**. The picture that it represents contains only items which are **annotation_occurrence** entities. A **view_dependent_annotation_representation** may be related only to a **presentation_view**.

EXPRESS specification:

```
*)
ENTITY view_dependent_annotation_representation
  SUBTYPE OF (presentation_representation);
WHERE
  WR1: SIZEOF (QUERY (item <* SELF\representation.items |
       NOT (SIZEOF (['PRESENTATION_DEFINITION_SCHEMA.' +
                     'ANNOTATION_OCCURRENCE',
                     'GEOMETRY_SCHEMA.AXIS2_PLACEMENT'] *
            TYPEOF(item)) = 1
       ))) = 0;
  WR2: SIZEOF (QUERY (item <* SELF\representation.items |
       ('PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_OCCURRENCE' IN
        TYPEOF (item))
       )) >= 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: The only kinds of **representation_items** that shall be in the set of items in a **view_dependent_annotation_representation** are **annotation_occurrence** or **axis2_placement** entities.

WR2: At least one of the items in a **view_dependent_annotation_representation** shall be an **annotation_occurrence**.

Informal propositions:

IP1: When a **view_dependent_annotation_representation** is presented, it shall be contained in a **presentation_view**.

NOTE – Constraints on the usage of this entity are found in the rules of **presentation_representation_relationship**.

4.4.9 presentation_size

A presentation_size is used to define the size of a **presentation_area** or **presentation_view**. The size for a **presentation_area** may be assigned directly or may be assigned dependent on a **presentation_set** which contains a **presentation_area**. This allows the assignment of different sizes to a single **presentation_area** for each **presentation_set** which contains the area.

EXPRESS specification:

```
*)
ENTITY presentation_size;
  unit : presentation_size_assignment_select;
  size : planar_box;
WHERE
  WR1: (('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_REPRESENTATION'
         IN TYPEOF (SELF.unit)) AND
         item_in_context (SELF.size,
                          SELF.unit\representation.context_of_items)
       )
            OR
       (
       ('PRESENTATION_ORGANIZATION_SCHEMA.AREA_IN_SET'
         IN TYPEOF (SELF.unit)) AND
       (SIZEOF (QUERY ( ais <* SELF.unit\area_in_set.in_set.areas |
              NOT item_in_context (SELF.size, ais.area\representation.
                                                 context_of_items) )) = 0)
       );
END_ENTITY;
(*
```

Attribute definitions:

unit: a **presentation_view**, **presentation_area**, or an **area_in_set** as it relates a **presentation_area** with a **presentation_set**.

size: a **planar_box** which describes the size of a unit.

Formal propositions:

WR1: If the **unit** is a **presentation_representation**, the **planar_box** shall be an item in a representation which has the same context as the **unit**. If the **unit** is an **area_in_set** the **planar_box** shall be an item in the context of each area belonging to the set.

4.4.10 background_colour

A **background_colour** is the colour used for the background of a presentation area or view.

EXPRESS specification:

```
*)
ENTITY background_colour
  SUBTYPE OF (colour);
  presentation : area_or_view;
UNIQUE
  UR1:  presentation;
END_ENTITY;
(*
```

Attribute definitions:

presentation: a **presentation_area** or **presentation_view** for which this background colour is specified.

Formal propositions:

UR1: Only one background colour may be specified for any **presentation_area** or **presentation_view**.

4.4.11 presentation_representation_relationship

A **presentation_representation_relationship** is a relationship between **presentation_representation** entities. The relationship is directed, meaning that the child representation (**rep_2**) is transformed into the parent representation (**rep_1**), and no meaning is intended for the inverse transformation. The **presentation_representation_relationship** restricts the relationships between certain **presentation_representations** to ensure that **presentation_representation** entities are arranged in a hierarchy as described in clause 4.1.

EXPRESS specification:

```
*)
ENTITY presentation_representation_relationship
  SUBTYPE OF (representation_relationship_with_transformation);
WHERE
  WR1: 'PRESENTATION_DEFINITION_SCHEMA.PRESENTATION_REPRESENTATION' IN
         TYPEOF (SELF\representation_relationship.rep_1);
  WR2: 'PRESENTATION_DEFINITION_SCHEMA.PRESENTATION_REPRESENTATION' IN
         TYPEOF (SELF\representation_relationship.rep_2);
  WR3: acyclic_presentation_representation_relationship (SELF,
         [SELF\representation_relationship.rep_2]);
  WR4: NOT (('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_AREA' IN
         TYPEOF (SELF\representation_relationship.rep_1))
```

```
                AND
        NOT (SIZEOF (['PRESENTATION_ORGANIZATION_SCHEMA.' +
                      'PRODUCT_DATA_REPRESENTATION_VIEW',
                      'PRESENTATION_ORGANIZATION_SCHEMA.' +
                      'VIEW_DEPENDENT_ANNOTATION_REPRESENTATION'] *
        TYPEOF (SELF\representation_relationship.rep_2)) = 0));
  WR5: NOT (('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_VIEW'
        IN TYPEOF (SELF\representation_relationship.rep_1))
                AND
      NOT (SIZEOF (['PRESENTATION_ORGANIZATION_SCHEMA.' +
                    'PRESENTATION_AREA',
                    'PRESENTATION_ORGANIZATION_SCHEMA.' +
                    'PRESENTATION_VIEW',
                    'PRESENTATION_ORGANIZATION_SCHEMA.' +
                    'AREA_DEPENDENT_ANNOTATION_REPRESENTATION'] *
             TYPEOF (SELF\representation_relationship.rep_2))=0));
  WR6: (NOT ('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_VIEW' IN
        TYPEOF(SELF\representation_relationship.rep_2)))
             XOR
        ('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_AREA'IN
        TYPEOF(SELF\representation_relationship.rep_1));
  WR7: (NOT ('PRESENTATION_ORGANIZATION_SCHEMA.' +
            'PRODUCT_DATA_REPRESENTATION_VIEW' IN
        (TYPEOF(SELF\representation_relationship.rep_1) +
         TYPEOF(SELF\representation_relationship.rep_2))))
             XOR
      ('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_VIEW' IN
        TYPEOF(SELF\representation_relationship.rep_1))
               AND
      ('PRESENTATION_ORGANIZATION_SCHEMA.PRODUCT_DATA_REPRESENTATION_VIEW' IN
        TYPEOF(SELF\representation_relationship.rep_2));
  WR8: 'PRESENTATION_ORGANIZATION_SCHEMA.GRAPHICAL_TRANSFORMATION' IN
        TYPEOF(SELF\representation_relationship_with_transformation.
                    transformation_operator);
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_relationship.rep_1: the **presentation_representation** that plays the role of a parent in a tree of **presentation_representations**.

SELF\representation_relationship.rep_2: the **presentation_representation** that plays the role of a child in a tree of **presentation_representations**.

Formal propositions:

WR1: **rep_1** shall be a **presentation_representation**.

WR2: **rep_2** shall be a **presentation_representation**.

WR3: A **presentation_representation_relationship** shall not participate in a tree of **presentation_representations** where the root of the tree is also a leaf of its own tree.

WR4: Pictures represented by **product_data_representation_view** or **view_dependent_annotation_representation** shall not be related to a picture represented by a **presentation_area**.

WR5: Pictures represented by **presentation_area**, **presentation_view**, or **area_dependent_annotation_representation** shall not be related to a picture represented by a **presentation_view**.

WR6: The picture represented by a **presentation_view** may only be related to the picture represented by a **presentation_area**.

WR7: A **product_data_representation_view** shall only participate in a **presentation_representation_relationship**, if the **rep_1** is a **presentation_view**.

WR8: Pictures that are related to other pictures relate to those pictures by **graphical_transformations**.

4.4.12 graphical_transformation

A **graphical_transformation** is a transformation which allows arbitrary two–dimensional positioning, rotation, and uniform scaling. It is used by **presentation_representation_relationship** to define the transformation between related **presentation_representations**. The transformation is defined as follows.

The **transform_item_1.location** is transformed into the **transform_item_2.placement.location**. Additionally the x axis defined by **transform_item_1.p[1]** is mapped to the x axis **transform_item_2.placement.p[1]**. A similar mapping is performed for the y axes **transform_item_1.p[2]** and **transform_item_2.placement.p[2]**. Finally the scaling specified by **transform_item_2.scaling** is applied.

NOTE 1 – Figure 5 shows the mechanism of **graphical_transformation**.

EXPRESS specification:

```
*)
ENTITY graphical_transformation
  SUBTYPE OF (item_defined_transformation);
WHERE
  WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_2D' IN
       TYPEOF (SELF\item_defined_transformation.transform_item_1);
  WR2: 'PRESENTATION_RESOURCE_SCHEMA.PRESENTATION_SCALED_PLACEMENT' IN
        TYPEOF (SELF\item_defined_transformation.transform_item_2);
END_ENTITY;
(*
```

Attribute definitions:

SELF\item_defined_transformation.transform_item_1: an **axis2_placement_2d** specifying the starting point of the transformation.

SELF\item_defined_transformation.transform_item_2: a **presentation_scaled_placement** specifying the ending point of the transformation.

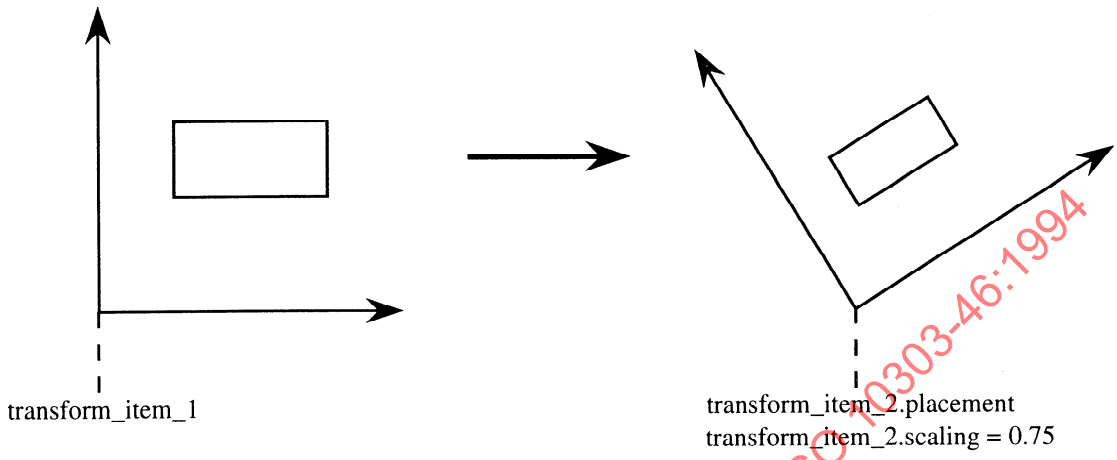


Figure 5 – Graphical transformation

Formal propositions:

WR1: The **tranform_item_1** attribute of a **graphical_transformation** shall be an **axis2_placement_2d**.

WR2: The **tranform_item_2** attribute of a **graphical_transformation** shall be a **presentation_scaled_placement**.

NOTE 2 – The attributes **tranform_item_1** and **tranform_item_2** are defined in ISO 10303-43.

4.5 Presentation organization schema entity definitions: camera model and projection

4.5.1 camera_model

A **camera model** contains the information needed to create a projection or mapping from a representation to a picture of that representation.

EXPRESS specification:

```
*)
ENTITY camera_model
  SUPERTYPE OF (ONEOF(camera_model_d2, camera_model_d3))
  SUBTYPE OF (geometric_representation_item);
WHERE
  WR1: (SIZEOF (USEDIN (SELF, 'REPRESENTATION_SCHEMA.' +
                              'ITEM_DEFINED_TRANSFORMATION.' +
                              'TRANSFORM_ITEM_1')) +
        SIZEOF (USEDIN (SELF, 'REPRESENTATION_SCHEMA.' +
                              'REPRESENTATION_MAP.MAPPING_ORIGIN'))
```

```
      ) > 0;
  WR2: SIZEOF(USEDIN(SELF,'PRESENTATION_APPEARANCE_SCHEMA.'+
                          'STYLED_ITEM.ITEM')) = 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: A **camera_model** shall specify the projection for at least one **representation**.

WR2: A **camera_model** shall not be referenced by a **styled_item** entity.

4.5.2 camera_model_d2

A **camera_model_d2** contains the information needed to create a two–dimensional mapping from a representation to a picture of that representation.

EXPRESS specification:

```
*)
ENTITY camera_model_d2
  SUBTYPE OF (camera_model);
  view_window          : planar_box;
  view_window_clipping : BOOLEAN;
WHERE
  WR1: SELF\geometric_representation_item.dim = 2;
END_ENTITY;
(*
```

Attribute definitions:

view_window: the rectangular boundaries in the coordinate space of the **representation** that is being projected into the **product_data_representation_view**. A translation and possibly non–uniform scaling are applied to the **view_window** so, that the edges of the **view_window** coincide with the edges of the **mapping_target** of the **camera_image**.

view_window_clipping: the determination of whether or not clipping is performed against the **view_window**. A value of TRUE indicates that clipping against the **view_window** is performed. A value of FALSE indicates that no clipping against the **view_window** is performed.

NOTES

1 – If **view_window_clipping** has a value of FALSE, the two–dimensional projection may extend beyond the boundaries of the **mapping_target** specified by the **camera_image**.

2 – Clipping is performed prior to projection.

Formal propositions:

WR1: A **camera_model_d2** shall be two–dimensional.

NOTE 3 – The mechanism of the **camera_model_d2** is shown in figure 6.

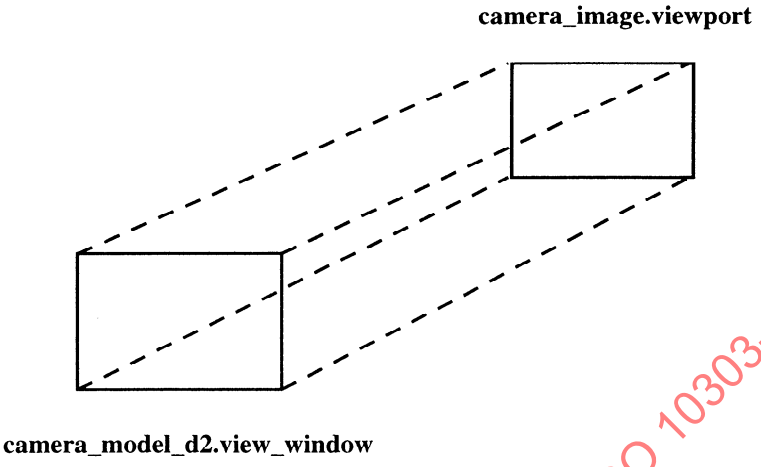


Figure 6 – Camera model d2

4.5.3 camera_model_d2_shape_clipping

A **camera_model_d2_shape_clipping** is a **camera_model_d2** with additional clipping specified for the mapped representation.

EXPRESS specification:

```
*)
ENTITY camera_model_d2_shape_clipping
  SUBTYPE OF (camera_model_d2);
  shape_clipping : curve;
END_ENTITY;
(*
```

Attribute definitions:

shape_clipping: a closed curve indicating a clipping region for the representation which is projected.

NOTE – If **view_window_clipping** has a value of TRUE, the representation which is projected is clipped against both the **view_window** and the region specified by **shape_clipping**.

Informal propositions:

IP1: The curve specified for **shape_clipping** shall be closed and not self–intersecting.

4.5.4 camera_model_d3

A **camera_model_d3** contains the information needed to create the projection from a three–dimensional representation to a two–dimensional picture of that representation. See 4.5.5 for details of the projection.

EXPRESS specification:

```
*)
ENTITY camera_model_d3
  SUBTYPE OF (camera_model);
  view_reference_system : axis2_placement_3d;
  perspective_of_volume : view_volume;
WHERE
  WR1: (dot_product (SELF.view_reference_system.p[3],
         SELF.perspective_of_volume.view_window.placement.p[3]) = 1.0)
         AND
       (SELF.view_reference_system.location.coordinates[3] =
        SELF.perspective_of_volume.view_window.
             placement.location.coordinates[3]);
  WR2: SELF\geometric_representation_item.dim = 3;
END_ENTITY;
(*
```

Attribute definitions:

view_reference_system: an intermediate three–dimensional coordinate space in the coordinate space of the representation being projected.

perspective_of_volume: the information needed to determine how to project the geometry of the representation. The **perspective_of_volume** is defined in the intermediate three–dimensional coordinate space of the **view_reference_system**.

Formal propositions:

WR1: the rectangle indicated by the **view_window** of the **perspective_of_volume** is in the same plane as the x and y axis of the **axis2_placement** indicated by the **view_reference_system**.

WR2: A **camera_model_d3** is three–dimensional.

4.5.5 view_volume

A **view_volume** is defined in the **view_reference_system** of the camera model which uses the volume. It defines a volume that is projected onto the viewport of the **product_data_representation_view**. This volume is either a truncated pyramid or a parallelepiped. The contents of this volume are projected onto the rectangle defined by the **view_window** which is then mapped onto the viewport.

If the type of projection is **parallel**, the projection is made parallel to the line from the **projection_point** to the geometric centre of the **view_window**, and the **view_volume** is a parallelepiped. If the type of projection is **central**, the projection is made along lines emanating from a **projection_point** and intersecting with the **view_window**, and the **view_volume** is a truncated pyramid. See figure 8 for the definition of the parallelepiped and figure 7 for the definition of the truncated pyramid.

EXPRESS specification:

```
*)
ENTITY view_volume;
  projection_type            : central_or_parallel;
  projection_point           : cartesian_point;
  view_plane_distance        : length_measure;
  front_plane_distance       : length_measure;
  front_plane_clipping       : BOOLEAN;
  back_plane_distance        : length_measure;
  back_plane_clipping        : BOOLEAN;
  view_volume_sides_clipping : BOOLEAN;
  view_window                : planar_box;
END_ENTITY;
(*
```

Attribute definitions:

projection_type: the indication of whether the projection is central or parallel.

projection_point: the centre of the projection, i.e., the location from which the items are viewed. For a parallel projection, this point indicates a line from itself to the geometric centre of the **view_window**. For a central projection, this point is the apex of the truncated pyramid.

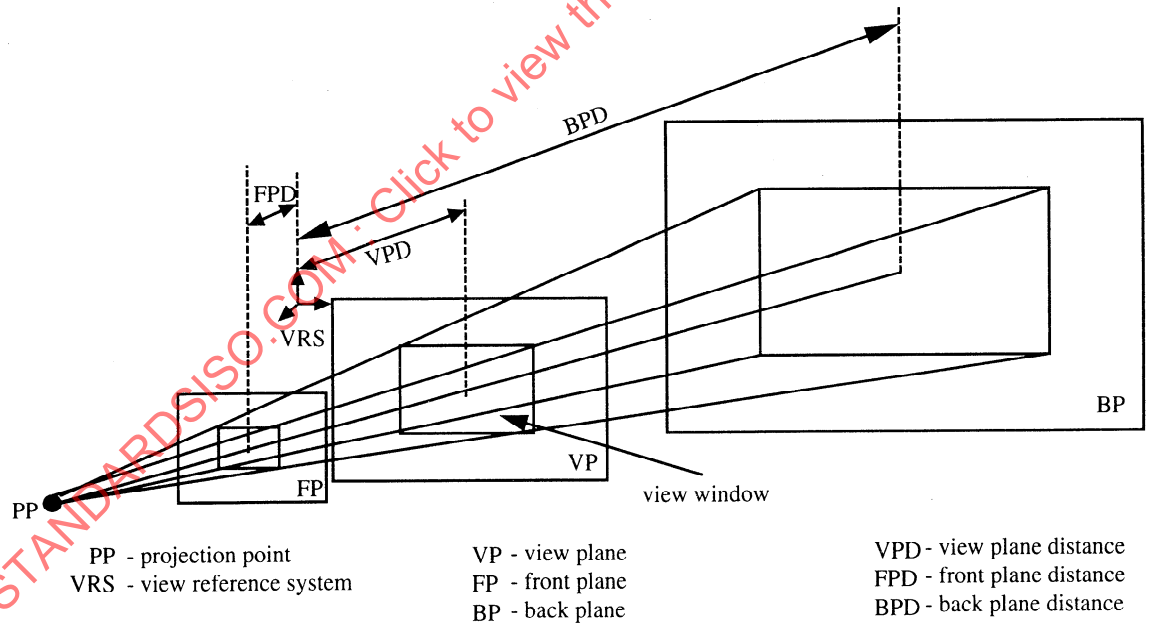
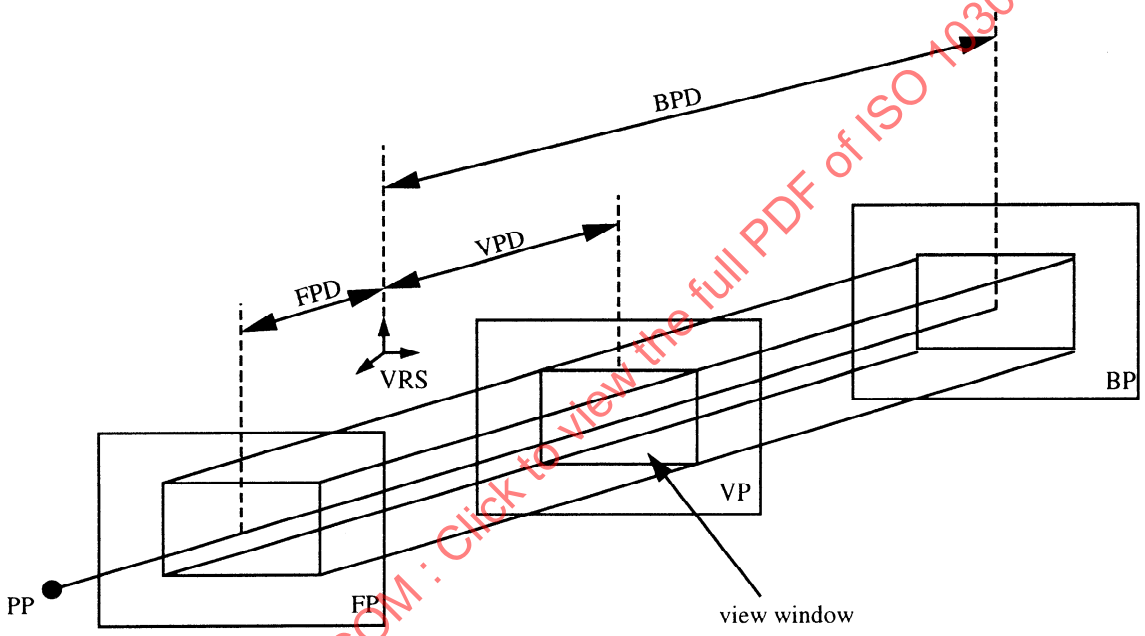


Figure 7 – View volume, projection type CENTRAL



PP - projection point
VRS - view reference system

VP - view plane
FP - front plane
BP - back plane

VPD - view plane distance
FPD - front plane distance
BPD - back plane distance

Figure 8 – View volume, projection type PARALLEL

view_plane_distance: a distance along the z axis of the **axis2_placement** indicated by the **view_reference_system**. This distance positions the origin of the **view_window**.

front_plane_distance: a signed distance along the z axis of the **axis2_placement** indicated by the **view_reference_system**. This distance specifies a plane parallel to the **view_window** plane. It is the distance to either the top of the truncated pyramid or the front of the parallelepiped, depending on the type of projection.

front_plane_clipping: the indication of whether or not to clip the geometry of the projected representation against the plane represented by the **front_plane_distance**. A value of TRUE indicates that clipping is performed; a value of FALSE indicates that clipping is not performed.

NOTE 1 – If **front_plane_clipping** has a value of FALSE, objects behind the projection point may be visible. Behind in this sense means in the opposite direction from the projection point than the view plane.

back_plane_distance: a distance along the z axis of the **axis2_placement** indicated by the **view_reference_system** attribute. This distance specifies a plane parallel to the **view_window** plane. It is either the bottom of the truncated pyramid or the back of the parallelepiped, depending on the type of projection.

back_plane_clipping: the indication of whether or not to clip the geometry of the projected representation against the plane represented by the **back_plane_distance**. A value of TRUE indicates that clipping is performed; a value of FALSE indicates that clipping is not performed.

view_volume_sides_clipping: the indication of whether or not to clip the geometry of the projected representation against the planes which are the sides of the volume defined by the **view_volume**. A value of TRUE indicates that clipping is performed; a value of FALSE indicates that clipping is not performed.

NOTE 2 – If **view_volume_sides_clipping** has a value of FALSE, the two–dimensional projection may extend beyond the boundaries of the **viewport** specified by the **camera_image**.

view_window: a rectangle on the **view_plane**. The representation is mapped to this rectangle which is then mapped to the **viewport** of a **camera_image**.

Informal propositions:

IP1: The plane represented by the **front_plane_distance** shall be closer to the **projection_point** than the plane represented by the **back_plane_distance**.

IP2: The rectangle indicated by the **view_window** attribute shall be in the view plane.

IP3: The **projection_point** shall not lie in the view plane.

4.5.6 camera_model_d3_with_hlhsr

A **camera_model_d3_with_hlhsr** is a **camera_model_d3** that indicates whether hidden line and hidden surface removal shall be performed.

EXPRESS specification:

```
*)
ENTITY camera_model_d3_with_hlhsr
  SUBTYPE OF (camera_model_d3);
  hidden_line_surface_removal : BOOLEAN;
END_ENTITY;
(*
```

Attribute definitions:

hidden_line_surface_removal: an indication of whether hidden lines and hidden surfaces shall be removed while projecting a three–dimensional representation. A value of TRUE indicates that hidden lines and hidden surfaces shall be removed during projection. A value of FALSE indicates that hidden lines and hidden surfaces shall not be removed during projection.

4.5.7 camera_model_d3_multi_clipping

A **camera_model_d3_multi_clipping** contains the information needed to create a projection from a representation to a picture of that representation using planes in the coordinate space of the projected representation to clip the geometry of the representation prior to projection.

EXPRESS specification:

```
*)
ENTITY camera_model_d3_multi_clipping
  SUBTYPE OF (camera_model_d3);
  shape_clipping : SET [1:?] OF plane;
END_ENTITY;
(*
```

Attribute definitions:

shape_clipping: The planes that bound the clipping region. Each plane defines an acceptance region, which is the infinite halfspace defined by the plane and its positive normal. The clipping region is defined as the intersection of all acceptance regions.

NOTES

1 – The clipping region defined by **shape_clipping** may be infinite.

2 – Clipping is performed prior to projection.

4.5.8 camera_model_with_light_sources

A **camera_model_with_light_sources** contains the information needed to create a projection from a representation to a picture of that representation with additional information about the light sources to be used for shading.

EXPRESS specification:

```
*)
ENTITY camera_model_with_light_sources
  SUBTYPE OF (camera_model_d3);
  sources : SET [1:?] OF light_source;
END_ENTITY;
(*
```

Attribute definitions:

sources: a set of **light_source**s which define the lighting model of the representation being projected.

4.5.9 light_source

A **light_source** affects the display of surfaces. Lighting is applied on a surface by surface basis, i.e., no interactions between surfaces are defined.

EXPRESS specification:

```
*)
ENTITY light_source
  SUPERTYPE OF (ONEOF(light_source_ambient,
                      light_source_directional,
                      light_source_positional,
                      light_source_spot))
  SUBTYPE OF (geometric_representation_item);
  light_colour : colour;
WHERE
  WR1: SIZEOF(USEDIN(SELF,'PRESENTATION_APPEARANCE_SCHEMA.'+
                          'STYLED_ITEM')) = 0;
END_ENTITY;
(*
```

Attribute definitions:

light_colour: the colour of the light to be used for shading.

Formal propositions:

WR: A **light_source** shall not be referenced by a **styled_item** entity.

4.5.10 light_source_ambient

A **light_source_ambient** affects a surface independent of the orientation and position of the surface.

EXPRESS specification:

```
*)
ENTITY light_source_ambient
  SUBTYPE OF (light_source);
END_ENTITY;
(*
```

4.5.11 light_source_directional

A **light_source_directional** affects a surface based on its orientation but independent of the position of the surface. All the rays of light are parallel to the direction.

NOTE – Figure 9 shows the definition of **light_source_directional**.

EXPRESS specification:

```
*)
ENTITY light_source_directional
  SUBTYPE OF (light_source);
  orientation : direction;
END_ENTITY;
(*
```

Attribute definitions:

orientation: the direction of the **light_source** in the coordinate space of the representation being projected.

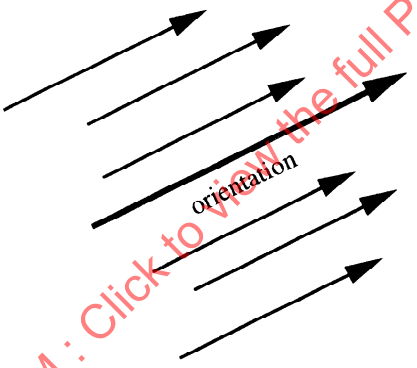


Figure 9 – Light source directional

4.5.12 light_source_positional

A **light_source_positional** affects a surface based on the orientation and position of the surface.

NOTE 1 – Figure 10 shows the definition of **light_source_positional**.

EXPRESS specification:

```
*)
ENTITY light_source_positional
  SUBTYPE OF (light_source);
  position             : cartesian_point;
  constant_attenuation : REAL;
  distance_attenuation : REAL;
END_ENTITY;
(*
```

Attribute definitions:

position: the position of the **light_source** in the coordinate space of the representation being projected.

constant_attenuation: the value of attenuation in the reflectance equation that is constant.

distance_attenuation: the value of attenuation in the reflectance equation that is proportional to the distance from the **light_source**.

NOTE 2 – Examples of reflectance equations can be found in Annex D of this part of ISO 10303.

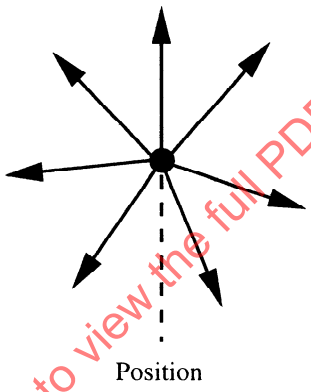


Figure 10 – Light source positional

4.5.13 light_source_spot

A **light_source_spot** affects a surface based on the position and direction of the surface and the cone of influence of the light source. The cone of influence is determined by the position, orientation, and **spread_angle** of the **light_source_spot**. Only those parts of an object that lie within the cone of influence will be affected by a **light_source_spot**.

NOTE 1 – Figure 11 shows the definition of **light_source_spot**.

EXPRESS specification:

```
*)
ENTITY light_source_spot
  SUBTYPE OF (light_source);
  position               : cartesian_point;
  orientation            : direction;
  concentration_exponent : REAL;
  constant_attenuation   : REAL;
  distance_attenuation   : REAL;
  spread_angle           : positive_plane_angle_measure;
```

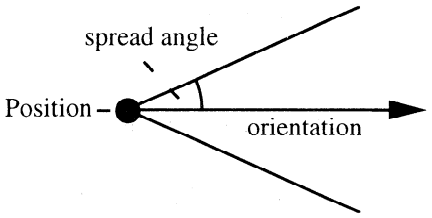


Figure 11 – Light source spot

```
END_ENTITY;
(*
```

Attribute definitions:

position: the position of the **light_source** in the coordinate space of the representation being projected.

orientation: The direction of the axis of the cone of influence of the **light_source** specified in the coordinate space of the representation being projected.

concentration_exponent: the exponent on the cosine of the angle between the line that starts at the position of the spot **light_source** and is in the direction of the orientation of the spot **light_source** and a line that starts at the position of the spot **light_source** and goes through a point on the surface being shaded. The positions on the surface involved in the reflectance equation are determined by the **surface_style_rendering_with_properties**.

constant_attenuation: the value of the attenuation in the reflectance equation that is constant.

distance_attenuation: the value of the attenuation in the reflectance equation that is proportional to the distance from the light_source.

spread_angle: one half of the apex angle of the cone of influence.

Informal propositions:

IP1: The spread angle shall not be greater than 180 degrees.

NOTE 2 – Examples of reflectance equations can be found in Annex D of this part of ISO 10303.

4.5.14 camera_image

A **camera_image** is the result of projecting two– or three–dimensional geometry where the projection transformation is the mapping of the **camera_model** to the viewport.

EXPRESS specification:

```
*)
ENTITY camera_image
  SUBTYPE OF (mapped_item);
WHERE
  WR1: 'PRESENTATION_ORGANIZATION_SCHEMA.CAMERA_USAGE'
       IN TYPEOF (SELF\mapped_item.mapping_source);
  WR2: 'PRESENTATION_RESOURCE_SCHEMA.PLANAR_BOX'
       IN TYPEOF (SELF\mapped_item.mapping_target);
  WR3: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM'
       IN TYPEOF (SELF);
END_ENTITY;
(*
```

Attribute definitions:

SELF\mapped_item.mapping_source: a **camera_usage** which is the **representation** that is projected plus the **camera_model** which is the source part of the projection mapping.

SELF\mapped_item.mapping_target: a **planar_box** onto which the view window associated with a two– or three–dimensional **camera_model** is projected.

Formal propositions:

WR1: The source of the mapping shall be a **camera_usage**.

WR2: The target of the mapping shall be an **axis2_placement**.

WR3: The **camera_image** shall be a **geometric_representation_item**.

4.5.15 camera_usage

A **camera_usage** is the association between two– or three–dimensional representation, and the origin for the mapping. The **camera_model** is the source part of the projection mapping.

EXPRESS specification:

```
*)
ENTITY camera_usage
  SUBTYPE OF (representation_map);
WHERE
  WR1: NOT ('PRESENTATION_ORGANIZATION_SCHEMA.PRESENTATION_REPRESENTATION'
       IN TYPEOF(SELF\representation_map.mapped_representation));
  WR2: 'PRESENTATION_ORGANIZATION_SCHEMA.CAMERA_MODEL'
       IN TYPEOF (SELF\representation_map.mapping_origin);
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_map.mapping_origin: the **camera_model** which is the source part of the projection mapping.

Formal propositions:

WR1: The **mapped_representation** shall not be a **presentation_representation**.

WR2: The origin of the **camera_usage** shall be **camera_model**.

4.6 Presentation organization schema entity definitions: layers

4.6.1 presentation_layer_assignment

A **presentation_layer_assignment** assigns an identifier to a set of **presentation_representations** or **representation_items**. This set contains the pictures, or elements of pictures, that are assigned to a layer. A **representation_item** that is assigned to a layer shall be a **styled_item**, or shall be defined by one or more **styled_items**. In the case of a **presentation_representation**, the assignment also associates the layer identifier with each styled **representation_item** that is contained in or referenced by that **presentation_representation**, unless this assignment is over-ridden by a **representation_item_dependent_layer_assignment**. In the case of a **representation_item**, the assignment also associates the layer identifier with each styled **representation_item** that is referenced directly or indirectly by that item, unless this assignment is over-ridden by a **representation_item_dependent_layer_assignment**.

NOTE – This assignment is for the purpose of determining visibility and style in a picture represented by a **presentation_representation** through the use of the **presentation_layer_usage** entity.

EXPRESS specification:

```
*)
ENTITY presentation_layer_assignment;
  name           : label;
  description    : text;
  assigned_items : SET [1:?] OF layered_item;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the layer defined by the **presentation_layer_assignment** is referred to.

description: text that relates the nature of the **presentation_layer_assignment**.

assigned_items: the set of items assigned to the layer defined by the **presentation_layer_assignment**.

4.6.2 representation_item_dependent_layer_assignment

A **representation_item_dependent_layer_assignment** is a **presentation_layer_assignment** that assigns a layer to a **presentation_representation** or a styled **representation_item** only as it participates in the definition of a **representation_item**. This assignment over-rides

any layer assignment for the **presentation_representation** or styled **representation_item** through its participation in the definition of the **representation_item.**

EXAMPLE 5 – In a draughting application, a dimension consists of several occurrences of curves and text. The dimension is assigned to layer 'DIMENSION', while the text that presents the value of the dimension is assigned to layer 'DIMENSION VALUE'. To indicate that the dimension is assigned to layer 'DIMENSION', a **presentation_layer_assignment** includes the dimension as a member of the **assigned_items** set. To indicate that the text is on layer 'DIMENSION VALUE', a **representation_item_dependent_layer_assignment** includes the text as a member of the **assigned_items** set, and specifies the dimension as the context for the assignment.

EXPRESS specification:

```
*)
ENTITY representation_item_dependent_layer_assignment
  SUBTYPE OF (presentation_layer_assignment);
  item_context : representation_item;
END_ENTITY;
(*
```

Attribute definitions:

item_context: a **representation_item** that defines the context for the assignment of the **assigned_items** to the layer.

Informal propositions:

IP1: The **assigned_items** shall participate in the definition of the **item_context.**

4.6.3 presentation_layer_usage

A **presentation_layer_usage** relates a **presentation_layer_assignment** to a **presentation_representation** in order to provide a context for the assignment of style and visibility to items assigned to a layer. Many **presentation_representations** may be related to a single **presentation_layer_assignment**; a single **presentation_representation** may be related to many **presentation_layer_assignments.**

NOTE – This allows many different pictures to be assigned to the same layer, and also allows a picture to be assigned to many layers.

EXPRESS specification:

```
*)
ENTITY presentation_layer_usage;
  assignment   : presentation_layer_assignment;
  presentation : presentation_representation;
UNIQUE
  UR1: assignment, presentation;
END_ENTITY;
(*
```

Attribute definitions:

assignment: the layer to be presented in the **presentation_representation** referenced by the **presentation** attribute.

presentation: the **presentation_representation** that contains the presentation of the layer referenced by the **assignment** attribute.

Formal propositions:

UR1: The pair (**assignment**, **presentation**) shall be unique. A layer shall not be presented in a picture more than once.

4.7 Presentation organization schema entity definitions: association of presentation and product model

4.7.1 presented_item_representation

A **presented_item_representation** is the association of **presentation_representation** or **presentation_set** with the item for which it is a picture.

NOTE – This allows a many–to–many relationship between the representation and presentation.

EXPRESS specification:

```
*)
ENTITY presented_item_representation;
  presentation : presentation_representation_select;
  item         : presented_item;
END_ENTITY;
(*
```

Attribute definitions:

presentation: a **presentation_representation** or a **presentation_set**.

item: the item which is presented in the picture.

4.7.2 presented_item

A **presented_item** is the identification of the subject of a picture. The specification of the subject of a picture is provided in an Application Protocol.

EXAMPLE 6 – In an Application Protocol, the properties associated with a specific product definition are presented. The Application Protocol completes the **presented_item** construct to relate the presentation to the **product_definition**.

EXPRESS specification:

```
*)
ENTITY presented_item
  ABSTRACT SUPERTYPE;
END_ENTITY;
(*
```

4.8 Presentation organization schema rule definitions

4.8.1 symbol_representation_rule

The **symbol_representation_rule** ensures that a **presentation_representation_relationship**, which relates two **symbol_representations**, is a **symbol_representation_relationship**.

EXPRESS specification:

```
*)
RULE symbol_representation_rule
  FOR (presentation_representation_relationship);
WHERE
  WR1: SIZEOF(QUERY(each_1 <* presentation_representation_relationship  |
               NOT ('PRESENTATION_DEFINITION_SCHEMA.'+
               'SYMBOL_REPRESENTATION_RELATIONSHIP' IN TYPEOF(each_1)) AND
         (SIZEOF(QUERY(each_2 <* [each_1\representation_relationship.rep_1,
                                  each_1\representation_relationship.rep_2] |
         'PRESENTATION_DEFINITION_SCHEMA.SYMBOL_REPRESENTATION' IN TYPEOF(each_2)
         )) > 0)
       )) = 0;
END_RULE;
(*
```

Formal propositions:

WR1: presentation_representation_relationships which are not **symbol_representation_relationships** cannot relate **symbol_representations**.

4.9 Presentation organization schema function definitions

4.9.1 acyclic_presentation_representation_relationship

The **acyclic_presentation_representation_relationship** function checks whether the graph defined by a **presentation_representation_relationship**, the related **presentation_representation**, and all **presentation_representation_relationships** referring to the related **presentation_representation** is acyclic. The function returns TRUE if the graph is acyclic; and FALSE if it is cyclic.

EXPRESS specification:

```
*)
FUNCTION acyclic_presentation_representation_relationship
  ( relation : presentation_representation_relationship;
    children : SET OF presentation_representation ) : BOOLEAN;

  LOCAL
    x : SET OF presentation_representation_relationship;
    i : INTEGER;
    local_children : SET OF presentation_representation;
  END_LOCAL;
```

```
  REPEAT i:=1 TO HIINDEX(children);
    IF relation\representation_relationship.rep_1 :=: children[i] THEN
       RETURN(FALSE);
    END_IF;
  END_REPEAT;

  x := USEDIN ( relation\representation_relationship.rep_1,
                  'REPRESENTATION_SCHEMA.'+
                  'REPRESENTATION_RELATIONSHIP.REP_2');
  local_children := children + relation\representation_relationship.rep_1;

  IF SIZEOF (x) > 0 THEN
    REPEAT i:=1 TO HIINDEX (x);
       IF NOT acyclic_presentation_representation_relationship
              (x[i] , local_children) THEN
         RETURN (FALSE);
       END_IF;
    END_REPEAT;
  END_IF;

  RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

relation: the **presentation_representation_relationship** which is tested. This is input to the function.

children: the **presentation_representations** referenced by **relation**. This is input to the function. On initial input this set contains as its only element the **rep_2** of the **relation**.

EXPRESS specification:

```
*)
END_SCHEMA; -- presentation_organization_schema
(*
```

5 Presentation definition

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

EXPRESS specification:

```
*)
SCHEMA presentation_definition_schema;

REFERENCE FROM external_reference_schema
    (externally_defined_item,
     pre_defined_item);
```

```
REFERENCE FROM geometry_schema
    (axis2_placement,
     curve,
     geometric_representation_item,
     point
     );

REFERENCE FROM measure_schema
    (positive_ratio_measure);

REFERENCE FROM presentation_appearance_schema
    (styled_item);

REFERENCE FROM presentation_resource_schema
    (character_glyph_symbol,
     planar_box,
     planar_extent,
     font_select,
     presentable_text,
     text_font);

REFERENCE FROM representation_schema
    (item_in_context,
     mapped_item,
     representation,
     representation_item,
     representation_map,
     representation_relationship,
     representation_relationship_with_transformation,
     using_representations);

REFERENCE FROM support_resource_schema
    (label,
     text);
(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

external_reference_schema	ISO 10303-41
geometry_schema	ISO 10303-42
presentation_appearance_schema	Clause 6 of this part of ISO 10303
presentation_resource_schema	Clause 7 of this part of ISO 10303
representation_schema	ISO 10303-43

2 – The EXPRESS-G diagrams for this schema may be found in Annex E of this part of ISO 10303.

5.1 Introduction

The **presentation_definition_schema** provides the structure for the definition of annotation and assignment of style properties to annotation primitives. Annotation in this part of ISO 10303 is always planar, but may be located in three–dimensional space.

5.2 Fundamental concepts and assumptions

Annotation primitives are the elements from which symbolic presentations are constructed. The annotation primitives defined in this schema include annotation points, curves, fill areas, texts, symbols, and tables.

An annotation point is a point which is presented using a **point_style**. For more information on **point_style** see clause 6. The presentation appearance schema does not define a special entity for annotation points because the **point** entity defined in ISO 10303-42 is sufficient for annotation purposes.

An annotation curve is a planar curve which is presented using a **curve_style**. For more information on **curve_style** see clause 6. The presentation appearance schema does not define a special entity for annotation curves, because the **curve** entity defined in ISO 10303-42 is sufficient for annotation purposes.

Annotation text is a collection of characters, character strings, collection of strings, and more complex collections of strings and characters. Annotation text is defined by the **annotation_text** entity which uses the concept of **mapped_item**. For more information on **mapped_item** see ISO 10303-43. The **annotation_text** entity positions and orients a collection of characters, simple character strings, collection of strings, and more complex collections of strings and characters, which is defined in a **text_string_representation**. Several **annotation_text** entities can reference the same **text_string_representation**. A **text_string_representation** itself is a set of **annotation_text**, **defined_character_glyph**, **annotation_text_character**, **text_literal**, or **composite_text** entities. This structure allows the recursive construction of an **annotation_text**. Characters may be pre–defined, externally defined, or may be defined within a conforming exchange using concepts of this part of ISO 10303. In the last case, the **annotation_text_character** entity refers to the **character_glyph_symbol** entity. This entity contains the geometric representation of a character. The **text_literal** allows the specification of a text string as part of an **annotation_text**. A complex collection that may be placed and styled as a whole can be defined by the **composite_text** entity. Specializations of **annotation_text** allow additionally the specification of blanking boxes, surrounding curves, delineations, or extent rectangles for that text. An **annotation_text** is presented using a **text_style**. For more information on **text_style** see clause 6.

An annotation symbol is a pre–defined symbol, an externally defined symbol, or a collection of **representation_items** which make up a graphical symbol. An annotation symbol is defined either by the **defined_symbol** entity or by the **annotation_symbol** entity. The **defined_symbol** entity scales, positions, and orients an implicit description of a symbol. The **annotation_symbol** uses the concept of **mapped_item**. For more information on **mapped_item** see ISO 10303-43. The **annotation_symbol** entity scales, positions, and orients a a collection of **representation_items** which are defined in a **symbol_representation**. Several **annotation_symbol** entities can reference the same **symbol_representation**. The **representation_items**

which make up a symbol can be elements of geometry (see ISO 10303-42), annotation primitives, or annotation occurrences. This structure allows the recursive construction of symbols. Moreover, a **symbol_representation** itself can be built of several **symbol_representations**. Such a hierarchy is defined by establishing a relationship between two **symbol_representations**. A relationship is established by instantiating an entity of type **symbol_representation_relationship** which refers to a pair of representations. One of these representations is called **rep_1**, the other one is called **rep_2**, and the relationship is directed from the parent (**rep_1**) to the child (**rep_2**). The relationship specifies additionally a transformation. This transformation shall be executed to transform elements in the **rep_2** to the coordinate system of the **rep_1**. A single **symbol_representation** is then the collection of all **representation_items** in that representation plus all **symbol_representations** which are nodes in the relationship tree associated with that representation. Symbols are presented using **symbol_style**, which is a collection of presentation styles. For more information on **symbol_style** see clause 6.

An annotation table is a special type of **annotation_symbol** which represents a table. An annotation table is defined by the **annotation_table** entity which uses the concept of **mapped_item**. For more information on **mapped_item** see ISO 10303-43. The **annotation_table** entity scales, positions, and orients a table which is defined in a **table_representation**. Several **annotation_table** entities can reference the same **table_representation**. A **table_representation** can be built of **table_record_representations** which itself can be made of several **table_record_field_representations**. A **table_record_representation** usually corresponds to a column or row of a table, while a **table_record_field_representation** corresponds to a single cell of a table. Nevertheless, more complex structures can be built because **table_record_field_representations** may be built of other **table_record_field_representations**. The hierarchy making up a **table_representation** is built by relating the components by a **table_representation_relationship** which is a specialization of **symbol_representation_relationship**. The **annotation_table** defines only an empty table. To put text into a table, **annotation_text** occurrences are defined individually in such a way that they are placed in a table. The **table_text_relationship** can be used to associate such text with a specific field of a table.

Annotation primitives can occur only in conjunction with a style assignment. An annotation primitive, together with its style, is called an **annotation_occurrence**. For each type of primitive a special **annotation_occurrence** is defined which restricts the presentation style to appropriate types. The **annotation_occurrence_relationship** allows the definition of a relationship between two **annotation_occurrences**. The **table_text_relationship** associates an **annotation_text_occurrence** with a **table_record_field_representation** of an annotation table.

5.3 Presentation definition schema type definitions

5.3.1 text_delineation

The **text_delineation** type is provided to control the delineation of text.

NOTE – Application protocols may specify legal values of **text_delineation** and associate precise meaning with those values.

EXPRESS specification:

```
*)
TYPE text_delineation = label;
END_TYPE;
(*
```

EXAMPLE 7 – An application protocol may specify that the only permitted values of this type are 'underline' and 'overline', and associate these with the delineation of text as shown in figure 12.

Figure 12 – Examples of text delineation

5.3.2 defined_symbol_select

The **defined_symbol_select** specifies the implicit description of a **defined_symbol**.

EXPRESS specification:

```
*)
TYPE defined_symbol_select = SELECT
  (pre_defined_symbol,
   externally_defined_symbol);
END_TYPE;
(*
```

5.3.3 text_or_character

The **text_or_character** specifies the items which can be used in an **annotation_text** or a **composite_text**.

EXPRESS specification:

```
*)
TYPE text_or_character = SELECT
  (annotation_text,
   annotation_text_character,
   defined_character_glyph,
   composite_text,
   text_literal);
END_TYPE;
(*
```

5.3.4 text_alignment

The **text_alignment** is provided to control the alignment of text.

NOTE – Application protocols shall specify legal values of the **text_alignment** and shall associate precise meaning with those values.

EXPRESS specification:

```
*)
TYPE text_alignment = label;
END_TYPE;
(*
```

EXAMPLE 8 – An Application Protocol may specify that the only permitted values of this type are 'left', 'centre' and 'right', and associate these to the positioning of text with respect to alignment points as shown in figure 13.

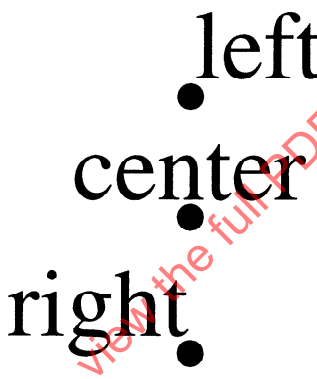


Figure 13 – Examples of text alignment

5.3.5 defined_glyph_select

The **defined_glyph_select** is a selection between a **pre_defined_character_glyph** and an **externally_defined_character_glyph**.

EXPRESS specification:

```
*)
TYPE defined_glyph_select = SELECT
  (pre_defined_character_glyph,
   externally_defined_character_glyph);
END_TYPE;
(*
```

5.3.6 text_path

The **text_path** specifies the direction of the location of the next text character relative to the current character. The next text character can appear to the left of the current character, to the right of the current character, above the current character, or below the current character.

EXPRESS specification:

```
*)
TYPE text_path = ENUMERATION OF
  (left,
   right,
   up,
   down);
END_TYPE;
(*
```

Enumerated item definitions:

left: The next character is placed to the left of the current character.

right: The next character is placed to the right of the current character.

up: The next character is placed above the current character.

down: The next character is placed below the current character.

5.4 Presentation definition schema entity definitions: annotation primitives

5.4.1 annotation_fill_area

An **annotation_fill_area** is a set of curves that may be filled with hatching, shading, colour, or tiling. The **annotation_fill_area** is described by boundaries which consist of non–intersecting, non–self–intersecting closed curves. These curves form the boundary of planar areas to be filled according to the style for the **annotation_fill_area**. The filling is defined by the following rules:

– A curve that is not surrounded by any other curve is a border between an unfilled area on the outside and a filled area on the inside.

NOTE 1 – see figure 14a.

– A curve surrounds an unfilled area if it is surrounded by another curve whose inside is a filled area.

NOTE 2 – see figure 14b.

– If a third curve is placed inside of the second curve this curve surrounds a filled area.

NOTE 3 – see figure 14c.

– For each additional curve the procedure is applied in the same manner.

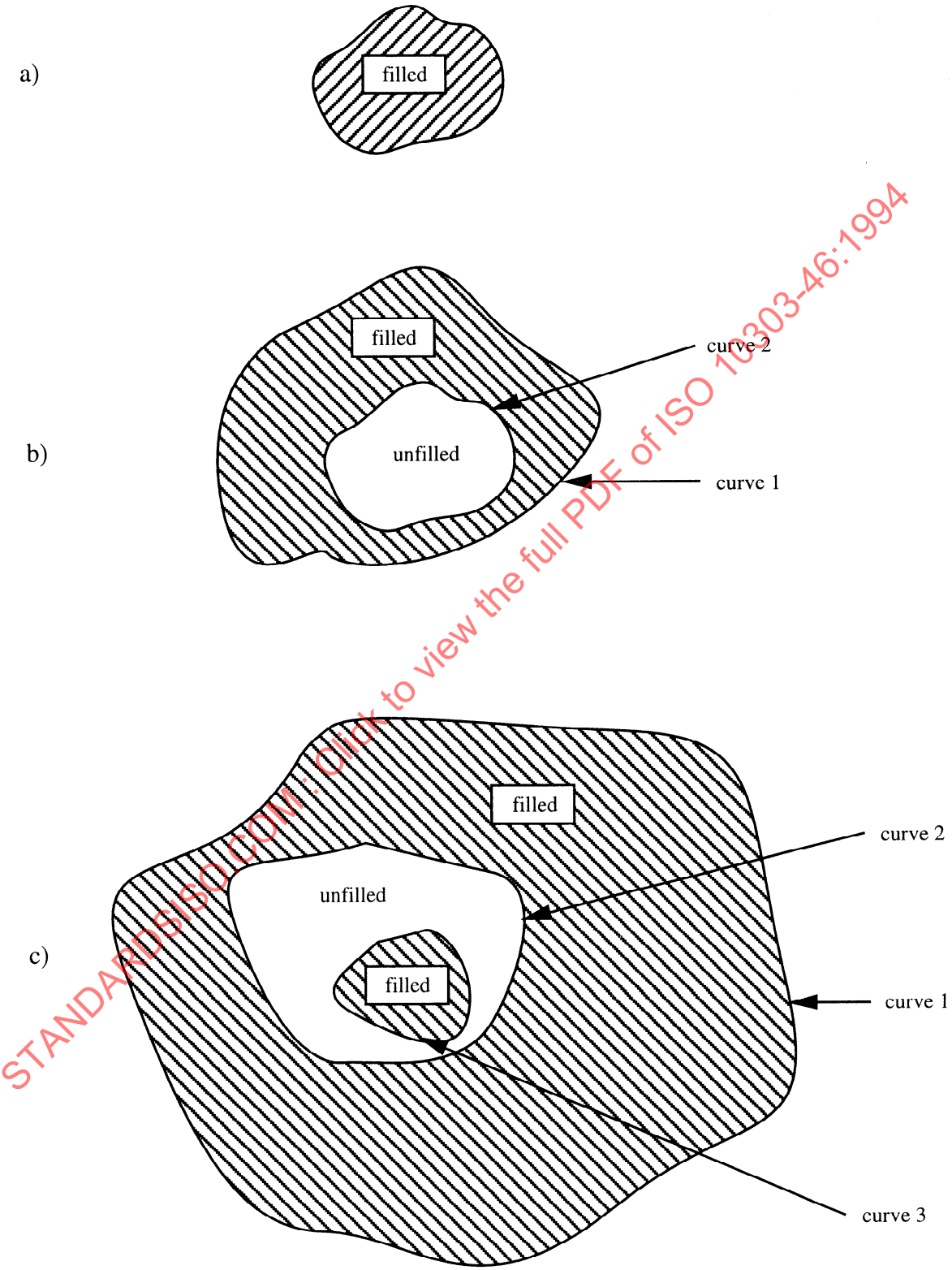


Figure 14 – Filling of annotation fill areas

EXPRESS specification:

```
*)
ENTITY annotation_fill_area
  SUBTYPE OF (geometric_representation_item);
  boundaries : SET [1:?] OF curve;
END_ENTITY;
(*
```

Attribute definitions:

boundaries: a set of curves that define the boundaries of the fill area.

Informal propositions:

IP1: All the curves in the set **boundaries** shall be closed and planar.

IP2: If there are two or more curves in the set **boundaries**, all of these curves shall be coplanar, and no two curves shall intersect each other.

5.4.2 defined_symbol

A **defined_symbol** is a symbol that has an implicit definition, either through a **pre_defined_symbol** or an **externally_defined_symbol**.

EXPRESS specification:

```
*)
ENTITY defined_symbol
  SUBTYPE OF(geometric_representation_item);
  definition : defined_symbol_select;
  target     : symbol_target;
END_ENTITY;
(*
```

Attribute definitions:

definition: an implicit description of a symbol, either pre-defined or externally defined.

target: a description of the scaling, placement, and orientation of the **defined_symbol**.

5.4.3 defined_table

A **defined_table** is a **defined_symbol** that specifies an implicit definition of a table.

EXPRESS specification:

```
*)
ENTITY defined_table
  SUBTYPE OF(defined_symbol);
END_ENTITY;
(*
```

5.4.4 symbol_target

A **symbol_target** is the target of the transformation which determines the location and orientation of a **symbol_representation** used as an **annotation_symbol**, or the target of a **defined_symbol**. It consists of an oriented position and a scaling in x and y.

EXPRESS specification:

```
*)
ENTITY symbol_target
  SUBTYPE OF (geometric_representation_item);
  placement          : axis2_placement;
  x_scale            : positive_ratio_measure;
  y_scale            : positive_ratio_measure;
END_ENTITY;
(*
```

Attribute definitions:

placement: The location and orientation of a **symbol_representation** or of a **defined_symbol** used as an **annotation_symbol**.

x_scale: The scale applied to the x coordinates of the **symbol_representation** or of the **defined_symbol**.

y_scale: The scale applied to the y coordinates of the **symbol_representation** or of the **defined_symbol**.

5.4.5 pre_defined_symbol

A **pre_defined_symbol** allows a conforming exchange to define an application–specific symbol. The actual symbol shall be defined by an Application Protocol.

EXPRESS specification:

```
*)
ENTITY pre_defined_symbol
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

5.4.6 externally_defined_symbol

An **externally_defined_symbol** is an **externally_defined_item** that makes an external reference to a symbol.

EXPRESS specification:

```
*)
ENTITY externally_defined_symbol
  SUBTYPE OF (externally_defined_item);
END_ENTITY;
(*
```

5.4.7 annotation_symbol

An **annotation_symbol** is the mapping of a **symbol_representation** as a **geometric_representation_item** to present that **symbol_representation** as part of a picture.

NOTE – Figure 15 shows examples of annotation symbols.

EXPRESS specification:

```
*)
ENTITY annotation_symbol
  SUBTYPE OF(mapped_item);
WHERE
  WR1: 'PRESENTATION_DEFINITION_SCHEMA.SYMBOL_REPRESENTATION_MAP' IN
         TYPEOF (SELF\mapped_item.mapping_source);
  WR2: 'PRESENTATION_DEFINITION_SCHEMA.SYMBOL_TARGET' IN
         TYPEOF (SELF\mapped_item.mapping_target);
  WR3: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN
         TYPEOF (SELF);
END_ENTITY;
(*
```

Attribute definitions:

SELF\mapped_item.mapping_source: a **symbol_representation_map** which maps the **symbol_representation**

SELF\mapped_item.mapping_target: a **symbol_target** indicating where the symbol is to be placed.

Formal propositions:

WR1: The **mapping_source** shall be a **symbol_representation_map**.

WR2: The **mapping_target** shall be a **symbol_target**.

WR3: An instance of **annotation_symbol** shall also be an instance of **geometric_representation_item**.

5.4.8 annotation_table

An **annotation_table** is the mapping of a **table_representation** to present the **table_representation** as part of a picture.

EXPRESS specification:

```
*)
ENTITY annotation_table
  SUBTYPE OF(annotation_symbol);
WHERE
  WR1: 'PRESENTATION_DEFINITION_SCHEMA.TABLE_REPRESENTATION' IN
         TYPEOF (SELF\mapped_item.mapping_source.mapped_representation);
END_ENTITY;
(*
```

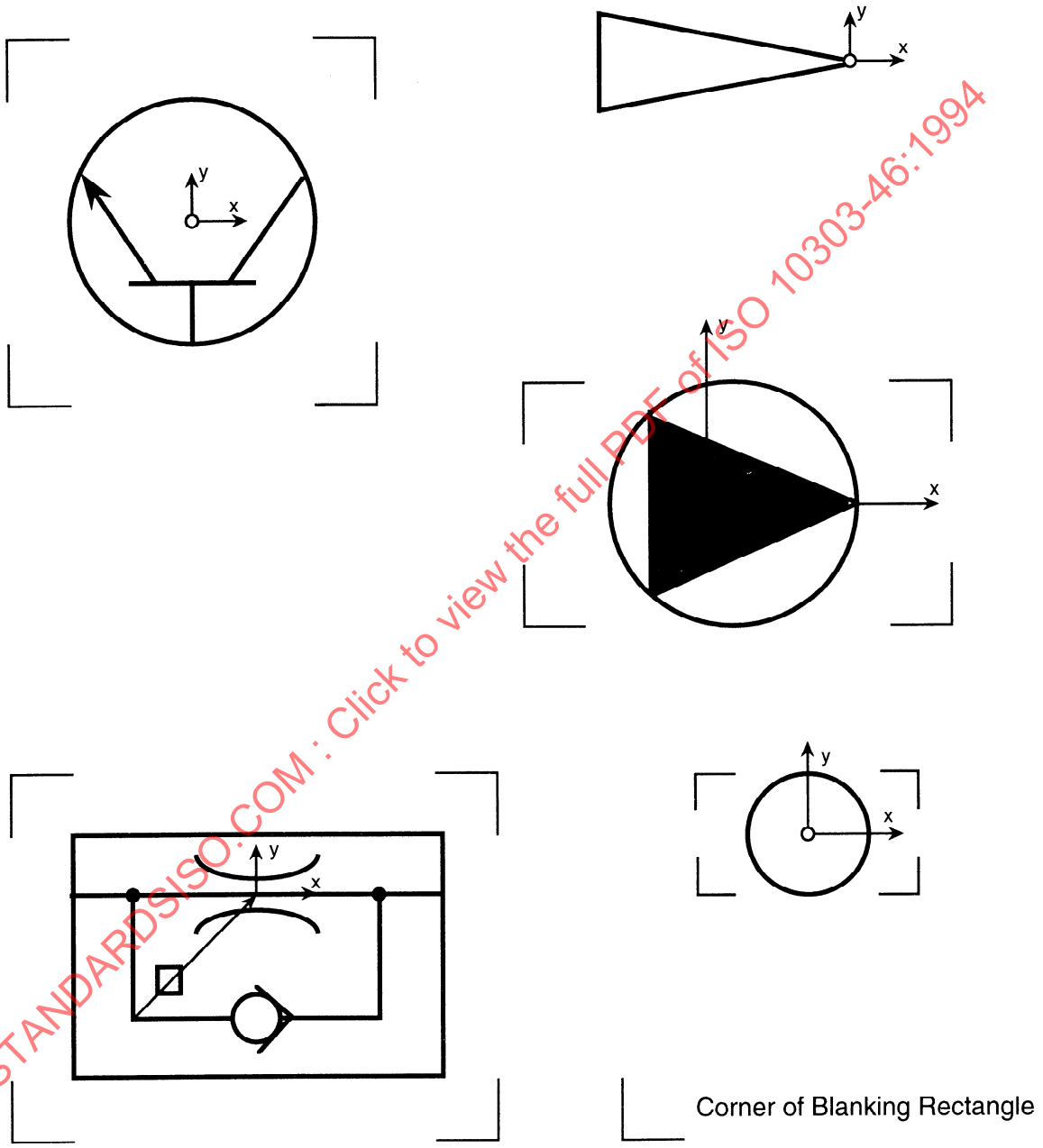


Figure 15 – Examples of annotation symbols

Formal propositions:

WR1: The **representation** that is mapped shall be a **table_representation**.

5.4.9 symbol_representation_map

The **symbol_representation_map** is a **representation_map** which maps a **symbol_representation** to an **annotation_symbol**.

EXPRESS specification:

```
*)
ENTITY symbol_representation_map
  SUBTYPE OF (representation_map);
WHERE
  WR1: 'PRESENTATION_DEFINITION_SCHEMA.SYMBOL_REPRESENTATION' IN
         TYPEOF (SELF\representation_map.mapped_representation);
  WR2: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT' IN
         TYPEOF (SELF\representation_map.mapping_origin);
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_map.mapped_representation: the **symbol_representation** to be included in an **annotation_symbol**.

SELF\representation_map.mapping_origin: an **axis2_placement** defining the origin of the mapping.

Formal propositions:

WR1: The **mapped_representation** shall be a **symbol_representation**.

WR2: The **mapping_origin** shall be an **axis2_placement**.

5.4.10 symbol_representation

A **symbol_representation** is a kind of **representation** used in the assembly of a symbol. It allows also the definition of annotation_tables which are a specialization of **annotation_symbols**. **Symbol_representations** may only be related to each other using **symbol_representation_relationships**.

EXPRESS specification:

```
*)
ENTITY symbol_representation
  SUBTYPE OF (representation);
END_ENTITY;
(*
```

5.4.11 symbol_representation_with_blanking_box

The **symbol_representation_with_blanking_box** is a **symbol_representation** that includes a box which blanks out all other elements in the box.

EXPRESS specification:

```
*)
ENTITY symbol_representation_with_blanking_box
  SUBTYPE OF (symbol_representation);
  blanking : planar_box;
WHERE
  WR1: item_in_context (SELF.blanking, SELF\representation.context_of_items);
END_ENTITY;
(*
```

Attribute definitions:

blanking: a box which blanks out any element in the box with exception of the symbol itself.

Formal propositions:

WR1: The blanking box shall be in the context **SELF\context_of_items**.

5.4.12 table_representation

A **table_representation** is a kind of symbol used for creating tables. **Table_representations** may only be related with **table_representation_relationship** entities. If a **table_representation** plays the role of **rep_1** in a **table_representation_relationship**, only a **table_record_representation** may play the role of **rep_2**.

EXPRESS specification:

```
*)
ENTITY table_representation
  SUBTYPE OF (symbol_representation);
END_ENTITY;
(*
```

5.4.13 table_record_representation

A **table_record_representation** is a kind of symbol used for creating records within tables. **Table_record_representations** may only be related by **table_representation_relationships**. If a **table_record_representation** plays the role of **rep_1** in a **table_representation_relationship**, only a **table_record_field_representation** may play the role of **rep_2**.

EXPRESS specification:

```
*)
ENTITY table_record_representation
  SUBTYPE OF (symbol_representation);
WHERE
  WR1: (SIZEOF(USEDIN(SELF, 'REPRESENTATION_SCHEMA.'+
                            'REPRESENTATION_RELATIONSHIP.REP_2')) > 0)
                       OR
```

```
      (SIZEOF( QUERY( map_item <* USEDIN(SELF, 'REPRESENTATION_SCHEMA.'+
                                              'MAPPED_ITEM.MAPPING_SOURCE.'+
                                              'MAPPED_REPRESENTATION') |
        'PRESENTATION_DEFINITION_SCHEMA.TABLE_REPRESENTATION' IN
        TYPEOF (using_representations (map_item)) )) > 0);
END_ENTITY;
(*
```

Formal propositions:

WR1: Either a **table_record_representation** shall be used as a **rep_2** in a **table_representation_relationship**, or it shall be mapped into a **table_representation**.

5.4.14 table_record_field_representation

A **table_record_field_representation** is a kind of symbol used for creating fields within records of a table. **Table_record_field_representations** may only be related by **table_representation_relationship** entities. If a **table_record_field_representation** plays the role of **rep_1** in a **table_representation_relationship**, only a **table_record_field_representation** may play the role of **rep_2**.

EXPRESS specification:

```
*)
ENTITY table_record_field_representation
  SUBTYPE OF (symbol_representation);
WHERE
  WR1: (SIZEOF(USEDIN(SELF, 'REPRESENTATION_SCHEMA.'+
                            'REPRESENTATION_RELATIONSHIP.REP_2')) > 0)
                       OR
       (SIZEOF( QUERY( map_item <* USEDIN(SELF, 'REPRESENTATION_SCHEMA.'+
                                               'MAPPED_ITEM.MAPPING_SOURCE.'+
                                               'MAPPED_REPRESENTATION') |
                                               'PRESENTATION_DEFINITION_SCHEMA.' +
                                               'TABLE_RECORD_REPRESENTATION' IN
         TYPEOF (using_representations (map_item)) )) > 0);
END_ENTITY;
(*
```

Formal propositions:

WR1: Either a **table_record_field_representation** shall be used as a **rep_2** in a **table_representation_relationship**, or it shall be mapped into a **table_record_representation**.

5.4.15 table_record_field_representation_with_clipping_box

A **table_record_field_representation_with_clipping_box** is a **table_record_field_representation** which includes a clipping box to blank out all elements outside of the box.

EXPRESS specification:

```
*)
ENTITY table_record_field_representation_with_clipping_box
  SUBTYPE OF (table_record_field_representation);
```

```
  clipping_box : planar_box;
WHERE
   WR1: item_in_context (SELF.clipping_box,
                         SELF\representation.context_of_items);
END_ENTITY;
(*
```

Attribute definitions:

clipping_box: a **planar_box** that defines the boundary for clipping the contents of the **table_record_field_representation**.

Formal propositions:

WR1: The **clipping_box** shall be in the same context as the **table_record_field_representation_with_clipping_box**.

5.4.16 symbol_representation_relationship

A **symbol_representation_relationship** is a kind of **representation_relationship_with_transformation** used to relate **symbol_representations**.

EXPRESS specification:

```
*)
ENTITY symbol_representation_relationship
  SUBTYPE OF (representation_relationship_with_transformation);
WHERE
  WR1: acyclic_symbol_representation_relationship (SELF,
                                                   [SELF\representation_relationship.
                                                        rep_2]);
  WR2: 'PRESENTATION_DEFINITION_SCHEMA.SYMBOL_REPRESENTATION' IN
         TYPEOF (SELF\representation_relationship.rep_1);
  WR3: 'PRESENTATION_DEFINITION_SCHEMA.SYMBOL_REPRESENTATION'IN
          TYPEOF (SELF\representation_relationship.rep_2);
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation_relationship.rep_1: a **symbol_representation** that plays the role of a parent (root) in a tree of **symbol_representations**.

SELF\representation_relationship.rep_2: a **symbol_representation** that plays the role of a child (leaf) in a tree of **symbol_representations**.

Formal propositions:

WR1: A **symbol_representation_relationship** shall not participate in a tree of **symbol_representations** where the root of the tree is also a leaf of its own tree.

WR2: The **rep_1** attribute of a **symbol_representation_relationship** shall be a **symbol_representation**.

WR2: The **rep_2** attribute of a **symbol_representation_relationship** shall be a **symbol_representation**.

NOTE – The attributes **rep_1** and **rep_2** are defined in ISO 10303-43.

5.4.17 table_representation_relationship

The **table_representation_relationship** is a kind of **symbol_representation_relationship** used to relate **table_representations**, **table_record_representations**, and **table_record_field_representations**.

EXPRESS specification:

```
*)
ENTITY table_representation_relationship
  SUBTYPE OF (symbol_representation_relationship);
WHERE
  WR1: NOT ('PRESENTATION_DEFINITION_SCHEMA.TABLE_RECORD_REPRESENTATION' IN
            TYPEOF (SELF\representation_relationship.rep_1))
          XOR
       ('PRESENTATION_DEFINITION_SCHEMA.TABLE_RECORD_FIELD_REPRESENTATION' IN
          TYPEOF (SELF\representation_relationship.rep_2));
  WR2: NOT ('PRESENTATION_DEFINITION_SCHEMA.TABLE_REPRESENTATION' IN
             TYPEOF (SELF\representation_relationship.rep_1))
          XOR
       ('PRESENTATION_DEFINITION_SCHEMA.TABLE_RECORD_REPRESENTATION' IN
          TYPEOF (SELF\representation_relationship.rep_2));
  WR3: NOT ('PRESENTATION_DEFINITION_SCHEMA.TABLE_RECORD_FIELD_REPRESENTATION' IN
            TYPEOF (SELF\representation_relationship.rep_1))
          XOR
       ('PRESENTATION_DEFINITION_SCHEMA.TABLE_RECORD_FIELD_REPRESENTATION' IN
          TYPEOF (SELF\representation_relationship.rep_2));
END_ENTITY;
(*
```

Formal propositions:

WR1: If a **table_record_representation** plays the role of **SELF\representation_relationship.rep_1**, then a **table_record_field_representation** shall play the role of **SELF\representation_relationship.rep_2**.

WR2: If a **table_representation** plays the role of **SELF\representation_relationship.rep_1**, then a **table_record_representation** shall play the role of **SELF\representation_relationship.rep_2**.

WR3: If a **table_record_field_representation** plays the role of **SELF\representation_relationship.rep_1**, then a **table_record_field_representation** shall play the role of **SELF\representation_relationship.rep_2**.

5.4.18 annotation_text

An **annotation_text** is the mapping of a a **text_string_representation** which contains **annotation_text_character**, **text_literal**, **composite_text**, or **annotation_text** entities as a **geometric_representation_item** to present that **text_string_representation** as part of a picture. This structure allows an **annotation_text** to be a simple string of characters, a collection of strings, or a more complex collection of strings and characters.

EXPRESS specification:

```
*)
ENTITY annotation_text
  SUBTYPE OF (mapped_item);
WHERE
  WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT' IN
       TYPEOF( SELF\mapped_item.mapping_target);
  WR2: 'PRESENTATION_DEFINITION_SCHEMA.TEXT_STRING_REPRESENTATION' IN
       TYPEOF( SELF\mapped_item.mapping_source.mapped_representation);
  WR3: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN
       TYPEOF( SELF);
END_ENTITY;
(*
```

Attribute definitions:

SELF\mapped_item.mapping_source: a **representation_map** which maps the **text_string_representation** that supplies text, characters, or **text_literal_symbols** for the **annotation_text**.

SELF\mapped_item.mapping_target: an **axis2_placement** which positions and orients the **annotation_text_map**.

Formal propositions:

WR1: The **mapping_target** shall be an **axis2_placement**.

WR2: The **mapped_representation** shall be a **text_string_representation**.

WR3: An instance of **annotation_text** shall also be an instance of **geometric_representation_item**.

5.4.19 annotation_text_with_extent

An **annotation_text_with_extent** is an **annotation_text** whose extent is explicitly specified.

EXPRESS specification:

```
*)
ENTITY annotation_text_with_extent
  SUBTYPE OF (annotation_text);
  extent : planar_extent;
END_ENTITY;
(*
```

Attribute definitions:

extent: the extent of the **annotation_text** in x and y directions of the local coordinate system defined by the **placement** attribute.

5.4.20 annotation_text_with_delineation

The **annotation_text_with_delineation** is an **annotation_text** that specifies the delineation of the text. The type of delineation and its effect on the appearance of the text is specified in an Application Protocol.

EXPRESS specification:

```
*)
ENTITY annotation_text_with_delineation
  SUBTYPE OF (annotation_text);
  delineation : text_delineation;
END_ENTITY;
(*
```

Attribute definitions:

delineation: the specification of the delineation to be applied to the text.

5.4.21 annotation_text_with_blanking_box

An **annotation_text_with_blanking_box** is an **annotation_text** that contains a blanking box.

EXPRESS specification:

```
*)
ENTITY annotation_text_with_blanking_box
  SUBTYPE OF (annotation_text);
  blanking : planar_box;
END_ENTITY;
(*
```

Attribute definitions:

blanking: a **planar_box** that defines a rectangular area within which only the **annotation_text** is presented.

5.4.22 annotation_text_with_associated_curves

An **annotation_text_with_associated_curves** is an **annotation_text** that contains one ore more **curves**.

EXPRESS specification:

```
*)
ENTITY annotation_text_with_associated_curves
  SUBTYPE OF (annotation_text);
  associated_curves : SET[1:?] of curve;
END_ENTITY;
```

(*

Attribute definitions:

associated_curves: a set of **curves** associated with the **annotation_text**.

NOTE – If the **curves** associated with an **annotation_text_with_associated_curves** are to be presented, then they may be assigned style by **annotation_curve_occurrence**.

5.4.23 text_string_representation

A **text_string_representation** is a **representation** which has a collection of text strings and characters to be mapped to an **annotation_text** entity.

EXPRESS specification:

```
*)
ENTITY text_string_representation
  SUBTYPE OF (representation);
WHERE
  WR1: SIZEOF (
         QUERY (item <* SELF\representation.items |
           SIZEOF (['PRESENTATION_DEFINITION_SCHEMA.TEXT_LITERAL',
                    'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT',
                    'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT_CHARACTER',
                    'PRESENTATION_DEFINITION_SCHEMA.DEFINED_CHARACTER_GLYPH',
                    'PRESENTATION_DEFINITION_SCHEMA.COMPOSITE_TEXT',
                    'GEOMETRY_SCHEMA.AXIS2_PLACEMENT'] * TYPEOF (item)) = 0
         )) = 0;
  WR2: SIZEOF (
         QUERY (item <* SELF\representation.items |
           NOT (SIZEOF (['PRESENTATION_DEFINITION_SCHEMA.TEXT_LITERAL',
                         'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT',
                         'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT_CHARACTER',
                         'PRESENTATION_DEFINITION_SCHEMA.DEFINED_CHARACTER_GLYPH',
                         'PRESENTATION_DEFINITION_SCHEMA.COMPOSITE_TEXT'] *
                    TYPEOF (item)) = 0)
         )) >= 1;
  WR3: SIZEOF (
         QUERY (a2p <*
           QUERY (item <* SELF\representation.items |
             'GEOMETRY_SCHEMA.AXIS2_PLACEMENT' IN TYPEOF (item)) |
           NOT ((SIZEOF (
             QUERY (at <*
               QUERY (item <* SELF\representation.items |
                  'PRESENTATION_DEFINITION_SCHEMA.' +
                  'ANNOTATION_TEXT' IN TYPEOF (item)) |
               (at\mapped_item.mapping_target :=: a2p))) >= 1) OR
           (SIZEOF (
             QUERY (atc <*
               QUERY (item <* SELF\representation.items |
                 'PRESENTATION_DEFINITION_SCHEMA.' +
                 'ANNOTATION_TEXT_CHARACTER' IN TYPEOF (item)) |
```

```
            (atc\mapped_item.mapping_target :=: a2p))) >= 1)
        ))) = 0;
END_ENTITY;
(*
```

Attribute definitions:

SELF\representation.items: an **items** set redefined to be the set of strings and the placement.

Formal propositions:

WR1: Each item in the **text_string_representation** shall be an **annotation_text**, an **annotation_text_character**, a **text_literal**, a **composite_text**, or an **axis2_placement**.

WR2: The **text_string_representation** shall contain one or more **annotation_text**, **annotation_text_character**, **text_literal**, or **composite_text**.

WR3: Each **axis2_placement** shall be the **mapping_target** of an **annotation_text** or an **annotation_text_character** in the **text_string_representation**.

5.4.24 annotation_text_character

An **annotation_text_character** is a **mapped_item** which has a character as its **mapping_source**. That character is a glyph that exists within a conforming exchange.

EXPRESS specification:

```
*)
ENTITY annotation_text_character
  SUBTYPE OF (mapped_item);
  alignment : text_alignment;
WHERE
  WR1: 'PRESENTATION_RESOURCE_SCHEMA.CHARACTER_GLYPH_SYMBOL' IN
       TYPEOF (SELF\mapped_item.mapping_source.mapped_representation);
  WR2: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT' IN
       TYPEOF (SELF\mapped_item.mapping_target);
  WR3: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN
       TYPEOF (SELF);
END_ENTITY;
(*
```

Attribute definitions:

alignment: the specification of the point by which the character is located.

SELF\mapped_item.mapping_source: the character glyph that exists within a conforming exchange.

SELF\mapped_item.mapping_target: an **axis2_placement** which is the target of the mapping transformation.

Formal propositions:

WR1: The source of the **annotation_text_character** shall be a **character_glyph_symbol**.

WR2: The target of the **annotation_text_character** shall be an **axis2_placement**.

WR3: The **annotation_text_character** shall be a **geometric_representation_item**.

5.4.25 defined_character_glyph

A **defined_character_glyph** is a character glyph that has an implicit definition, either through a **pre_defined_character_glyph** or an **externally_defined_character_glyph**.

EXPRESS specification:

```
*)
ENTITY defined_character_glyph
  SUBTYPE OF(geometric_representation_item);
  definition : defined_glyph_select;
  placement  : axis2_placement;
END_ENTITY;
(*
```

Attribute definitions:

definition: an implicit description of a character glyph, either pre–defined or externally defined.

placement: a description of the placement and orientation of the character glyph.

5.4.26 externally_defined_character_glyph

An **externally_defined_character_glyph** is an **externally_defined_item** that makes an external reference to a character glyph.

EXPRESS specification:

```
*)
ENTITY externally_defined_character_glyph
  SUBTYPE OF (externally_defined_item);
END_ENTITY;
(*
```

5.4.27 pre_defined_character_glyph

A **pre_defined_character_glyph** allows a conforming exchange to define an application–specific character glyph. The actual character glyph shall be defined by an Application Protocol.

EXPRESS specification:

```
*)
ENTITY pre_defined_character_glyph
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

5.4.28 text_literal

A **text_literal** is a definition of a text string using a string literal.

EXPRESS specification:

```
*)
ENTITY text_literal
  SUBTYPE OF (geometric_representation_item);
  literal    : presentable_text;
  placement  : axis2_placement;
  alignment  : text_alignment;
  path       : text_path;
  font       : font_select;
END_ENTITY;
(*
```

Attribute definitions:

literal: the text literal to be presented.

placement: an **axis2_placement** which determines the position and orientation of the presented string. The y axis is taken as reference direction for **box_rotate_angle** and **box_slant_angle**.

NOTE – If both **box_rotation_angle** and **box_slant_angle** are zero the baseline of each character box is parallel to the x axis and the upright direction of each character box is parallel to the y axis. See figure 17.

alignment: the alignment of the text literal relative to its position.

path: the writing direction of the text literal.

font: the pre–defined or externally defined font which shall be used for presenting the **text_literal**.

5.4.29 text_literal_with_extent

A **text_literal_with_extent** is a **text_literal** whose extent is explicitly specified.

EXPRESS specification:

```
*)
ENTITY text_literal_with_extent
  SUBTYPE OF (text_literal);
  extent : planar_extent;
END_ENTITY;
(*
```

Attribute definitions:

extent: the extent in x and y direction of the **text_literal_symbol**.

5.4.30 text_literal_with_delineation

A **text_literal_with_delineation** is a **text_literal** that specifies the delineation of the text. The type of delineation and its effect on the appearance of the text is specified in an Application Protocol.

EXPRESS specification:

```
*)
ENTITY text_literal_with_delineation
  SUBTYPE OF (text_literal);
  delineation : text_delineation;
END_ENTITY;
(*
```

Attribute definitions:

delineation: the specification of the delineation to be applied to the text.

5.4.31 text_literal_with_blanking_box

An **text_literal_with_blanking_box** is a **text_literal** that contains a blanking box.

EXPRESS specification:

```
*)
ENTITY text_literal_with_blanking_box
  SUBTYPE OF (text_literal);
  blanking : planar_box;
END_ENTITY;
(*
```

Attribute definitions:

blanking: a **planar_box** that defines a rectangular area within which only the **text_literal** is presented.

5.4.32 text_literal_with_associated_curves

A **text_literal_with_associated_curves** is a **text_literal** that contains one ore more **curves**.

EXPRESS specification:

```
*)
ENTITY text_literal_with_associated_curves
  SUBTYPE OF (text_literal);
  associated_curves : SET[1:?] of curve;
END_ENTITY;
(*
```

Attribute definitions:

associated_curves: a set of **curves** associated with the **text_literal**.

NOTE – If the **curves** associated with a **text_literal_with_associated_curves** are to be presented, then they may be assigned style by **annotation_curve_occurrence**.

5.4.33 composite_text

A **composite_text** is a collection of **text_literal**, **annotation_text**, **annotation_text_character**, **defined_character_glyph**, or other **composite_text** that may be placed and styled as a collection.

EXPRESS specification:

```
*)
ENTITY composite_text
  SUBTYPE OF (geometric_representation_item);
  collected_text : SET[2:?] of text_or_character;
WHERE
  WR1: acyclic_composite_text( SELF, SELF.collected_text);
END_ENTITY;
(*
```

Attribute definitions:

collected_text: the set of **text_literal**, **annotation_text**, **annotation_text_character**, or other **composite_text** that may be placed and styled as a collection.

Formal propositions:

WR1: A composite text shall not participate in its own definition.

5.4.34 composite_text_with_extent

A **composite_text_with_extent** is a **composite_text** whose extent is explicitly specified.

EXPRESS specification:

```
*)
ENTITY composite_text_with_extent
  SUBTYPE OF (composite_text);
  extent : planar_extent;
END_ENTITY;
(*
```

Attribute definitions:

extent: the extent in x and y directions of the **composite_text_with_extent**.

5.4.35 composite_text_with_delineation

A **composite_text_with_delineation** is a **composite_text** that specifies the delineation of the text. The type of delineation and its effect on the appearance of the text is specified in an Application Protocol.

EXPRESS specification:

```
*)
ENTITY composite_text_with_delineation
  SUBTYPE OF (composite_text);
  delineation : text_delineation;
END_ENTITY;
```

```
(*
```

Attribute definitions:

delineation: the specification of the delineation to be applied to the text.

5.4.36 composite_text_with_blanking_box

An **composite_text_with_blanking_box** is a **composite_text** that contains a blanking box.

EXPRESS specification:

```
*)
ENTITY composite_text_with_blanking_box
  SUBTYPE OF (composite_text);
  blanking : planar_box;
END_ENTITY;
(*
```

Attribute definitions:

blanking: a **planar_box** that defines a rectangular area within which only the **composite_text** is presented.

5.4.37 composite_text_with_associated_curves

A **composite_text_with_associated_curves** is a **composite_text** that contains one ore more **curves**.

EXPRESS specification:

```
*)
ENTITY composite_text_with_associated_curves
  SUBTYPE OF (composite_text);
  associated_curves : SET[1:?] of curve;
END_ENTITY;
(*
```

Attribute definitions:

associated_curves: a set of **curves** associated with the **composite_text**.

NOTE – If the **curves** associated with a **composite_curve_with_associated_curves** are to be presented, then they may be assigned style by **annotation_curve_occurrence**.

5.5 Presentation definition schema entity definitions: annotation occurrences

5.5.1 annotation_occurrence

An **annotation_occurrence** defines occurrences of annotation by combining two–dimensional geometry or annotation elements with style for presentation purposes. See clause 6 for more detail about assigning style. **Annotation_occurrences** shall be used only in **representations**

which are defined for annotation purposes, i.e., **area_dependent_annotation_representation**, **view_dependent_annotation_representation**, **curve_style_curve_pattern**, **fill_area_style_tile_curve_with_style**, or **fill_area_style_tile_coloured_region**.

EXPRESS specification:

```
*)
ENTITY annotation_occurrence
  SUPERTYPE OF (ONEOF(annotation_point_occurrence,
                      annotation_curve_occurrence,
                      annotation_fill_area_occurrence,
                      annotation_text_occurrence,
                      annotation_symbol_occurrence))
  SUBTYPE OF (styled_item);
WHERE
  WR1: 'GEOMETRY_SCHEMA.GEOMETRIC_REPRESENTATION_ITEM' IN
         TYPEOF (SELF);
END_ENTITY;
(*
```

Formal propositions:

WR1: A **annotation_occurrence** shall be a **geometric_representation_item**.

5.5.2 annotation_point_occurrence

An **annotation_point_occurrence** is a **point** with a style assignment.

EXPRESS specification:

```
*)
ENTITY annotation_point_occurrence
  SUBTYPE OF (annotation_occurrence);
WHERE
  WR1: 'GEOMETRY_SCHEMA.POINT' IN TYPEOF (SELF\styled_item.item);
END_ENTITY;
(*
```

Formal propositions:

WR1: The styled item shall be a **point**.

5.5.3 annotation_curve_occurrence

An **annotation_curve_occurrence** is a **curve** with a style assignment.

EXPRESS specification:

```
*)
ENTITY annotation_curve_occurrence
  SUBTYPE OF (annotation_occurrence);
WHERE
  WR1: 'GEOMETRY_SCHEMA.CURVE' IN TYPEOF (SELF\styled_item.item);
END_ENTITY;
(*
```

Formal propositions:

WR1: The styled item shall be a **curve**.

5.5.4 annotation_fill_area_occurrence

An **annotation_fill_area_occurrence** is the assignment of a style to an **annotation_fill_area**; it includes the specification of the point to be used as the starting point of the **fill_area**.

EXPRESS specification:

```
*)
ENTITY annotation_fill_area_occurrence
  SUBTYPE OF (annotation_occurrence);
  fill_style_target : point;
WHERE
  WR1: 'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_FILL_AREA' IN
       TYPEOF (SELF.item);
END_ENTITY;
(*
```

Attribute definitions:

fill_style_target: the point that specifies the starting location for the fill_area_style assigned to the annotation_fill_area_occurrence.

Formal propositions:

WR1: The styled item shall be an **annotation_fill_area**.

5.5.5 annotation_text_occurrence

An **annotation_text_occurrence** is a **text_literal**, **annotation_text**, **annotation_text_character**, **defined_character_glyph**, or **composite_text** with a style assignment.

EXPRESS specification:

```
*)
ENTITY annotation_text_occurrence
  SUBTYPE OF (annotation_occurrence);
WHERE
  WR1: SIZEOF (
         ['PRESENTATION_DEFINITION_SCHEMA.TEXT_LITERAL',
          'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT',
          'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT_CHARACTER',
          'PRESENTATION_DEFINITION_SCHEMA.DEFINED_CHARACTER_GLYPH',
          'PRESENTATION_DEFINITION_SCHEMA.COMPOSITE_TEXT'] *
         TYPEOF (SELF\styled_item.item)) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The item to which style is assigned shall be the type of at least one of **text_literal**, **annotation_text**, **annotation_text_character**, **defined_character_glyph**, or **composite_text**.

5.5.6 annotation_symbol_occurrence

An **annotation_symbol_occurrence** is an **annotation_symbol** with a style assignment or a **defined_symbol** with a style assignment.

EXPRESS specification:

```
*)
ENTITY annotation_symbol_occurrence
  SUBTYPE OF (annotation_occurrence);
WHERE
  WR1: SIZEOF(
         ['PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_SYMBOL',
          'PRESENTATION_DEFINITION_SCHEMA.DEFINED_SYMBOL'] *
         TYPEOF(SELF\styled_item.item)) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The styled item shall be an **annotation_symbol** or a **defined_symbol** .

5.5.7 annotation_table_occurrence

An **annotation_table_occurrence** is an **annotation_table** with a style assignment or a **defined_table** with a style assignment.

EXPRESS specification:

```
*)
ENTITY annotation_table_occurrence
  SUBTYPE OF (annotation_symbol_occurrence);
WHERE
  WR1: SIZEOF (
         ['PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TABLE',
          'PRESENTATION_DEFINITION_SCHEMA.DEFINED_TABLE'] *
         TYPEOF (SELF\styled_item.item)) > 0;
END_ENTITY;
(*
```

Formal propositions:

WR1: The styled item shall be an **annotation_table** or a **defined_table**.

5.5.8 annotation_occurrence_relationship

An **annotation_occurrence_relationship** is the association of two **annotation_occurrences**.

EXPRESS specification:

```
*)
ENTITY annotation_occurrence_relationship;
  name                           : label;
  description                    : text;
  relating_annotation_occurrence : annotation_occurrence;
```

```
  related_annotation_occurrence  : annotation_occurrence;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **annotation_occurrence_relationship** is referred to.

description: text that relates the nature of the **annotation_occurrence_relationship**.

relating_annotation_occurrence: the first of two **annotation_occurrences** which are related.

related_annotation_occurrence: the second of two **annotation_occurrences** which are related.

NOTE – There is no significance to the ordering of the two related **annotation_occurrences**. The use of the different attribute names serves only to distinguish the names.

5.5.9 table_text_relationship

A **table_text_relationship** establishes a relationship between a field in an **annotation_table_occurrence** and an **annotation_text_occurrence**. It is used to include text in a field of a table.

EXPRESS specification:

```
*)
ENTITY table_text_relationship
  SUBTYPE OF (annotation_occurrence_relationship);
  field : table_record_field_representation;
WHERE
  WR1: 'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TABLE_OCCURRENCE'
       IN TYPEOF (SELF\annotation_occurrence_relationship.
                  relating_annotation_occurrence);
  WR2: 'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TABLE'
       IN TYPEOF (SELF\annotation_occurrence_relationship.
                  relating_annotation_occurrence\styled_item.item);
  WR3: 'PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT_OCCURRENCE'
       IN TYPEOF (SELF\annotation_occurrence_relationship.
                  related_annotation_occurrence);
  WR4: field_in_table (SELF.field,
                       SELF\annotation_occurrence_relationship.
                       relating_annotation_occurrence);
END_ENTITY;
(*
```

Attribute definitions:

field: the particular field in the table in which the text is positioned.

SELF\annotation_occurrence_relationship.relating_annotation_occurrence: the **annotation_table_occurrence** in which text is positioned.

SELF\annotation_occurrence_relationship.related_annotation_occurrence: the **annotation_text_occurrence** positioned in the table.

Formal propositions:

WR1: The **relating_annotation_occurrence** shall be an **annotation_table_occurrence**.

WR2: The **relating_annotation_occurrence** shall present an **annotation_table**.

WR3: The **related_annotation_occurrence** shall be an **annotation_text_occurrence**.

WR4: The field shall be a field in the table in which the text is positioned.

5.6 Presentation definition schema function definitions

5.6.1 acyclic_composite_text

The **acyclic_composite_text** function examines a **composite_text** instance to see if it participates in its own definition. It returns TRUE if the **composite_text** instance is acyclic (does not participate in its own definition) and FALSE if it is cyclic (does participate in its own definition).

The function first checks to see if it is among its own **collected_text** set, and returns FALSE if it is. Next, it creates a local set of all instances of **composite_text** in its own **collected_text** set. It then adds to this set all of the instances of **composite_text** that are referenced by the **text_string_representation** used by the **representation_map** used by any **annotation_text** in its own **collected_text** set. This local set is added to the set of instances already examined. If the set of instances already examined does not increase in size, then all possibilities have been examined and the function returns TRUE. Otherwise, the function is then called recursively to check further.

EXPRESS specification:

```
*)
FUNCTION acyclic_composite_text(start_composite : composite_text;
                                child_text : SET [1:?] OF
                                text_or_character) : LOGICAL;

  LOCAL
   i : INTEGER;
   local_composite_text : SET [0:?] OF composite_text;
   local_annotation_text : SET [0:?] OF annotation_text;
   local_children : SET [0:?] OF text_or_character;
  END_LOCAL;

  local_composite_text := QUERY (child <* child_text |
                          ('PRESENTATION_DEFINITION_SCHEMA.COMPOSITE_TEXT'
                           IN TYPEOF (child)));

  IF (SIZEOF (local_composite_text) > 0) THEN
    REPEAT i := 1 TO HIINDEX (local_composite_text);
      IF (start_composite :=: local_composite_text[i]) THEN
        RETURN (FALSE);
```

```
    END_IF;
  END_REPEAT;
END_IF;

local_children := child_text;

IF (SIZEOF (local_composite_text)) > 0 THEN
  REPEAT i := 1 TO HIINDEX (local_composite_text);
    local_children := local_children +
                      local_composite_text[i].collected_text;
  END_REPEAT;
END_IF;


local_annotation_text := QUERY (child <* child_text |
                         ('PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT'
                          IN TYPEOF (child)));

IF (SIZEOF (local_annotation_text) > 0) THEN
  REPEAT i := 1 TO HIINDEX (local_annotation_text);
    local_children := local_children +
    QUERY (item <* local_annotation_text[i]\mapped_item.
                   mapping_source.mapped_representation.items |
      SIZEOF(['PRESENTATION_DEFINITION_SCHEMA.ANNOTATION_TEXT',
              'PRESENTATION_DEFINITION_SCHEMA.COMPOSITE_TEXT'] *
             TYPEOF(item)) > 0);
  END_REPEAT;
END_IF;

IF (local_children :<>: child_text) THEN
  RETURN (acyclic_composite_text (start_composite, local_children));
ELSE
  RETURN (TRUE);
END_IF;

END_FUNCTION;
(*
```

Argument definitions:

start_composite: the **composite_text** which is tested. This is input to the function.

child_text: the **text_or_character**s referenced by **start_composite**. This is input to the function. On initial input this set contains as its only element the **collected_text** of the **start_composite**.

5.6.2 acyclic_symbol_representation_relationship

The **acyclic_symbol_representation_relationship** function determines if a **symbol_representation** is used in the tree of **symbol_representation_relationship**s that defines this **symbol_representation**. It returns TRUE if it is not acyclic, and FALSE if there is.

EXPRESS specification:

```
*)
FUNCTION acyclic_symbol_representation_relationship
  (relation : symbol_representation_relationship;
   children : SET OF symbol_representation ) : BOOLEAN;
  LOCAL
    x : SET OF symbol_representation_relationship;
    i : INTEGER;
    local_children : SET OF symbol_representation;
  END_LOCAL;

  REPEAT i:=1 TO HIINDEX(children);
    IF relation\representation_relationship.rep_1 :=: children[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;

  x := USEDIN ( relation\representation_relationship.rep_1,
                'REPRESENTATION_SCHEMA.'+
                'REPRESENTATION_RELATIONSHIP.'+ 'REP_2');
  local_children := children + relation\representation_relationship.rep_1;

  IF SIZEOF (x) > 0 THEN
    REPEAT i:=1 TO HIINDEX (x);
      IF NOT acyclic_symbol_representation_relationship(x[i] ,
                                                        local_children) THEN

        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;

  RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

relation: the **symbol_representation_relationship** which is tested. This is input to the function.

children: the **symbol_representations** referenced by **relation**. This is input to the function. On initial input this set contains as only element the **rep_2** of the **relation**.

5.6.3 field_in_table

The **field_in_table** function is a function that examines a **table_record_field_representation** and an **annotation_table_occurrence**, and returns the value TRUE if the field is in the table, and FALSE if the field is not in the table.

The function first finds the **table_representation** that the **annotation_table_occurrence** maps. It then finds all the **table_record_representations** which are either related to the

table_representation_entity by a **symbol_representation_relationship** or are included in the **table_representation** through a **mapped_item**. Finally, it returns FALSE if there are no **table_record_representation** entities which relate to the field by a **symbol_representation_relationship** or include the field through a **mapped_item**, and returns TRUE otherwise.

EXPRESS specification:

```
*)
FUNCTION field_in_table (field : table_record_field_representation;
                         table : annotation_table_occurrence): BOOLEAN;
  LOCAL
    i : INTEGER;
    table_rep : table_representation;
    symbol_rep_rel_set : SET [1:?] OF symbol_representation_relationship;
    mapped_item_set : SET [1:?] OF mapped_item;
    table_record_rep_set : SET [1:?] OF table_record_representation := [];
  END_LOCAL;

  table_rep := table\styled_item.item\mapped_item.mapping_source.
    mapped_representation;
  mapped_item_set := QUERY(item <* table_rep.items |
                       ('REPRESENTATION_SCHEMA.MAPPED_ITEM' IN
                        TYPEOF(item))
                                 AND
                       ('PRESENTATION_DEFINITIONS_SCHEMA.'+
                        'TABLE_RECORD_REPRESENTATION' IN
                         TYPEOF(item\mapped_item.mapping_source.
                                  mapped_representation ))
                     );

  REPEAT i := 1 TO HIINDEX(mapped_item_set);
    table_record_rep_set := table_record_rep_set +
          mapped_item_set[i].mapping_source.mapped_representation;
  END_REPEAT;

  symbol_rep_rel_set := USEDIN(table_rep,
                               'REPRESENTATION_SCHEMA.'+
                               'REPRESENTATION_RELATIONSHIP.REP_1');

  REPEAT i := 1 TO HIINDEX(symbol_rep_rel_set);
    table_record_rep_set := table_record_rep_set +
              symbol_rep_rel_set[i]\representation_relationship.rep_2;
  END_REPEAT;

  IF SIZEOF(QUERY( table_record_rep <* table_record_rep_set |
              (SIZEOF(QUERY( symbol_rep_rel <* USEDIN(table_record_rep,
                          'PRESENTATION_DEFINITION_SCHEMA.'+
                          'SYMBOL_REPRESENTATION_RELATIONSHIP') |
                        symbol_rep_rel\representation_relationship.rep_2 :=: field
                        )) > 0)
                        OR
              (SIZEOF(QUERY(item <* table_record_rep.items |
                         ('REPRESENTATION_SCHEMA.MAPPED_ITEM' IN
```

```
                        TYPEOF(item))
                                AND
                        (field :=: item\mapped_item.mapping_source.
                                   mapped_representation )
                        )) > 0)
            )) = 0 THEN
    RETURN(FALSE);
  END_IF;

  RETURN(TRUE);

END_FUNCTION;
(*
```

Argument definitions:

field: the table record field which shall be in the **table**. This is input to the function.

table: the annotation table which shall contain the **field**. This is input to the function.

EXPRESS specification:

```
*)
END_SCHEMA; -- presentation_definition_schema
(*
```

6 Presentation appearance

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

EXPRESS specification:

```
*)
SCHEMA presentation_appearance_schema;

REFERENCE FROM external_reference_schema
    (externally_defined_item,
     pre_defined_item);

REFERENCE FROM geometry_schema
    (axis2_placement,
     cartesian_point,
     curve,
     geometric_representation_item,
     point,
     vector);

REFERENCE FROM measure_schema
    (descriptive_measure,
     length_measure,
     measure_with_unit,
     plane_angle_measure,
```

```
     positive_length_measure,
     ratio_measure,
     positive_ratio_measure);

REFERENCE FROM presentation_organization_schema
     (area_dependent_annotation_representation,
      presentation_area,
      presentation_layer_assignment,
      presentation_layer_usage,
      presentation_representation,
      presentation_set,
      presentation_view,
      product_data_representation_view,
      view_dependent_annotation_representation);

REFERENCE FROM presentation_definition_schema
    (annotation_curve_occurrence,
     annotation_fill_area,
     annotation_symbol_occurrence,
     annotation_text_with_delineation,
     symbol_representation_with_blanking_box);

REFERENCE FROM presentation_resource_schema
    (character_glyph_symbol_outline,
     character_glyph_symbol_stroke,
     colour);

REFERENCE FROM representation_schema
    (mapped_item,
     representation,
     representation_item,
     representation_map,
     using_representations);

REFERENCE FROM support_resource_schema
    (label);

(*
    NOTES
```

1 The schemas referenced above can be found in the following parts of ISO 10303:

external_reference_schema	ISO 10303-41
geometry_schema	ISO 10303-42
management_resources_schema	ISO 10303-41
measure_schema	ISO 10303-41
presentation_organization_schema	Clause 4 of this part of ISO 10303

presentation_definition_schema	Clause 5 of this part of ISO 10303
presentation_resource_schema	Clause 7 of this part of ISO 10303
representation_schema	ISO 10303-43
support_resource_schema	ISO 10303-41

2 – The EXPRESS–G diagrams for this schema may be found in annex E of this part of ISO 10303.

6.1 Introduction

The subject of the **presentation_appearance_schema** is the specification of the intended graphical appearance of a presented picture. The **presentation_appearance_schema** also defines the mechanism which allows the appropriate association of these appearance attributes with **annotation_occurrences** in the context of a **presentation_representation**. Thus, the context of the **presentation_representation** can be used to determine the appearance of different kinds of elements in the picture.

6.2 Fundamental concepts and assumptions

6.2.1 Assignment of presentation style

The **presentation_appearance_schema** allows the association of appearance attributes with selected annotation primitives and with product shape elements. In this part of ISO 10303, presentation style can be assigned to any **representation_item**. A style assignment is made by instantiating a **styled_item** which refers to a **representation_item** together with its **presentation_style_assignment**. A **presentation_style_assignment** itself is a collection of different presentation styles such as point style, curve style, or text style. Styling an unstyled **representation_item** produces a new **representation_item** which has presentation style assigned. The **presentation_style_assignment** of a **styled_item** affects the appearance of the referenced **representation_item** as well as the appearance of all **representation_items** referenced directly or indirectly by that item. Only those **representation_items** are affected which are not already styled. This means styling a styled **representation_item** has no effect, styling a partially styled **representation_item** affects only the appearance of the unstyled parts, and styling an unstyled **representation_item** affects the appearance of the whole item. Only styled **representation_items** may be presented. Whether they are actually presented depends on several other facts, like layer invisibility and **invisibility**, hidden line and surface removal, as well as clipping. This part of ISO 10303 does not make any statement about the effect if style conflicts occurs. A style conflict occurs, for example, when a **representation_item** is used by several **styled_items**.

A **presentation_style_assignment** is used to assign style to a **representation_item** independently from any presentation context. A subtype of **presentation_style_assignment**, the **presentation_style_by_context**, allows the assignment of style for a specific presentation context. A presentation context can be any **presentation_set**, **representation**, or **representa-**

tion_item. In the last case, a style can be assigned to an item as a whole, and a different style can be assigned to a part of that item.

A style assignment can be over–riden by an **over_riding_styled_item**. A subtype of **over_riding_styled_item**, the **context_dependent_over_riding_styled_item**, allows the over–riding of a style for a specific presentation context.

6.2.2 Types of presentation styles

For each annotation primitive defined in the **presentation_definition_schema**, a special group of styles exist. These groups are point styles, curve styles, fill area styles, surface styles, text styles, and symbol styles. This part of ISO 10303 does not restrict style assignment to product shape elements. For example, it allows the assignment of surface style to a point. However, only appropriate styles affect the appearance of the product shape element; i.e., surface styles only affect surfaces, fill area styles and curve styles only affect curves, and point styles only affect points, curves, and surfaces. For high–level product shape elements such as solid models, the using Application Protocol has to specify which styles affect the appearance of these elements. Nevertheless, the style assignment for annotation primitives is more restrictive. Details on these restrictions can be found in clause 5 subtypes of **annotation_occurrence**.

Presentation styles can be specified using the resources of this schema, can be externally defined, or can be pre–defined by Application Protocols. The presentation styles defined in this schema include the following.

Point styles provide the resources to specify the visual appearance of points. They allow the specification of the marker symbol, marker size, and colour to be used for presenting points. Point styles can be specified by the entity **point_style**.

Curve styles provide the resources to specify the visual appearance of curves. They allow the specification of curve fonts, curve width, appearance of curve ends and corners, colour or patterns for filling visible curve segments. A curve font specifies whether a curve shall be drawn using solid, dashed, or dotted lines. The **presentation_appearance_schema** allows the specification of arbitrary patterns for curve fonts and the usage of externally defined or pre–defined curve fonts. The entities used for the definition of curve fonts are **externally_defined_curve_font**, **pre_defined_curve_font**, **curve_style_font**, **curve_style_font_pattern**, and **curve_style_font_and_scaling**. The curve width can be specified as measure or pre–defined size. Curve ends and corners may be drawn squared or rounded, and extension or shortage can be specified for curve ends. The entities supporting this structure are **curve_style_with_ends_and_corners** and **curve_style_with_extension**. **curve_style_wide** allows the specification of a fill area style which is used for filling visible curve segments. This has been included to meet draughting requirements for the presentation of curves. The **curve_style_rendering** controls the rendering of curves on a surface and is described together with surface styles.

Fill area styles provide the resources to specify the visual appearance of annotation fill areas. They allow the specification of a solid colour, hatches, or tiling patterns. The **fill_area_style_colour** can be used to specify a solid colour that is used for presenting a fill area. The hatching patterns can be composed of sets of parallel lines with arbitrary angle. Alternatively, hatching patterns may be externally defined or pre–defined. The entities for hatches are **fill_area_style_hatching**, **pre_defined_hatch_style**, **externally_defined_hatch_style**, and

one_direction_repeat_factor. The tiling patterns can be made of curves, coloured regions, or symbols. Alternatively, tiling pattern may be externally defined or pre–defined. Tiling patterns are repeated in two directions which are specified by arbitrary vectors. The entities for the definition of tiling patterns are **pre_defined_tile_style**, **externally_defined_tile_style**, **fill_area_style_tiles**, **fill_area_style_tile_curve_with_style**, **fill_area_style_tile_coloured_region**, **fill_area_style_tile_symbol_with_style**, **pre_defined_tile**, **externally_defined_tile**, and **two_direction_repeat_factor**.

Surface styles provide the resources to specify the visual appearance of surfaces. Separate surface styles may be applied to each side of a surface, or the same surface styles can be used for both sides. The surface style usage specifies a surface side style for one or both sides of a surface. Entities for the assignment of style to sides of a surface are **surface_style_usage** and **surface_side_style**. A surface side style may be any combination of fill area style, boundary style, silhouette style, segmentation curve style, control grid style, parameter line style, or rendering style.

A fill area style specifies a solid colour, hatches, or a tiling pattern for filling the side of a surface.

A boundary style specifies a curve style or a rendering method and properties for presenting the boundary curves of a surface. If no boundary style is specified, the boundary curves shall not be presented.

A silhouette style specifies a curve style or a rendering method and properties for presenting the silhouette curves of a surface. If no silhouette style is specified, silhouette curves shall not be presented.

A segmentation curve style specifies a curve style or a rendering method and properties for presenting the segmentation curves of a surface. This style affects only surfaces which are divided into segments such as B–spline surfaces. If no segmentation curve style is specified, segmentation curves shall not be presented.

A control grid style specifies a curve style or a rendering method and properties for presenting the mesh of control points which are used for the definition of a surface. This style affects only surfaces which are defined over a mesh of control points, such as B–spline surfaces. If no control grid style is specified, the control grid shall not be presented.

A parameter line style specifies a curve style or a rendering method and properties for presenting iso–parameter lines of a surface. The number of parameter lines in each parameter direction has to be specified for this style. If no parameter line style is specified, the parameter lines shall not be presented.

The entities defining these surface styles are **surface_fill_area**, **surface_style_boundary**, **surface_style_silhouette**, **surface_style_segmentation_curve**, **surface_style_control_grid**, and **surface_style_parameter_line**.

The entities **curve_style_rendering** and **surface_rendering_properties** specify the rendering method and properties for curves on a surface. A rendering style specifies the method which shall be used for rendering the surface. If this style is specified, surface rendering using reflectance calculations is performed for presenting the surface. For this style, a colour shall be specified for the surface. Additional rendering properties including transparency and ambient, diffuse, and specular reflectance coefficients can optionally be specified. The entities for rendering styles are **surface_style_rendering**, **surface_style_rendering_with_properties**, **sur-**

face_style_transparent, **surface_style_reflectance_ambient**, **surface_style_reflectance_ambient_diffuse**, and **surface_style_reflectance_ambient_diffuse_specular**. Besides those styles listed above, surface styles may also be pre–defined.

Text styles provide the resources to specify the visual appearance of annotation text. Text justification, box characteristics, character spacing, and mirror axis affect the appearance of the whole text, while character glyph style affects the appearance of the individual characters or symbols which make up the text. Text justification specifies how the text is aligned. The supported types of alignment are specified by Application Protocols. The box characteristics specify the geometry of the character box. The character box is a parallelogram for which the width, height, slant angle, and rotation angle may be specified. Text spacing defines the spacing which shall be included between adjacent character boxes of a string, in addition to the spacing included in the font definition. Text mirroring specifies a mirror axis which shall be used for mirroring the text. Mirroring shall be performed after all remaining text styles have been applied. The entities supporting the definition of these text styles are **text_style**, **text_style_with_spacing**, **text_style_with_mirror**, **text_style_with_justification**, and **text_style_with_box_characteristics**. A character glyph style may be a outline style, stroke style, or only a text colour. An outline style is a curve style that is used for presenting the outlines which make up a character glyph. Optionally, a fill area style for filling the outlines may be specified. An outline style affects only character glyphs which are defined by outlines. A stroke style is a curve style that is used for presenting the strokes which make up a character glyph. A stroke style affects only character glyphs which are defined by strokes. For pre–defined and externally defined fonts, only a colour shall be specified. The entities supporting the definition of these character glyph styles are **character_glyph_style_stroke**, **character_glyph_style_outline**, **character_glyph_style_outline_with_characteristics**, and **text_style_for_defined_font**.

Symbol styles provide the resources to specify the visual appearance of annotation symbols. A symbol style is an arbitrary collection of point styles, curve styles, fill area styles, surface styles, and text styles. The entities supporting the definition of these symbol styles are **symbol_style**, **symbol_element_style**, and **symbol_colour**.

6.2.3 Approximation tolerances

Approximation tolerances specify the allowable tolerances between the shape of the presented elements and their mathematically exact position and shape. Approximation tolerances apply to curves and surfaces and can be specified in parameter space, product shape space, or presentation area space. The space in which the tolerances are specified depends on the approximation method used. For the chordal deviation and chordal length approximation methods, the tolerance can be specified in product–shape or presentation–area space. For the parameter approximation method, the tolerance shall be specified in parameter space.

6.2.4 Occlusion and invisibility

The **presentation_appearance_schema** provides resources to specify the intended appearance of items which hide each other because they overlap in two–dimensional space. Those items include the elements of the presentation hierarchy which are defined in clause 4 in this part of ISO 10303, **annotation_fill_area**s, annotation texts with delineations, character glyphs, and

symbols. For this purpose, an **occlusion_precedence** can be specified for any two of these items.

In addition, the **presentation_appearance_schema** provides resources to specify the invisibility of **styled_item**s or layers.

6.3 Presentation appearance schema type definitions

6.3.1 style_context_select

The **style_context_select** selects between the entities which can be a context for the assignment or over–riding of presentation style.

EXPRESS specification:

```
*)
TYPE style_context_select = SELECT
  (representation,
   representation_item,
   presentation_set);
END_TYPE;
(*
```

6.3.2 presentation_style_select

The **presentation_style_select** is used by a **presentation_style_assignment** to associate style with a **representation_item**. A different style is provided for each kind of **representation_item** to be styled.

EXPRESS specification:

```
*)
TYPE presentation_style_select = SELECT
  (pre_defined_presentation_style,
   point_style,
   curve_style,
   surface_style_usage,
   symbol_style,
   fill_area_style,
   text_style,
   approximation_tolerance,
   externally_defined_style,
   null_style);
END_TYPE;
(*
```

6.3.3 null_style

The **null_style** specifies that no specific style is assigned directly to an item that is to be presented. The style or styles to be used in presenting the item are specified within the definition of the item. If no styles are specified within the definition, then the item shall not be presented.

EXAMPLE 9 – A symbol is defined using two **annotation_curve_occurrence**s, which are styled such that one is red and the other is blue. If an instance of the symbol is assigned **null_style**, it will be presented using the colours specified in the definition.

EXPRESS specification:

```
*)
TYPE null_style = ENUMERATION OF
  (null);
END_TYPE;
(*
```

Enumerated item definitions:

null: the **representation_item** to which the style is applied shall be presented using the style or styles contained in its definition, if any.

6.3.4 marker_select

The **marker_select** is a selection of a specific marker or a pre–defined marker used for the presentation of points.

EXPRESS specification:

```
*)
TYPE marker_select = SELECT
  (marker_type,
   pre_defined_marker);
END_TYPE;
(*
```

6.3.5 marker_type

The **marker_type** specifies the form for the presentation of points.

EXPRESS specification:

```
*)
TYPE marker_type = ENUMERATION OF
  (dot,
   x,
   plus,
   asterisk,
   ring,
   square,
   triangle);
END_TYPE;
(*
```

Enumerated item definitions:

dot: points are presented as dots (·).

x: points are presented as diagonal crosses (×).

plus: points are presented as plus signs (+).

asterisk: points are presented as asterisks (∗).

ring: points are presented as circles (○).

square: points are presented as squares (□).

triangle: points are presented as triangles (△).

6.3.6 size_select

The **size_select** is used to specify the size of marker symbols or the width of curves.

EXPRESS specification:

```
*)
TYPE size_select  = SELECT
  (positive_length_measure,
   measure_with_unit,
   descriptive_measure,
   pre_defined_size);
END_TYPE;
(*
```

6.3.7 curve_font_or_scaled_curve_font_select

The **curve_font_or_scaled_curve_font_select** is a selection of a **curve_style_font_select** or a **curve_style_font_and_scaling**. It is used to specify the font for presenting a curve.

EXPRESS specification:

```
*)
TYPE curve_font_or_scaled_curve_font_select = SELECT
  (curve_style_font_select,
   curve_style_font_and_scaling);
END_TYPE;
(*
```

6.3.8 curve_style_font_select

The **curve_style_font_select** is a selection of a **curve_style_font**, a **pre_defined_curve_font**, or an **externally_defined_curve_font**. It is used to specify an unscaled font for presenting a curve.

EXPRESS specification:

```
*)
TYPE curve_style_font_select = SELECT
  (curve_style_font,
   pre_defined_curve_font,
   externally_defined_curve_font);
END_TYPE;
(*
```

6.3.9 squared_or_rounded

The **squared_or_rounded** type specifies the appearance of curves at their corners or ends.

NOTE – See figure 16

EXPRESS specification:

```
*)
TYPE squared_or_rounded = ENUMERATION OF
  (squared,
   rounded);
END_TYPE;
(*
```

Enumerated item definitions:

squared: the curve is squared off at its end points and corners.

rounded: a filled semi-circular arc with diameter equal to the curve width is drawn around the curve's end points and corners.

6.3.10 fill_style_select

The **fill_style_select** is a selection between different fill styles.

EXPRESS specification:

```
*)
TYPE fill_style_select = SELECT
  (fill_area_style_colour,
   pre_defined_tile_style,
   externally_defined_tile_style,
   fill_area_style_tiles,
   pre_defined_hatch_style,
   externally_defined_hatch_style,
   fill_area_style_hatching);
END_TYPE;
(*
```

6.3.11 fill_area_style_tile_shape_select

The **fill_area_style_tile_shape_select** is used for the definition of **fill_area_style_tiles**. It selects between different shapes and sources of tiles.

EXPRESS specification:

```
*)
TYPE fill_area_style_tile_shape_select = SELECT
  (fill_area_style_tile_curve_with_style,
   fill_area_style_tile_coloured_region,
   fill_area_style_tile_symbol_with_style,
   pre_defined_tile,
   externally_defined_tile);
END_TYPE;
```

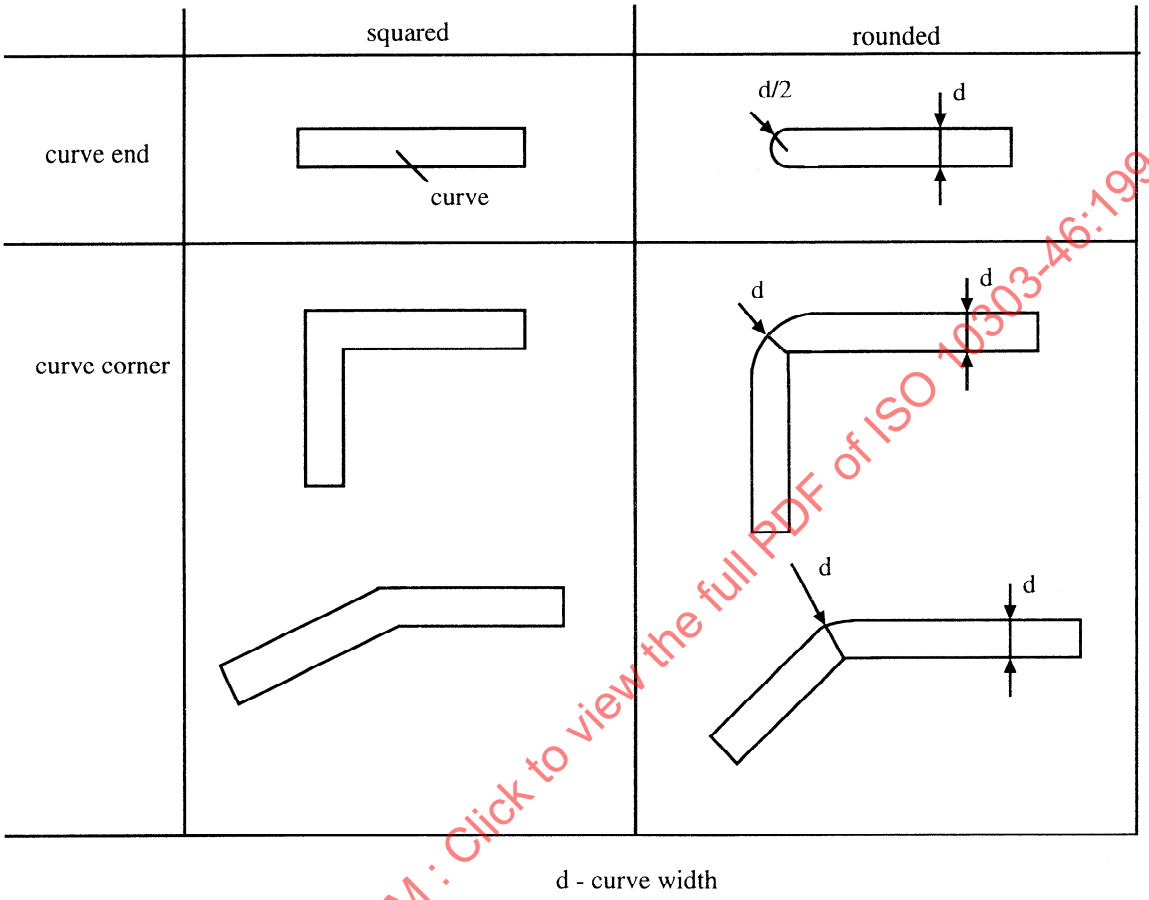


Figure 16 – Squared or rounded

(*

6.3.12 curve_or_annotation_curve_occurrence

The **curve_or_annotation_curve_occurrence** provides a **curve** or an **annotation_curve_occurrence** for defining the boundary of a **fill_area_style_tile_coloured_region**.

EXPRESS specification:

```
*)
TYPE curve_or_annotation_curve_occurrence = SELECT
  (curve,
   annotation_curve_occurrence);
END_TYPE;
(*
```

6.3.13 surface_side

The **surface_side** is used by **surface_style_usage** to specify the sides of a surface to which a surface style is applied.

EXPRESS specification:

```
*)
TYPE surface_side = ENUMERATION OF
  (positive,
   negative,
   both);
END_TYPE;
(*
```

Enumerated item definitions:

positive: the side of a surface which is in the same direction as the surface normal.

negative: the side of a surface which is in the opposite direction than the surface normal.

both: both sides.

6.3.14 surface_side_style_select

The **surface_side_style_select** is a selection of a **surface_side_style** or a **pre_defined_surface_side_style**.

EXPRESS specification:

```
*)
TYPE surface_side_style_select = SELECT
  (surface_side_style,
   pre_defined_surface_side_style);
END_TYPE;
(*
```

6.3.15 surface_style_element_select

The **surface_style_element_select** is a selection of the different surface styles to use in the presentation of the side of a surface.

EXPRESS specification:

```
*)
TYPE surface_style_element_select = SELECT
  (surface_style_fill_area,
   surface_style_boundary,
   surface_style_silhouette,
   surface_style_segmentation_curve,
   surface_style_control_grid,
   surface_style_parameter_line,
   surface_style_rendering);
END_TYPE;
(*
```

6.3.16 curve_or_render

The **curve_or_render** select type is used in the definition of several surface styles to control the appearance of curves on a surface.

EXPRESS specification:

```
*)
TYPE curve_or_render = SELECT
  (curve_style,
   curve_style_rendering);
END_TYPE;
(*
```

6.3.17 shading_curve_method

The **shading_curve_method** specifies the method which shall be used for the shading of curves. Shading of curves (approximated by polylines) is the colouring of the line segments according to colour values at certain points of the curve.

EXPRESS specification:

```
*)
TYPE shading_curve_method = ENUMERATION OF
  (constant_colour,
   linear_colour);
END_TYPE;
(*
```

Enumerated item definitions:

constant_colour: if the curve has a colour association table and is approximated by i polyline segments, the colour state at the segment boundaries shall be interpolated in this table according to the state variable value at each boundary. The i–th polyline segment shall then be shaded in constant colour according to the colour state at the i–th segment start.

linear_colour: if the curve has a colour association table and is approximated by i polyline segments, the colour state at the segment boundaries shall be interpolated in this table according to the state variable value at each boundary. The i–th polyline segment shall then be shaded in linearly interpolated colour according to the colour states at both segment boundaries.

6.3.18 direction_count_select

The **direction_count_select** is a selection between a **u_direction_count** and a **v_direction_count** for use in presenting parameter lines of a surface.

EXPRESS specification:

```
*)
TYPE direction_count_select = SELECT
  (u_direction_count,
   v_direction_count);
END_TYPE;
(*
```

6.3.19 u_direction_count

The **u_direction_count** is a positive integer indicating the number of parameter curves in the u direction of a parametric surface.

EXPRESS specification:

```
*)
TYPE u_direction_count = INTEGER;
WHERE
  WR1: SELF > 1;
END_TYPE;
(*
```

Formal propositions:

WR1: The u direction count shall be greater than 1.

6.3.20 v_direction_count

The **v_direction_count** is a positive integer indicating the number of parameter curves in the v direction of a parametric surface.

EXPRESS specification:

```
*)
TYPE v_direction_count = INTEGER;
WHERE
  WR1: SELF > 1;
END_TYPE;
(*
```

Formal propositions:

WR1: The v direction count shall be greater than 1.

6.3.21 shading_surface_method

The **shading_surface_method** specifies the method which shall be used for the shading of surfaces.

NOTES

1 – The descriptions of the different types of surface shading show that in some cases the method involves interpolating results from the lighting, and in other case it involves performing the reflectance calculation after interpolation. For this reason, the shading method may be thought of as selecting a location in a display system's graphics pipeline where interpolation takes place.

2 – The results of the shading methods should produce the effects according to the enumerated item definitions. One particular case where the effects are difficult to define is when the silhouette of a surface intersects itself or another silhouette of the same surface. In this case, the effect is implementation-dependent.

3 – The shading methods correspond to those provided by PHIGS PLUS (see annexes F and D).

EXPRESS specification:

```
*)
TYPE shading_surface_method = ENUMERATION OF
  (constant_shading,
   colour_shading,
   dot_shading,
   normal_shading);
END_TYPE;
(*
```

Enumerated item definitions:

constant_shading: a reflectance calculation is performed for each facet of the approximated surface to produce one reflected colour per facet. The point on the facet used in the calculation is implementation-dependent. The colour used in the reflectance calculation is the **surface_colour** specified in the relevant **surface_style_rendering** entity.

colour_shading: a reflectance calculation is performed at each vertex of each facet of the approximated product shape, using the **surface_colour** and the surface normals in the vertices. The resulting reflected colours are interpolated linearly across each facet.

dot_shading: any dot products needed by the reflectance equation are calculated from surface normals at a set of positions on the surface. These dot products are interpolated linearly across the surface. The reflectance calculation is performed at each interpolated position of the surface to produce a reflected colour based on the interpolated dot products and the **surface_colour** of the relevant **surface_style_rendering** entity.

normal_shading: the surface normals are interpolated linearly across the surface. The reflectance calculation is performed at each interpolated position of the surface to produce a reflected colour based on the interpolated surface normal and the **surface_colour** of the relevant **surface_style_rendering** entity.

If the **surface_colour** is specified through a **colour_specification**, the interpolation of colours shall be performed in the colour model of that specification. Otherwise the interpolation may be performed in an arbitrary model.

NOTES

4 – The result of colour interpolation depends on the colour model in which interpolation is performed.

5 – Examples of colour models are RGB, HSV, HLS.

6 – More information about colour models and colour interpolation can be found in [12], pp. 611 620.

6.3.22 rendering_properties_select

The **rendering_properties_select** is a selection between two kinds of properties for a surface: reflectance and transparency.

EXPRESS specification:

```
*)
TYPE rendering_properties_select = SELECT
  (surface_style_reflectance_ambient,
   surface_style_transparent);
END_TYPE;
(*
```

6.3.23 character_style_select

The **character_style_select** is a selection between a **character_glyph_style_stroke**, a **character_glyph_style_outline**, or a **text_style_for_defined_font**.

EXPRESS specification:

```
*)
TYPE character_style_select = SELECT
  (character_glyph_style_stroke,
   character_glyph_style_outline,
   text_style_for_defined_font);
END_TYPE;
(*
```

6.3.24 text_justification

The **text_justification** type is provided to control the justification of text.

NOTE – Application Protocols shall specify legal values of the **text_justification** and shall associate precise meaning to these values.

EXPRESS specification:

```
*)
TYPE text_justification = label;
END_TYPE;
(*
```

6.3.25 box_characteristic_select

The **box_characteristic_select** is a selection between **box_height**, **box_width**, **box_slant_angle** and **box_rotate_angle**.

EXPRESS specification:

```
*)
TYPE box_characteristic_select = SELECT
  (box_height,
   box_width,
   box_slant_angle,
   box_rotate_angle);
END_TYPE;
(*
```

6.3.26 box_height

The **box_height** is a height scaling factor used in the definition of a character glyph.

EXPRESS specification:

```
*)
TYPE box_height = positive_ratio_measure;
END_TYPE;
(*
```

6.3.27 box_width

The **box_width** is a width scaling factor used in the definition of a character glyph.

EXPRESS specification:

```
*)
TYPE box_width = positive_ratio_measure;
END_TYPE;
(*
```

6.3.28 box_slant_angle

A **box_slant_angle** is an angle which indicates that the box for a character glyph shall be presented as a parallelogram, with the angle being between the character up line and an axis perpendicular to the character base line.

NOTE – Figure 17 shows the definition of **box_slant_angle**.

EXPRESS specification:

```
*)
TYPE  box_slant_angle = plane_angle_measure;
END_TYPE;
(*
```

Informal propositions:

IP1: The **box_slant_angle** shall be between 0 and 90 degree.

6.3.29 box_rotate_angle

The **box_rotate_angle** is an angle which indicates that the box for a character glyph shall be presented at an angle to the baseline of the text string within which the glyph occurs, the angle being that between the baseline of the glyph and an axis perpendicular to the baseline of the text string.

NOTE – Figure 17 shows the definition of **box_rotate_angle**.

EXPRESS specification:

```
*)
TYPE box_rotate_angle = plane_angle_measure;
END_TYPE;
(*
```

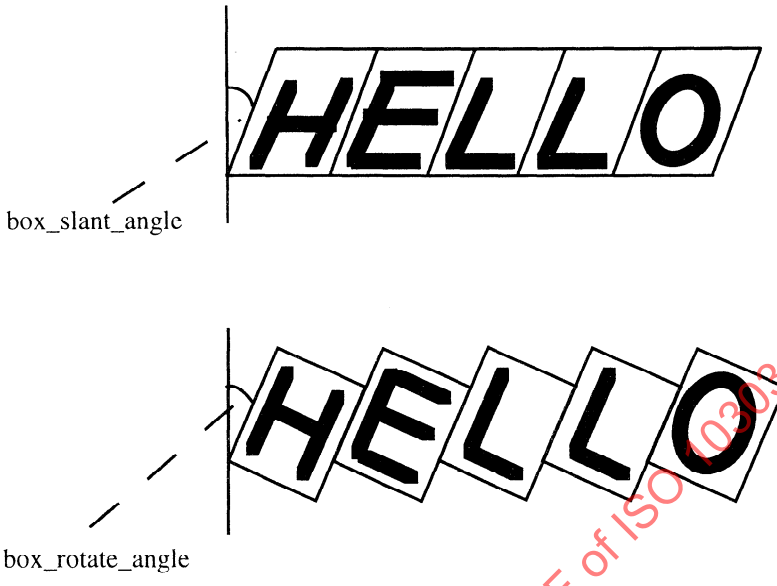


Figure 17 – Box slant and rotate angle

6.3.30 character_spacing_select

The **character_spacing_select** is a selection between a **length_measure**, a **ratio_measure**, a **measure_with_unit**, a **descriptive_measure**, and a **pre_defined_character_spacing**. It is used to specify the method of spacing between adjacent characters in a text string.

EXPRESS specification:

```
*)
TYPE character_spacing_select = SELECT
  (length_measure,
   ratio_measure,
   measure_with_unit,
   descriptive_measure,
   pre_defined_character_spacing);
END_TYPE;
(*
```

6.3.31 symbol_style_select

The **symbol_style_select** is a selection between a **symbol_element_style** and a **symbol_colour**. It is used to specify the style to be applied to elements of a symbol, or the colour to be applied to an entire symbol.

EXPRESS specification:

```
*)
TYPE symbol_style_select= SELECT
  (symbol_element_style,
   symbol_colour);
```

```
END_TYPE;
(*
```

ISSUE?

6.3.32 tolerance_select

The **tolerance_select** is used by the **approximation_tolerance** to select either an **approximation_tolerance_deviation** or an **approximation_tolerance_parameter**.

EXPRESS specification:

```
*)
TYPE tolerance_select = SELECT
  (approximation_tolerance_deviation,
   approximation_tolerance_parameter);
END_TYPE;
(*
```

6.3.33 approximation_method

The **approximation_method** is used to enumerate two possible methods for the tessellation of curves and surfaces with line segments or meshes of planar polygons.

EXPRESS specification:

```
*)
TYPE approximation_method = ENUMERATION OF
  (chordal_deviation,
   chordal_length);
END_TYPE;
(*
```

Enumerated item definitions:

chordal_deviation: curves are approximated in a way that the distance between the curve and the line segment does not exceed a given deviation value. Surfaces are approximated in a way that the distance between the surface and the polygon does not exceed a given deviation value. The distance is measured in the normal direction to the line segment or the planar polygon. The **chordal_deviation** is measured either in product shape space units or **presentation_area** units as specified by an **approximation_tolerance_deviation**.

chordal_length: curves are approximated in a way that the resulting line segments have uniform length. Surfaces are approximated in a way that the edges of the resulting planar polygons have uniform length. The **chordal_length** is measured either in product shape space units or **presentation_area** units as specified by an **approximation_tolerance_deviation**.

NOTE – Figure 18 shows the definition of **chordal_deviation** and **chordal_length**.

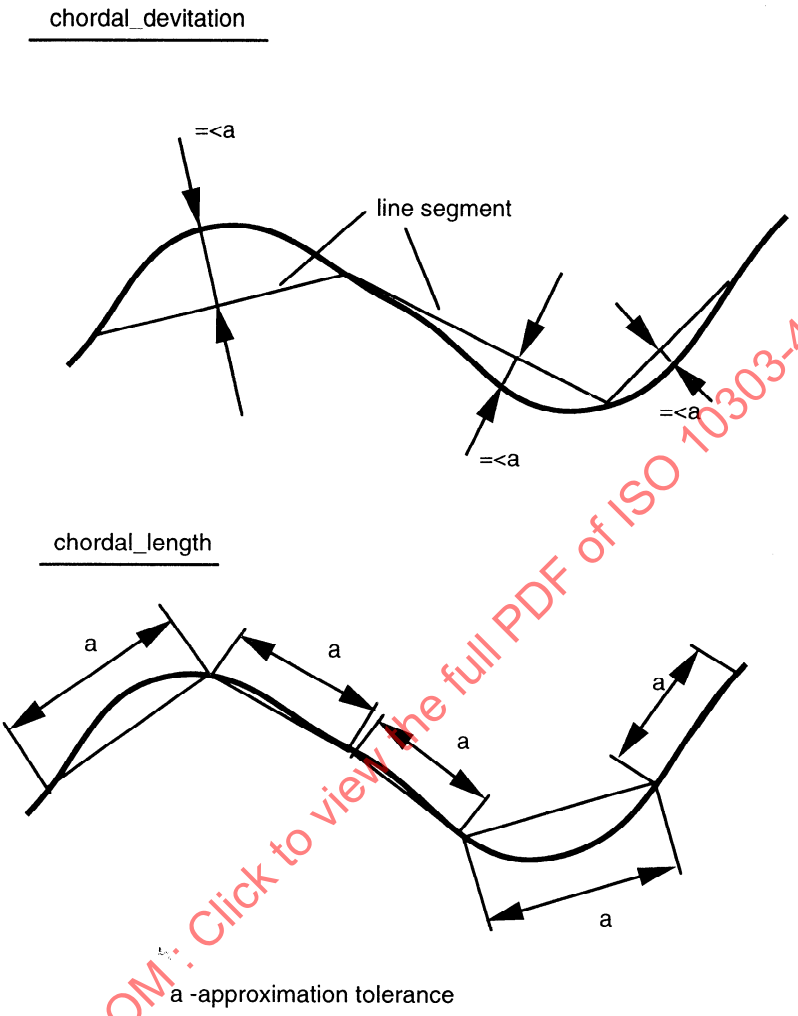


Figure 18 – Chordal deviation and length

6.3.34 tolerance_deviation_select

The **tolerance_deviation_select** is used by an **approximation_tolerance_deviation** to select either a **curve_tolerance_deviation** or a **surface_tolerance_deviation**.

EXPRESS specification:

```
*)
TYPE tolerance_deviation_select = SELECT
  (curve_tolerance_deviation,
   surface_tolerance_deviation);
END_TYPE;
(*
```

6.3.35 curve_tolerance_deviation

The **curve_tolerance_deviation** specifies an approximation tolerance for a curve by providing a deviation value.

EXPRESS specification:

```
*)
TYPE curve_tolerance_deviation = positive_length_measure;
END_TYPE;
(*
```

6.3.36 surface_tolerance_deviation

The **surface_tolerance_deviation** specifies an approximation tolerance for a surface by providing a deviation value.

EXPRESS specification:

```
*)
TYPE surface_tolerance_deviation = positive_length_measure;
END_TYPE;
(*
```

6.3.37 product_or_presentation_space

The **product_or_presentation_space** is used by an **approximation_tolerance_deviation** to specify the space in which the tolerance values are defined.

EXPRESS specification:

```
*)
TYPE product_or_presentation_space = ENUMERATION OF
  (product_shape_space,
   presentation_area_space);
END_TYPE;
(*
```

Enumerated item definitions:

product_shape_space: the definition space of the product shape item to be presented.

presentation_area_space: the coordinate space in which the **presentation_area** is founded.

6.3.38 tolerance_parameter_select

The **tolerance_parameter_select** is used by an **approximation_tolerance_parameter** to select either a **curve_tolerance_parameter** or a **surface_tolerance_parameter**.

EXPRESS specification:

```
*)
TYPE tolerance_parameter_select = SELECT
  (curve_tolerance_parameter,
```

```
   surface_tolerance_parameter);
END_TYPE;
(*
```

6.3.39 curve_tolerance_parameter

The **curve_tolerance_parameter** specifies an approximation tolerance for curves in parameter space units.

EXPRESS specification:

```
*)
TYPE curve_tolerance_parameter = REAL;
END_TYPE;
(*
```

6.3.40 surface_tolerance_parameter

The **surface_tolerance_parameter** specifies an approximation tolerance for surfaces in parameter space units.

EXPRESS specification:

```
*)
TYPE surface_tolerance_parameter = REAL;
END_TYPE;
(*
```

6.3.41 hiding_or_blanking_select

The **hiding_or_blanking_select** selects the entities which can hide or blank other entities in a presentation.

EXPRESS specification:

```
*)
TYPE hiding_or_blanking_select = SELECT
  (presentation_area,
   presentation_view,
   product_data_representation_view,
   annotation_fill_area,
   area_dependent_annotation_representation,
   view_dependent_annotation_representation,
   annotation_text_with_delineation,
   character_glyph_symbol_stroke,
   character_glyph_symbol_outline,
   symbol_representation_with_blanking_box);
END_TYPE;
(*
```

6.3.42 invisibility_context

The **invisibility_context** type selects the context in which elements of a picture may be denoted as invisible.

EXPRESS specification:

```
*)
TYPE invisibility_context= SELECT
  (presentation_layer_usage,
   presentation_representation,
   presentation_set);
END_TYPE;
(*
```

6.3.43 invisible_item

The **invisible_item** type selects elements of a picture that may be denoted as invisible.

EXPRESS specification:

```
*)
TYPE invisible_item = SELECT
  (styled_item,
   presentation_layer_assignment,
   presentation_representation);
END_TYPE;
(*
```

6.4 Presentation appearance schema entity definitions: style assignment

6.4.1 styled_item

A **styled_item** is a **representation_item** with associated presentation style.

EXPRESS specification:

```
*)
ENTITY styled_item
  SUBTYPE OF (representation_item);
  styles :  SET [1:?] OF presentation_style_assignment;
  item   :  representation_item;
WHERE
  WR1: (SIZEOF(SELF.styles) = 1)
                XOR
       (SIZEOF(QUERY(pres_style <* SELF.styles |
         NOT ('PRESENTATION_APPEARANCE_SCHEMA.' +
              'PRESENTATION_STYLE_BY_CONTEXT' IN
         TYPEOF(pres_style))
         )) = 0);
END_ENTITY;
(*
```

Attribute definitions:

styles: the styles assigned to the item.

item: the item to which styles are assigned.

Formal propositions:

WR1: The set **styles** shall contain only one style or all members of the set shall be **presentation_style_by_context** entities.

NOTE – This is to ensure that there are no style conflicts; more than one style may be specified only when the context in which each style applies is given.

6.4.2 over_riding_styled_item

A **over_riding_styled_item** is a **styled_item** where the style assignment takes precedence over another assigned style. The precedence happens when the **over_ridden_style styled_item** and the **over_riding_styled_item** are both included, directly or indirectly, in the same presentation.

EXAMPLE 10 – A circle instance is used by a **geometric_curve_set** which is an item in a **representation**. A **styled_item** instance is an item in the same **representation**. That **styled_item** has the **geometric_curve_set** as its item, and a **presentation_style_assignment** with a **curve_style** which has a colour of blue. An instance of **over_riding_styled_item** is also an item in the same representation. That **over_riding_styled_item** has the circle instance as its item, the **styled_item** instance as its **over_ridden_style**, and a **presentation_style_assignment** with a **curve_style** which has a colour of red. The red colour for the circle takes precedence over the blue colour of the **geometric_curve_set** for a presentation of the **geometric_curve_set**.

EXPRESS specification:

```
*)
ENTITY over_riding_styled_item
  SUBTYPE OF (styled_item);
  over_ridden_style : styled_item;
END_ENTITY;
(*
```

Attribute definitions:

over_ridden_style: the **styled_item** that will have its style overridden.

6.4.3 context_dependent_over_riding_styled_item

A **context_dependent_over_riding_styled_item** is an **over_riding_styled_item** where the style assignment takes precedence over another assigned style based on a **representation**, **representation_item**, or combination of **representation** and **representation_item** in which the item being styled is used.

EXAMPLE 11 – A circle instance is used by **geometric_curve_set** instance 1 which is an item in **representation** instance 1. In this example the instance of circle represents the head of a screw in a door hinge. A **styled_item** instance is an item in **representation** instance 1. That **styled_item** has the **geometric_curve_set** as its item, and a **presentation_style_assignment** with a **curve_style** which has a colour of blue. The **representation** instance 1 is included in **representation**

instance 2 through the use of **representation_map** instance 1 and **mapped_item** instance 1, as one hinge on a door. The **representation** instance 1 is included in a different location in **representation** instance 2 through **representation_map** 2 and **mapped_item** instance 2 as a second hinge. An instance of **context_dependent_over_riding_styled_item** is also an item in **representation** instance 2. That **context_dependent_over_riding_styled_item** has the circle instance as its item, the **styled_item** instance as its **over_ridden_style**, a **presentation_style_assignment** with a **curve_style** which has a colour of red, and a **style_context** of **mapped_item** instance 1. The red color for the circle takes precedence over the blue color of the **geometric_curve_set** for a **representation** of the **geometric_curve_set** as it is included in the presentation through **mapped_item** instance 1. A presentation of **representation** instance 2 would have the **geometric_curve_set** presented in two different places, the first having a red screw head and all the other curves blue, the second having all the curves blue.

EXPRESS specification:

```
*)
ENTITY context_dependent_over_riding_styled_item
  SUBTYPE OF(over_riding_styled_item);
  style_context : SET[1:2] OF style_context_select;
WHERE
  WR1: (SIZEOF(QUERY( sc <* SELF.style_context |
       'REPRESENTATION_SCHEMA.REPRESENTATION' IN
         TYPEOF(sc))) = 1 )
                AND
       (SIZEOF(QUERY( sc <* SELF.style_context |
       'REPRESENTATION_SCHEMA.REPRESENTATION_ITEM' IN
         TYPEOF(sc))) = 1);
END_ENTITY;
(*
```

Attribute definitions:

style_context: a set of one or two contexts for the overriding of the overridden style.

Formal propositions:

WR1: There shall be no more than one **representation** and one **representation_item** in the **style_context** set.

6.4.4 presentation_style_assignment

A **presentation_style_assignment** is a set of styles which are assigned to a **representation_item** for the purpose of presenting the item. Style definitions have an effect only on the appearance of an element of a special type. Surface style has only an effect on surfaces. Fill area style and curve style have only an effect on curves and surfaces. Point style has an effect on points, curves, and surfaces. Text style has only an effect on the appearance of annotation text. Symbol style has only an effect on symbols.

EXAMPLE 12 – If a line is given a style which is a curve style, it shall appear. If a line is given both curve and point style, both the curve and its related cartesian points shall appear.

EXPRESS specification:

```
*)
ENTITY presentation_style_assignment;
```

```
  styles : SET [1:?] OF presentation_style_select;
WHERE
  WR1: SIZEOF (QUERY (style1 <* SELF.styles |
         NOT (SIZEOF (QUERY (style2 <* (SELF.styles - style1) |
           NOT ((TYPEOF (style1) <> TYPEOF (style2)) OR
             (SIZEOF (['PRESENTATION_APPEARANCE_SCHEMA.' +
                       'SURFACE_STYLE_USAGE',
                       'PRESENTATION_APPEARANCE_SCHEMA.'+
                       'EXTERNALLY_DEFINED_STYLE'] *
                     TYPEOF (style1)) = 1)
           ))) = 0
         ))) = 0;
  WR2: SIZEOF (QUERY (style1 <* SELF.styles |
         'PRESENTATION_APPEARANCE_SCHEMA.SURFACE_STYLE_USAGE' IN
         TYPEOF(style1)
         )) <= 2;
END_ENTITY;
(*
```

Attribute definitions:

styles: the set of presentation styles that are assigned to a **representation_item.**

Formal propositions:

WR1: The same style shall not appear more than once in the set of styles, except for **externally_defined_style** and **surface_style_usage.**

WR2: **surface_style_usage** shall not occur more than twice in the set of styles.

Informal propositions:

IP1: Externally defined styles shall not conflict with other styles in the same **presentation_style_assignment** entity, including other externally defined styles.

> NOTE – For one style to conflict with the other, it specifies a different style for the same characteristic, such as colour or width. For example, one style might say blue, the other green, and both be applied to the same entity.

IP2: Each style type is unique.

IP3: If there are two instances of **surface_style_usage** in the set of styles, each shall specify the style for opposite sides of the surface being styled.

6.4.5 presentation_style_by_context

A **presentation_style_by_context** is a **presentation_style_assignment** which is assigned to a **representation_item** and is applicable only in a specified presentation context.

EXPRESS specification:

```
*)
ENTITY presentation_style_by_context
  SUBTYPE OF (presentation_style_assignment);
```

```
  style_context : style_context_select;
END_ENTITY;
(*
```

Attribute definitions:

style_context: the presentation context in which a style is assigned to a **representation_item**.

6.4.6 pre_defined_presentation_style

A **pre_defined_presentation_style** may be used to fix certain application-specific aspects of appearance attributes defined in this schema.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_presentation_style
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.4.7 externally_defined_style

An **externally_defined_style** is an external reference to a presentation style.

EXPRESS specification:

```
*)
ENTITY externally_defined_style
  SUBTYPE OF (externally_defined_item);
END_ENTITY;
(*
```

6.5 Presentation appearance schema entity definitions: presentation styles for points

6.5.1 point_style

A **point_style** specifies the visual appearance of points.

EXPRESS specification:

```
*)
ENTITY point_style;
  name          : label;
  marker        : marker_select;
  marker_size   : size_select;
  marker_colour : colour;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **point_style** is referred to.

marker: the kind of marker which shall be used to present a point.

marker_size: the size in **presentation_area** units used for drawing the marker.

marker_colour: the colour to be applied to the marker.

6.5.2 pre_defined_marker

A **pre_defined_marker** may be used to define an application-specific marker symbol.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_marker
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.5.3 pre_defined_size

A **pre_defined_size** may be used to define an application-specific size for markers.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_size
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.6 Presentation appearance schema entity definitions: presentation styles for curves

6.6.1 curve_style

A **curve_style** specifies the visual appearance of a curve.

EXPRESS specification:

```
*)
ENTITY curve_style;
  name         : label;
  curve_font   : curve_font_or_scaled_curve_font_select;
  curve_width  : size_select;
  curve_colour : colour;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **curve_style** is referred to.

curve_font: the **curve_style_font**, scaled **curve_style_font**, **pre_defined_curve_font**, scaled **pre_defined_curve_font**, **externally_defined_curve_font**, or scaled **externally_defined_curve_font** which is used to present a curve.

curve_width: the width of the visible part of the presented curve in **presentation_area** units.

curve_colour: the colour of the visible part of the curve.

6.6.2 curve_style_with_ends_and_corners

A **curve_style** which specifies the visual appearance of the ends and corners of a curve.

EXPRESS specification:

```
*)
ENTITY curve_style_with_ends_and_corners
  SUBTYPE OF (curve_style);
  curve_ends    : squared_or_rounded;
  curve_corners : squared_or_rounded;
END_ENTITY;
(*
```

Attribute definitions:

curve_ends: an indication of how to present the ends of a curve.

curve_corners: an indication of how to present the corners of a curve.

6.6.3 curve_style_with_extension

A **curve_style_with_extensions** is a **curve_style** that indicates how the curve ends shall appear when presented, either extended or shortened.

EXPRESS specification:

```
*)
ENTITY curve_style_with_extension
  SUBTYPE OF (curve_style);
  curve_extensions   : length_measure;
END_ENTITY;
(*
```

Attribute definitions:

curve_extensions: a **length_measure** indicating how to lengthen or shorten the ends of a curve. If the **length_measure** is positive, the curve shall be extended at both ends in a tangential direction by the specified length in **presentation_area** units. If the **length_measure** is negative, the curve shall be shortened by the absolute value of the specified length in **presentation_area** units.

NOTE – See figure 19.

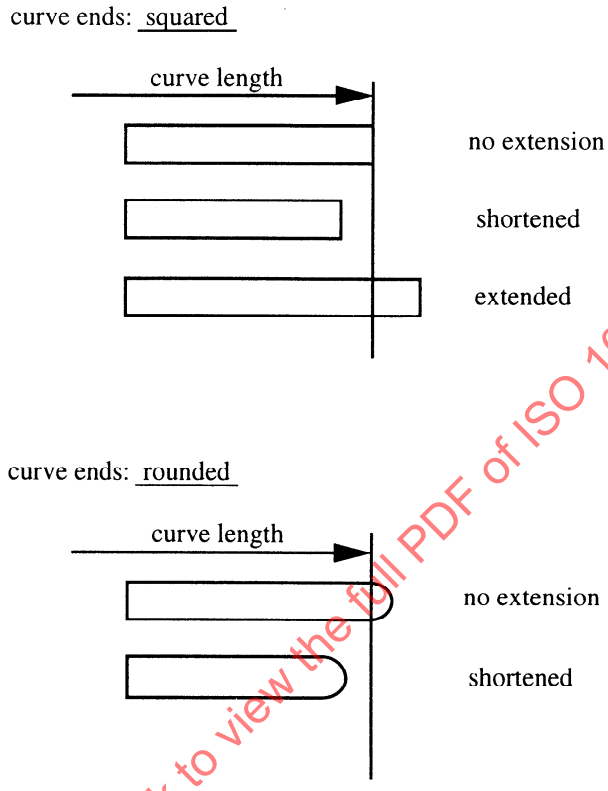


Figure 19 – Curve style with extension

6.6.4 pre_defined_curve_font

A **pre_defined_curve_font** may be used to define application–specific curve fonts.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_curve_font
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.6.5 externally_defined_curve_font

An **externally_defined_curve_font** makes an external reference to a curve font.

EXPRESS specification:

```
*)
ENTITY externally_defined_curve_font
  SUBTYPE OF (externally_defined_item);
END_ENTITY;
(*
```

6.6.6 curve_style_font

A **curve_style_font** combines several **curve_style_font_patterns** into a pattern. The resulting pattern is repeated along the curve.

EXPRESS specification:

```
*)
ENTITY curve_style_font;
  name         : label;
  pattern_list : LIST [1:?] OF curve_style_font_pattern;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **curve_style_font** is referred to.

pattern_list: a list of **curve_style_font_patterns** that contains the patterns used for drawing curves. The patterns are applied in the order in which they occur in the list.

Informal propositions:

IP1: The curve starts always with a complete pattern.

IP2: The font pattern is clipped off at the end of the curve and may therefore be incomplete.

6.6.7 curve_style_font_pattern

A **curve_style_font_pattern** is a pair of visible and invisible curve segment lengths measured in **presentation_area** units.

EXPRESS specification:

```
*)
ENTITY curve_style_font_pattern;
  visible_segment_length   : positive_length_measure;
  invisible_segment_length : positive_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

visible_segment_length: the length of the visible segment in the pattern definition measured in **presentation_area** units.

invisible_segment_length: the length of the invisible segment in the pattern definition measured in **presentation_area** units.

6.6.8 curve_style_wide

A **curve_style_wide** defines a style for filling the visible curve segments. The **curve_style** used for styling the tile curves or hatch lines is also applied to the boundary of the curve segments.

EXPRESS specification:

```
*)
ENTITY curve_style_wide
  SUBTYPE OF (curve_style_font);
  interior_style : fill_area_style;
END_ENTITY;
(*
```

Attribute definitions:

interior_style: the style for filling the visible curve segments with tiles or hatches.

6.6.9 curve_style_curve_pattern_set

A **curve_style_curve_pattern_set** defines a style for filling the visible curve segments with a set of repetitive patterns. This **pattern_set** is repeated along the visible curve segments.

EXPRESS specification:

```
*)
ENTITY curve_style_curve_pattern_set
  SUBTYPE OF (curve_style_font,
              geometric_representation_item);
  pattern_set : SET [1:?] OF curve_style_curve_pattern;
END_ENTITY;
(*
```

Attribute definitions:

pattern_set: the repetitive **pattern_set** consists of a set of **curve_style_curve_patterns**.

Informal propositions:

IP1: The curve starts always with a complete pattern.

IP2: The curve pattern is clipped off at the end of the curve and may therefore be incomplete.

6.6.10 curve_style_curve_pattern

A **curve_style_curve_pattern** specifies a curve pattern which is used in the **curve_style_curve_pattern_set**. The **curve_style_curve_pattern** is defined in a local pattern definition coordinate system, which will be placed along the visible segments of the derived curve. The projected curve tangent is the x axis and the normal to the projected curve is the y axis of the local pattern definition coordinate system.

NOTE – Figure 20 shows the definition of **curve_style_curve_pattern**.

EXPRESS specification:

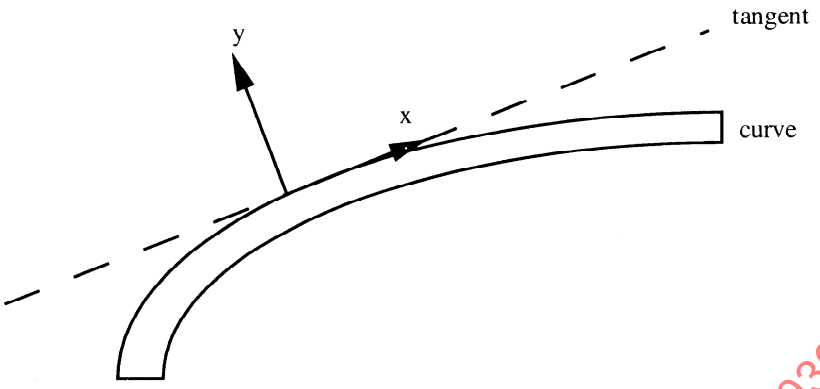


Figure 20 – Curve style curve pattern

```
*)
ENTITY curve_style_curve_pattern
  SUBTYPE OF (geometric_representation_item);
  pattern        : annotation_curve_occurrence;
  pattern_length : positive_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

pattern: the repetitive pattern for filling the curve. The pattern is defined as an **annotation_curve_occurrence** and is therefore itself associated with a presentation_style.

pattern_length: the length of the pattern in **presentation_area** units.

6.6.11 curve_style_font_and_scaling

A **curve_style_font_and_scaling** is used to apply scale to the specified **curve_style_font**.

EXPRESS specification:

```
*)
ENTITY curve_style_font_and_scaling;
  name               : label;
  curve_font         : curve_style_font_select;
  curve_font_scaling : REAL;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **curve_style_font_and_scaling** is referred to.

curve_font: the **curve_font** to be scaled.

curve_font_scaling: the scale factor.

6.7 Presentation appearance schema entity definitions: presentation styles for fill areas

6.7.1 fill_area_style

A style for filling visible curve segments, annotation fill areas, or surfaces with tiles or hatching.

EXPRESS specification:

```
*)
ENTITY fill_area_style;
  name       : label;
  fill_styles : SET [1:?] OF fill_style_select;
WHERE
  WR1: SIZEOF(QUERY(fill_style <* SELF.fill_styles |
         'PRESENTATION_APPEARANCE_SCHEMA.'+
         'FILL_AREA_STYLE_COLOUR' IN
         TYPEOF(fill_style)
         )) <= 1;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **fill_area_style** is referred to.

fill_styles: the set of fill area styles to use in presenting visible curve segments, annotation fill areas, or surfaces.

Formal propositions:

WR1: There shall be not more than one **fill_area_style_colour** in the **fill_styles** set.

6.7.2 fill_area_style_colour

A **fill_area_style_colour** defines a colour to be used for solid fill of visible curve segments, annotation fill areas, or surfaces.

EXPRESS specification:

```
*)
ENTITY fill_area_style_colour;
  name        : label;
  fill_colour : colour;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **fill_area_style_colour** is referred to.

fill_colour: the colour to be used for filling the area.

6.7.3 pre_defined_hatch_style

A **pre_defined_hatch_style** is a hatching style provided for Application Protocols to define an application–specific single or multiple hatching style.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_hatch_style
  SUBTYPE OF (pre_defined_item, geometric_representation_item);
END_ENTITY;
(*
```

6.7.4 externally_defined_hatch_style

A **externally_defined_hatch_style** makes an external reference to a hatching style.

EXPRESS specification:

```
*)
ENTITY externally_defined_hatch_style
  SUBTYPE OF (externally_defined_item, geometric_representation_item);
END_ENTITY;
(*
```

6.7.5 fill_area_style_hatching

A **fill_area_style_hatching** defines a styled pattern of curves for hatching visible curve segments, annotation fill areas, or surfaces.

EXPRESS specification:

```
*)
ENTITY fill_area_style_hatching
  SUBTYPE OF (geometric_representation_item);
  hatch_line_appearance          : curve_style;
  start_of_next_hatch_line       : one_direction_repeat_factor;
  point_of_reference_hatch_line : cartesian_point;
  pattern_start                  : cartesian_point;
  hatch_line_angle               : plane_angle_measure;
END_ENTITY;
(*
```

Attribute definitions:

hatch_line_appearance: the **curve_style** of the hatching lines. Any **curve_style** pattern shall start at the origin of each hatch line. The origin of the reference hatch line is specified by **pattern_start**. The origin of any other hatch line is determined by adding a multiple of **start_of_next_hatch_line** to **pattern_start**.

start_of_next_hatch_line: the displacement between adjacent hatch lines, specified as a vector.

point_of_reference_hatch_line: the origin for mapping the **fill_area_style_hatching** onto a curve, annotation fill area, or surface.

pattern_start: the start point for the **curve_style** of the **reference_hatch_line**.

hatch_line_angle: the angle determining the direction of the parallel hatching lines.

NOTE – Figure 21 shows the definition of **fill_area_style_hatching**.

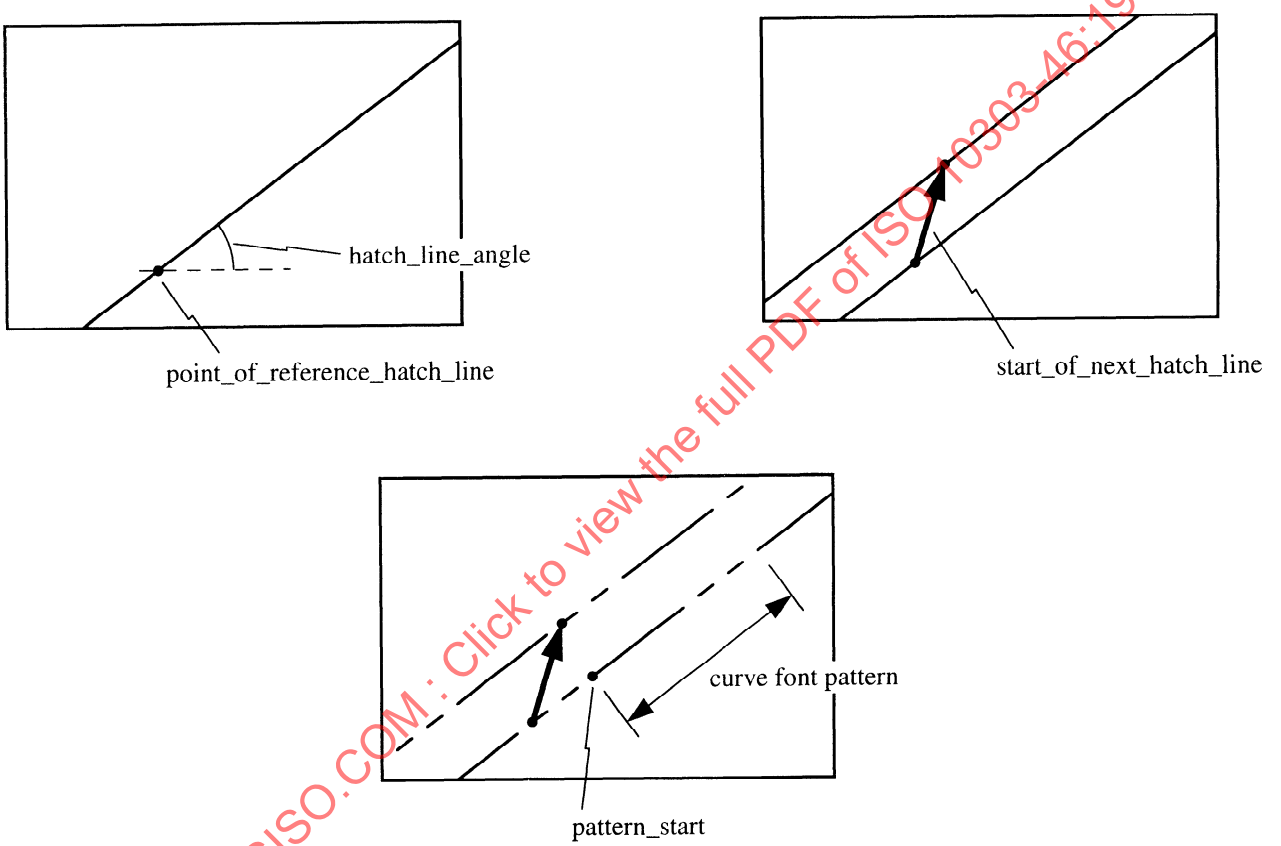


Figure 21 – Fill area style hatching

6.7.6 pre_defined_tile_style

A **pre_defined_tile_style** is a tile style provided for Application Protocols to define an application–specific tile style.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_tile_style
  SUBTYPE OF (pre_defined_item, geometric_representation_item);
```

```
END_ENTITY;
(*
```

6.7.7 externally_defined_tile_style

An **externally_defined_tile_style** makes an external reference to a tiling style.

EXPRESS specification:

```
*)
ENTITY externally_defined_tile_style
  SUBTYPE OF (externally_defined_item, geometric_representation_item);
END_ENTITY;
(*
```

6.7.8 fill_area_style_tiles

A **fill_area_style_tiles** defines a two–dimensional tile to be used for the filling of annotation fill areas or other closed regions. The content of a tile is defined by the **tiles** set, and the placement of each tile is determined by the **tiling_pattern** which indicates how to place tiles next to each other. Tiles or parts of tiles outside the annotation fill area or closed region shall be clipped at the boundaries of the area or region.

EXPRESS specification:

```
*)
ENTITY fill_area_style_tiles
  SUBTYPE OF (geometric_representation_item);
  tiling_pattern : two_direction_repeat_factor;
  tiles          : SET [1:?] OF fill_area_style_tile_shape_select;
  tiling_scale   : positive_ratio_measure;
END_ENTITY;
(*
```

Attribute definitions:

tiling_pattern: the **two_direction_repeat_factor** defining the shape and relative positioning of the tiles.

tiles: the set of constituents of the tile.

tiling_scale: the scaling factor applied to each tile as it is placed in the **annotation_fill_area**.

6.7.9 fill_area_style_tile_curve_with_style

A **fill_area_style_tile_curve_with_style** contains a styled curve which acts as a constituent in a **fill_area_style_tiles**.

EXPRESS specification:

```
*)
ENTITY fill_area_style_tile_curve_with_style
  SUBTYPE OF (geometric_representation_item);
```

```
  styled_curve : annotation_curve_occurrence;
END_ENTITY;
(*
```

Attribute definitions:

styled_curve: the two–dimensional styled curve defined in the local coordinate system of a fill area tile.

6.7.10 fill_area_style_tile_coloured_region

A **fill_area_style_tile_coloured_region** is a closed curve that is filled with a colour and acts as a constituent in a **fill_area_style_tiles**.

EXPRESS specification:

```
*)
ENTITY fill_area_style_tile_coloured_region
  SUBTYPE OF (geometric_representation_item);
  closed_curve  : curve_or_annotation_curve_occurrence;
  region_colour : colour;
END_ENTITY;
(*
```

Attribute definitions:

closed_curve: the closed curve which defines a coloured region. The referenced curve may also be associated with style.

region_colour: the colour of that region which is defined by the interior of the closed curve.

Informal propositions:

IP1: The **closed_curve** shall be closed and not self–intersecting.

6.7.11 fill_area_style_tile_symbol_with_style

A **fill_area_style_tile_symbol_with_style** is a symbol which acts as a constituent in a **fill_area_style_tiles**.

EXPRESS specification:

```
*)
ENTITY fill_area_style_tile_symbol_with_style
  SUBTYPE OF (geometric_representation_item);
  symbol : annotation_symbol_occurrence;
END_ENTITY;
(*
```

Attribute definitions:

symbol: a styled annotation symbol.

6.7.12 pre_defined_tile

A **pre_defined_tile** may be used to define an application–specific tile.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_tile
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.7.13 externally_defined_tile

An **externally_defined_tile** is a tile that is defined by reference to some external source.

EXPRESS specification:

```
*)
ENTITY externally_defined_tile
  SUBTYPE OF (externally_defined_item);
END_ENTITY;
(*
```

6.7.14 one_direction_repeat_factor

A **one_direction_repeat_factor** is a vector used in a **fill_area_style_hatching** for determining the origin of a repeated hatch line relative to the origin of the previous hatch line. Given the initial position I of any hatch line, the **one_direction_repeat_factor** R determines two new positions according to the expression:

$$I + k \cdot R \qquad k = -1, 1$$

NOTE – Figure 22 shows the positions defined by a **one_direction_repeat_factor**.

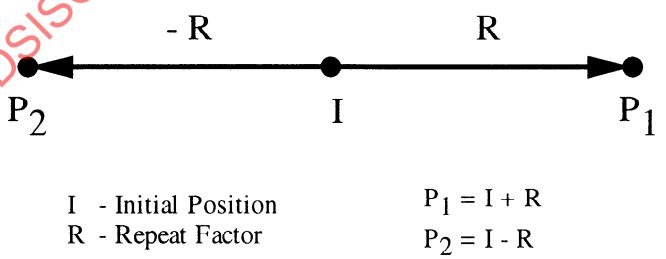


Figure 22 – One direction repeat factor

EXPRESS specification:

```
*)
ENTITY one_direction_repeat_factor
  SUBTYPE OF (geometric_representation_item);
  repeat_factor : vector;
```

```
END_ENTITY;
(*
```

Attribute definitions:

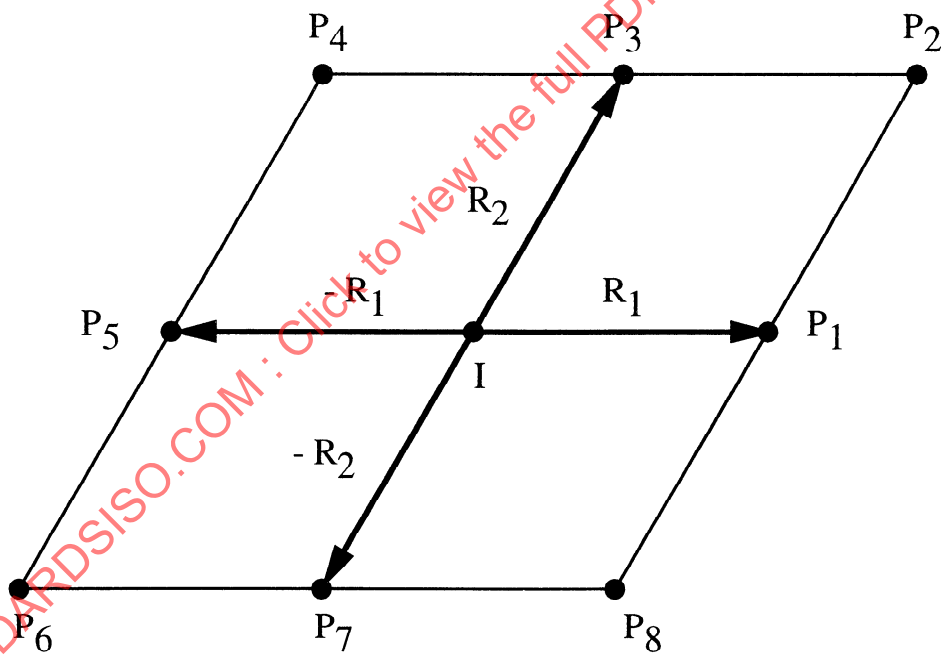
repeat_factor: the vector which specifies the relative positioning of hatch lines.

6.7.15 two_direction_repeat_factor

A **two_direction_repeat_factor** combines two vectors which are used in a **fill_area_style_tiles** for determining the shape and relative location of tiles. Given the initial position I of any tile, the **two_direction_repeat_factor** $R = (R_1, R_2)$ determines eight new positions according to the expression:

$$I + k_1 \cdot R_1 + k_2 \cdot R_2 \qquad k_1, k_2 = -1, 0, 1; \; k_1^2 + k_2^2 \neq 0$$

NOTE – Figure 23 shows the positions defined by a **two_direction_repeat_factor**.



I - Initial Position
R_1 - Repeat Factor
R_2 - Second Repeat Factor

$P_1 = I + R_1$
$P_2 = I + R_1 + R_2$
$P_3 = I + R_2$
$P_4 = I - R_1 + R_2$
$P_5 = I - R_1$
$P_6 = I - R_1 - R_2$
$P_7 = I - R_2$
$P_8 = I + R_1 - R_2$

Figure 23 – Two direction repeat factor

EXPRESS specification:

```
*)
ENTITY two_direction_repeat_factor
  SUBTYPE OF (one_direction_repeat_factor);
  second_repeat_factor : vector;
END_ENTITY;
(*
```

Attribute definitions:

second_repeat_factor: the vector which specifies the relative positioning of tiles in the second direction.

6.8 Presentation appearance schema entity definitions: presentation styles for surfaces

6.8.1 surface_style_usage

A **surface_style_usage** is the application of a **surface_side_style_select** to the positive side, negative side, or both sides of a surface.

EXPRESS specification:

```
*)
ENTITY surface_style_usage;
  side  : surface_side;
  style : surface_side_style_select;
END_ENTITY;
(*
```

Attribute definitions:

side: the indication of which side of the surface to apply the style.

style: the style which shall be applied to the surface.

6.8.2 pre_defined_surface_side_style

A **pre_defined_surface_side_style** may be used to define application-specific **surface_side_styles**.

NOTE - Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_surface_side_style
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.8.3 surface_side_style

A **surface_side_style** is a collection of surface styles used in the presentation of the side of a

surface.

EXPRESS specification:

```
*)
ENTITY surface_side_style;
  name   : label;
  styles : SET [1:7] OF surface_style_element_select;
WHERE
  WR1: SIZEOF(QUERY( style1 <* SELF.styles |
         SIZEOF(QUERY( style2 <* SELF.styles - style1 |
           TYPEOF(style1) = TYPEOF(style2)
           )) > 0
         )) = 0;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **surface_side_style** is referred to.

styles: a collection of different surface styles.

Formal propositions:

WR1: All of the styles shall be of different types.

6.8.4 surface_style_fill_area

A **surface_style_fill_area** is the surface style that presents a surface by mapping a fill area onto the surface.

EXPRESS specification:

```
*)
ENTITY surface_style_fill_area;
  fill_area : fill_area_style;
END_ENTITY;
(*
```

Attribute definitions:

fill_area: the **fill_area_style** associated with the two-dimensional parameter space of a surface that will be mapped onto the surface itself.

6.8.5 surface_style_boundary

A **surface_style_boundary** is the surface style that is applied to the boundary curves of a surface.

EXPRESS specification:

```
*)
ENTITY surface_style_boundary;
  style_of_boundary : curve_or_render;
END_ENTITY;
(*
```

Attribute definitions:

style_of_boundary: the style for the boundary curves of a surface.

6.8.6 curve_style_rendering

A **curve_style_rendering** allows the visualization of curves on a surface by application of rendering techniques.

EXPRESS specification:

```
*)
ENTITY curve_style_rendering;
  rendering_method     : shading_curve_method;
  rendering_properties : surface_rendering_properties;
END_ENTITY;
(*
```

Attribute definitions:

rendering_method: specifies the method which shall be used for interpolating colours along curves on a surface.

rendering_properties: specifies the rendering properties of the surface which contains the curves.

6.8.7 surface_rendering_properties

Surface_rendering_properties define those properties of a surface which are required to compute a realistic visualization of surfaces by use of rendering techniques.

EXPRESS specification:

```
*)
ENTITY surface_rendering_properties;
  rendered_colour : colour;
END_ENTITY;
(*
```

Attribute definitions:

rendered_colour: the presentation colour of a surface for use during the rendering process.

6.8.8 surface_style_silhouette

A **surface_style_silhouette** is the surface style that is applied to the silhouette curves of a surface.

EXPRESS specification:

```
*)
ENTITY surface_style_silhouette;
  style_of_silhouette : curve_or_render;
END_ENTITY;
(*
```

Attribute definitions:

style_of_silhouette: the style for the silhouette curves within a surface.

6.8.9 surface_style_segmentation_curve

A **surface_style_segmentation_curve** is the surface style that is applied to the curves on the segment borders of a surface.

EXPRESS specification:

```
*)
ENTITY surface_style_segmentation_curve;
  style_of_segmentation_curve : curve_or_render;
END_ENTITY;
(*
```

Attribute definitions:

style_of_segmentation_curve: the style for the segmentation curves of a surface.

NOTE – This style has only an effect on surfaces which have segmentation curves. These surfaces include the following types:

– B–spline surfaces;

– rectangular trimmed surfaces, curve bounded surfaces, rectangular composite surfaces, surface patches, offset surfaces, and surface replicas which refer to a B–spline surface as basis or parent surface.

6.8.10 surface_style_control_grid

A **surface_style_control_grid** is the surface style that is applied to the mesh of control points which are used for definition of surfaces.

EXPRESS specification:

```
*)
ENTITY surface_style_control_grid;
  style_of_control_grid : curve_or_render;
END_ENTITY;
(*
```

Attribute definitions:

style_of_control_grid: the style for the control grid of a surface.

NOTE – This style has only an effect on surfaces which are defined over a mesh of control points. These surfaces include the following types:

– B spline surfaces;

– rectangular trimmed surfaces, curve bounded surfaces, rectangular composite surfaces, surface patches, offset surfaces, and surface replicas which refer to a B–spline surface as basis or parent surface.

6.8.11 surface_style_parameter_line

A **surface_style_parameter_line** is the surface style that is applied to the iso-parameter lines on a surface.

EXPRESS specification:

```
*)
ENTITY surface_style_parameter_line;
  style_of_parameter_lines : curve_or_render;
  direction_counts         : SET [1:2] OF direction_count_select;
WHERE
  WR1: (HIINDEX(SELF.direction_counts) = 1)
                        XOR
       (TYPEOF(SELF.direction_counts[1]) <>
          TYPEOF(SELF.direction_counts[2]));
END_ENTITY;
(*
```

Attribute definitions:

style_of_parameter_lines: the style for the iso-parameter curves of a surface.

direction_counts: a set of **u_direction_count** and **v_direction_count** indicating the number of iso-parameter curves in the u and v directions.

Formal propositions:

WR1: If there are two members of the **direction_counts** set, they shall not be of the same type.

6.8.12 surface_style_rendering

A **surface_style_rendering** allows the realistic visualization of surfaces by use of rendering techniques. The effect of the reflectance calculation shall be as if the calculation is performed in the coordinate-system in which the **camera_model** is founded.

EXPRESS specification:

```
*)
ENTITY surface_style_rendering;
  rendering_method : shading_surface_method;
  surface_colour   : colour;
END_ENTITY;
(*
```

Attribute definitions:

rendering_method: the method for interpolating colours across surfaces.

surface_colour: the colour used to render the surface.

6.8.13 surface_style_rendering_with_properties

A **surface_style_rendering_with_properties** allows the realistic visualization of surfaces with

properties which determine transparency and reflection characteristics.

EXPRESS specification:

```
*)
ENTITY surface_style_rendering_with_properties
  SUBTYPE OF (surface_style_rendering);
  properties : SET [1:2] OF rendering_properties_select;
WHERE
  WR1: (HIINDEX(SELF.properties) = 1)
                    XOR
       (TYPEOF(SELF.properties[1]) <> TYPEOF(SELF.properties[2]));
END_ENTITY;
(*
```

Attribute definitions:

properties: the collection of rendering properties for a surface.

Formal propositions:

WR1: All of the properties shall be of different types.

6.8.14 surface_style_reflectance_ambient

A **surface_style_reflectance_ambient** specifies the ambient part of the reflectance behaviour of a surface.

NOTES

1 – The reflectance calculation is conceptually applied at one or more points on a surface being lit and shaded and produces a colour at such points. Input to the reflectance calculation includes the position at which the reflectance equation is to be applied, the surface normal, the surface colour at that position, the light sources, and the three–dimensional camera model.

2 – Suggested reflectance equations can be found in Annex D.

EXPRESS specification:

```
*)
ENTITY surface_style_reflectance_ambient;
  ambient_reflectance : REAL;
END_ENTITY;
(*
```

Attribute definitions:

ambient_reflectance: the reflectance coefficient for the ambient part of the reflectance equation.

6.8.15 surface_style_reflectance_ambient_diffuse

A **surface_style_reflectance_ambient_diffuse** specifies the diffuse part of the reflectance behaviour of a surface.

NOTE – Suggested reflectance equations can be found in Annex D.

EXPRESS specification:

```
*)
ENTITY surface_style_reflectance_ambient_diffuse
  SUBTYPE OF (surface_style_reflectance_ambient);
  diffuse_reflectance : REAL;
END_ENTITY;
(*
```

Attribute definitions:

diffuse_reflectance: the reflectance coefficient for the diffuse part of the reflectance equation.

6.8.16 surface_style_reflectance_ambient_diffuse_specular

The **surface_style_reflectance_ambient_diffuse_specular** specifies the specular part of the reflectance behaviour of a surface.

NOTE – Suggested reflectance equations can be found in Annex D.

EXPRESS specification:

```
*)
ENTITY surface_style_reflectance_ambient_diffuse_specular
  SUBTYPE OF (surface_style_reflectance_ambient_diffuse);
  specular_reflectance : REAL;
  specular_exponent    : REAL;
  specular_colour      : colour;
END_ENTITY;
(*
```

Attribute definitions:

specular_reflectance: the reflectance coefficient for the specular part of the reflectance equation.

specular_exponent: the exponent for the specular part of the reflectance equation.

specular_colour: the colour for the specular part of the reflectance equation.

6.8.17 surface_style_transparent

A **surface_style_transparent** is the surface rendering property that specifies the degree of transparency of a surface.

EXPRESS specification:

```
*)
ENTITY surface_style_transparent;
  transparency : REAL;
WHERE
  WR1: {0.0 <= transparency <= 1.0};
END_ENTITY;
(*
```

Attribute definitions:

transparency: the degree of transparency indicated by the percentage of light traversing the surface.

Formal propositions:

WR1: The transparency shall be between 0.0 and 1.0.

6.9 Presentation appearance schema entity definitions: presentation styles for text

6.9.1 text_style

A **text_style** specifies the presentation style for annotation text.

EXPRESS specification:

```
*)
ENTITY text_style;
  name                : label;
  character_appearance : character_style_select;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **text_style** is referred to.

character_appearance: the character style to be used for presenting the text.

6.9.2 character_glyph_style_stroke

A **character_glyph_style_stroke** is a character glyph style for text that is made up of curves rather than closed regions.

EXPRESS specification:

```
*)
ENTITY character_glyph_style_stroke;
  stroke_style : curve_style;
END_ENTITY;
(*
```

Attribute definitions:

stroke_style: the **curve_style** applied to the curves which define a **character_glyph_symbol_stroke**.

6.9.3 character_glyph_style_outline

A **character_glyph_style_outline** is a character glyph style for text that is made up of closed regions rather than curves.

EXPRESS specification:

```
*)
ENTITY character_glyph_style_outline;
  outline_style : curve_style;
END_ENTITY;
(*
```

Attribute definitions:

outline_style: the **curve_style** applied to the curves which define a **character_glyph_symbol_outline**.

6.9.4 character_glyph_style_outline_with_characteristics

A **character_glyph_style_outline_with_characteristics** is a **character_glyph_style_outline** with additional characteristics to be applied to the regions of the **character_glyph_style_outline**.

EXPRESS specification:

```
*)
ENTITY character_glyph_style_outline_with_characteristics
  SUBTYPE OF (character_glyph_style_outline);
  characteristics : fill_area_style;
END_ENTITY;
(*
```

Attribute definitions:

characteristics: the characteristics about the regions of the character glyph.

6.9.5 text_style_for_defined_font

A **text_style_for_defined_font** is a character glyph style for pre-defined or externally defined text fonts.

EXPRESS specification:

```
*)
ENTITY text_style_for_defined_font;
  text_colour : colour;
END_ENTITY;
(*
```

Attribute definitions:

text_colour: the colour to be used for presenting the text.

6.9.6 text_style_with_justification

A **text_style_with_justification** is a **text_style** that specifies the justification of text.

EXPRESS specification:

```
*)
ENTITY text_style_with_justification
  SUBTYPE OF (text_style);
```

```
  justification : text_justification;
END_ENTITY;
(*
```

Attribute definitions:

justification: the method of text justification in a line.

6.9.7 text_style_with_box_characteristics

A **text_style_with_box_characteristics** is a **text_style** that specifies the characteristics of the character boxes within the text.

EXPRESS specification:

```
*)
ENTITY text_style_with_box_characteristics
  SUBTYPE OF (text_style);
  characteristics : SET [1:4] OF box_characteristic_select;
WHERE
  WR1: SIZEOF( QUERY( c1 <* SELF.characteristics |
         SIZEOF( QUERY( c2 <* SELF.characteristics - c1 |
           TYPEOF (c1) = TYPEOF (c2)
           )) > 0
       )) = 0;
END_ENTITY;
(*
```

Attribute definitions:

characteristics: the characteristics of the character boxes. These characteristics determine height, width, rotation angle, and slant angle of the character boxes.

Formal propositions:

WR1: The characteristics shall be of different types.

6.9.8 text_style_with_spacing

A **text_style_with_spacing** is a **text_style** that has a spacing defined for the characters.

EXPRESS specification:

```
*)
ENTITY text_style_with_spacing
  SUBTYPE OF (text_style);
  character_spacing : character_spacing_select;
END_ENTITY;
(*
```

Attribute definitions:

character_spacing: the distance between the character boxes of adjacent characters.

6.9.9 pre_defined_character_spacing

A **pre_defined_character_spacing** is a character spacing for the definition of an application–specific character spacing.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_character_spacing
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

6.9.10 text_style_with_mirror

A **text_style_with_mirror** is a **text_style** that has a mirroring axis defined for the characters.

NOTE – Figure 24 shows the definition of **text_style_with_mirror**.

EXPRESS specification:

```
*)
ENTITY text_style_with_mirror
  SUBTYPE OF (text_style);
  mirror_placement : axis2_placement;
END_ENTITY;
(*
```

Attribute definitions:

mirror_placement: the placement and orientation of the mirror axis. The mirror axis passes through the **location** of the **axis2_placement** and is parallel to the x axis defined by the **axis2_placement**.

6.10 Presentation appearance schema entity definitions: presentation styles for symbols

6.10.1 symbol_style

A **symbol_style** is the **presentation_style** that specifies the visual appearance of **annotation_symbol**s. The style is specified as one or more styles for the constituents of the symbol, or as a colour to be used in presenting the entire symbol.

EXPRESS specification:

```
*)
ENTITY symbol_style;
  name            : label;
  style_of_symbol : symbol_style_select;
END_ENTITY;
(*
```

Attribute definitions:

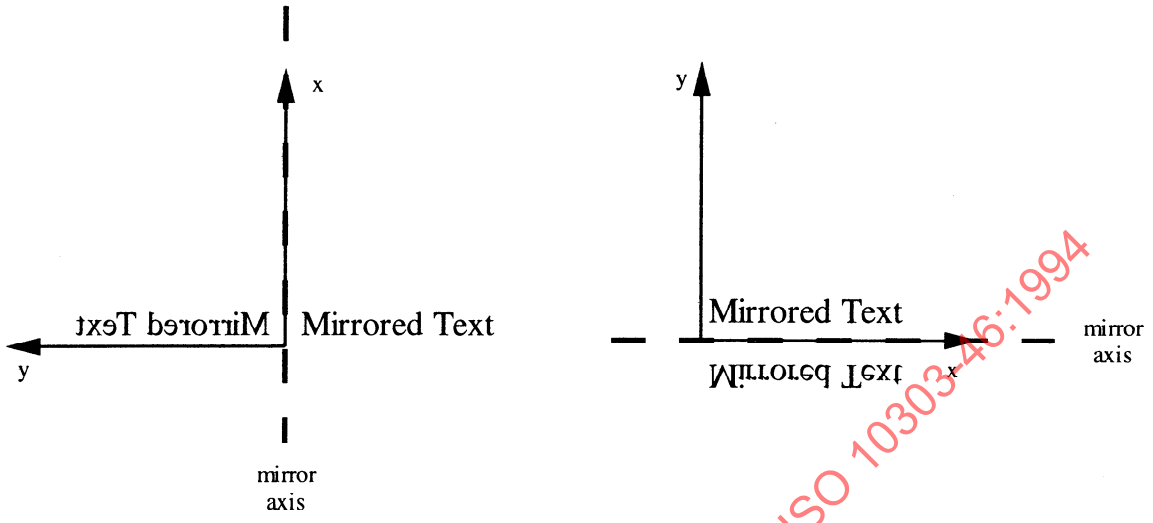


Figure 24 – Text style with mirror

name: the word, or group of words, by which the **symbol_style** is referred to.

style_of_symbol: the **symbol_element_style** or **symbol_colour** that specifies the visual appearance of a symbol.

6.10.2 symbol_element_style

A **symbol_element_style** is the presentation style that specifies the visual appearance of the constituent parts of **annotation_symbols**.

EXPRESS specification:

```
*)
ENTITY symbol_element_style;
  style_of_symbol : presentation_style_assignment;
WHERE
  WR1: SIZEOF (QUERY ( style <* SELF.style_of_symbol.styles |
         'PRESENTATION_APPEARANCE_SCHEMA.SYMBOL_STYLE' IN
         TYPEOF (style))) = 0;
  WR2: NOT ('PRESENTATION_APPEARANCE_SCHEMA.' +
            'PRESENTATION_STYLE_BY_CONTEXT' IN
            TYPEOF (SELF.style_of_symbol)
           );
END_ENTITY;
(*
```

Attribute definitions:

style_of_symbol: the **presentation_style_assignment** for the constituent parts of a symbol.

Formal propositions:

WR1: There shall be no **symbol_style** in the **style_of_symbol**.

WR2: The **style_of_symbol** shall not be dependent on a context.

6.10.3 symbol_colour

A **symbol_colour** is the **presentation_style** that specifies the colour of **annotation_symbols**.

EXPRESS specification:

```
*)
ENTITY symbol_colour;
  colour_of_symbol : colour;
END_ENTITY;
(*
```

Attribute definitions:

colour_of_symbol: the **colour** for the symbol.

6.11 Presentation appearance schema entity definitions: approximation tolerances

6.11.1 approximation_tolerance

An **approximation_tolerance** serves the visualization requirements of every displayable element. It specifies the position and shape tolerance of the presented elements in the picture with respect to their mathematically exact projected position and shape.

EXPRESS specification:

```
*)
ENTITY approximation_tolerance;
  tolerance : tolerance_select;
END_ENTITY;
(*
```

Attribute definitions:

tolerance: the tolerances to be used for approximating curves and surfaces.

NOTE – If no **approximation_tolerance** is specified, the accuracy of rendering is implementation-dependent.

6.11.2 approximation_tolerance_deviation

An **approximation_tolerance_deviation** specifies a deviation measurement for the approximation of curves and surfaces. The deviation value can be specified in product-shape or presentation area space.

EXPRESS specification:

```
*)
ENTITY approximation_tolerance_deviation;
  tessellation_type : approximation_method;
  tolerances        : SET [1:2] OF tolerance_deviation_select;
  definition_space  : product_or_presentation_space;
WHERE
  WR1: (HIINDEX(SELF.tolerances) = 1)
                XOR
       (TYPEOF(SELF.tolerances[1]) <> TYPEOF(SELF.tolerances[2]));
END_ENTITY;
(*
```

Attribute definitions:

tessellation_type: the selected **approximation_method** which determines the kind of tessellation with which curves and surfaces are approximated by graphical primitives.

tolerances: the set of tolerances which specify the maximum allowable deviation for the approximation of curves and surfaces.

definition_space: the coordinate space in which the tolerances are specified. The tolerances can be specified in the definition space of the curve or surface to which the tolerances are applied or in the definition space of the **presentation_area** which contains the curve or surface.

Formal propositions:

WR1: If there are two members of the tolerances set, they shall not be of the same type.

6.11.3 approximation_tolerance_parameter

An **approximation_tolerance_parameter** specifies an approximation tolerance for curves and surfaces in parameter space units.

EXPRESS specification:

```
*)
ENTITY approximation_tolerance_parameter;
  tolerances : SET [1:2] OF tolerance_parameter_select;
WHERE
  WR1: (HIINDEX (SELF.tolerances) = 1 )
               XOR
       (TYPEOF (SELF.tolerances[1]) <> TYPEOF (SELF.tolerances[2]));
END_ENTITY;
(*
```

Attribute definitions:

tolerances: the set of tolerances used for approximating curves and surfaces. Curves and surfaces are approximated in such a way that uniform steps in parameter space are taken as the basis of the approximation. The specified lengths are measured in parameter space units.

Formal propositions:

WR1: If there are two members of the tolerances set, they shall not be of the same type.

6.12 Presentation appearance schema entity definitions: occlusion and visibility

6.12.1 occlusion_precedence

An **occlusion_precedence** is a relationship between two entities that can hide or blank out other entities. This relationship establishes which one is to hide or blank out the other if they should overlap in a presentation.

This relationship is transitive. If entity A hides entity B, and entity B hides entity C, then entity A also hides entity C.

This relationship only applies if the two entities are in the same representation.

NOTE – If two such entities overlap and do not participate in an **occlusion_precedence** relationship, which entity has precedence is up to the particular implementation which presents it.

EXPRESS specification:

```
*)
ENTITY occlusion_precedence;
  higher_precedence : hiding_or_blanking_select;
  lower_precedence  : hiding_or_blanking_select;
  occlusion_context : representation;
WHERE
  WR1: acyclic_occlusion_precedence (SELF, [SELF.lower_precedence]);
END_ENTITY;
(*
```

Attribute definitions:

higher_precedence: the entity which can blank or hide the **lower_precedence** entity.

lower_precedence: the entity which can be blanked or hidden by the **higher_precedence** entity.

occlusion_context: the **representation** in which the precedence has meaning.

Formal propositions:

WR1: An **occlusion_precedence** entity shall not participate in a tree of **hiding_or_blanking_select** entities where the root of the tree is also a leaf of its own tree.

6.12.2 invisibility

invisibility specifies that a collection of one or more **styled_items**, elements assigned to a layer by **presentation_style_assignment**, or elements of a **presentation_representation** shall not be presented.

EXPRESS specification:

```
*)
ENTITY invisibility;
  invisible_items : SET [1:?] OF invisible_item;
END_ENTITY;
```

```
(*
```

Attribute definitions:

items: a set of **styled_items**, **presentation_layer_assignments**, or **presentation_representations** that are denoted as being invisible.

6.12.3 context_dependent_invisibility

A **context_dependent_invisibility** is an **invisibility** that is applied in the context of a picture or a layer. The elements specified as invisible are invisible only in the context of the identified **presentation_set**, **presentation_representation**, or **presentation_layer_usage**.

EXAMPLE 13 – a symbol is included as an item in two separate views, but is to be presented in only the first view. **context_dependent_invisibility** is used to specify that the symbol is not visibible in the second view, by specifying this view as the context for the invisibility.

EXPRESS specification:

```
*)
ENTITY context_dependent_invisibility
  SUBTYPE OF (invisibility);
  presentation_context : invisibility_context;
END_ENTITY;
(*
```

Attribute definitions:

presentation_context: the **presentation_set**, **presentation_representation**, or **presentation_layer_usage** that provides the context for the specification of invisibility.

6.13 Presentation_appearance schema function definitions

6.13.1 acyclic_occlusion_precedence

The **acyclic_occlusion_precedence** function is a function which checks to see if there is any tree containing a set of **hiding_or_blanking_select** and a given **occlusion_precedence** which has a **hiding_or_blanking_select** which is both a root and a leaf of the same subtree. It returns TRUE if there is no such subtree, and FALSE if there is.

EXPRESS specification:

```
*)
FUNCTION acyclic_occlusion_precedence
   ( relation : occlusion_precedence;
     set_of_lower : SET OF hiding_or_blanking_select ) : BOOLEAN;
   LOCAL
      x : SET OF occlusion_precedence;
      i : INTEGER;
      local_set_of_lower : SET OF hiding_or_blanking_select;
   END_LOCAL;
   REPEAT i:=1 TO HIINDEX(set_of_lower);
      IF relation.higher_precedence :=: set_of_lower[i] THEN
         RETURN(FALSE);
```

```
      END_IF;
   END_REPEAT;
   x := USEDIN ( relation.higher_precedence,
                    'PRESENTATION_APPEARANCE_SCHEMA.'+
            'OCCLUSION_PRECEDENCE.LOWER_PRECEDENCE');
   local_set_of_lower := set_of_lower + relation.higher_precedence;
   IF SIZEOF (x) > 0 THEN
      REPEAT i:=1 TO HIINDEX (x);
         If NOT acyclic_occlusion_precedence(x[i] ,

                                        local_set_of_lower) THEN
            RETURN (FALSE);
         END_IF;
      END_REPEAT;
   END_IF;
   RETURN (TRUE);
END_FUNCTION;
(*
```

Argument definitions:

relation: the **occlusion_precedence** which is tested. This is input to the function.

set_of_lower: the set of items that are referenced directly or indirectly by **lower_precedence** of the **relation**. This argument is input to the function. On initial input this set contains as its only element the **lower_precedence** item of the **relation**.

EXPRESS specification:

```
*)
END_SCHEMA; -- presentation_appearance_schema
(*
```

7 Presentation resource schema

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

EXPRESS specification:

```
*)
SCHEMA presentation_resource_schema;

REFERENCE FROM external_reference_schema
    (externally_defined_item,
     pre_defined_item);

REFERENCE FROM geometry_schema
    (axis2_placement,
     curve,
     geometric_representation_item
     );

REFERENCE FROM measure_schema
    (length_measure,
     positive_length_measure,
     positive_ratio_measure,
     ratio_measure);

REFERENCE FROM presentation_definition_schema
    (annotation_fill_area,
     symbol_representation);

REFERENCE FROM representation_schema
    (item_in_context,
     representation);

REFERENCE FROM support_resource_schema
    (identifier,
     label,
     text);

(*
```

NOTES

1 – The schemas referenced above can be found in the following parts of ISO 10303:

external_reference_schema	ISO 10303-41
geometry_schema	ISO 10303-42
management_resources_schema	ISO 10303-41
measure_schema	ISO 10303-41

presentation_definition_schema	Clause 5 of this part of ISO 10303
representation_schema	ISO 10303-43
support_resource_schema	ISO 10303-41

2 – The EXPRESS–G diagrams for this schema may be found in Annex E of this part of ISO 10303.

7.1 Introduction

The subject of the presentation_resource_schema is the specification of basic resources for presentation. There are three types of information specified in the presentation_resource_schema:

- text font resources;
- colour definition resources;
- geometric resources.

This schema specifies the resources necessary for construction of character fonts and annotation symbol fonts. The character fonts and annotation symbol fonts are defined within a local coordinate system. The characters and annotation symbols may be scaled and transformed depending on application usage.

There are two types of colour definition resources. The first is a direct colour specification based on the RGB colour model. The second makes use of a colour mapping table to associate a colour with a state variable along a continuous scale.

Geometric resources are geometric elements used in this part of ISO 10303 to support miscellaneous aspects of picture construction.

7.2 Presentation resource schema type definitions

7.2.1 staircase_or_linear

The **staircase_or_linear** type specifies the interpolation method for colours in a **colour_association_table**.

EXPRESS specification:

```
*)
TYPE staircase_or_linear = ENUMERATION OF
  (staircase,
   linear);
END_TYPE;
(*
```

Enumerated item definitions:

staircase: The colours are interpolated using a staircase function.

linear: The colours are interpolated linearly.

7.2.2 presentable_text

A **presentable_text** is any string which can be presented.

EXPRESS specification:

```
*)
TYPE presentable_text = STRING;
END_TYPE;
(*
```

Informal propositions:

IP1: The string shall not contain any control characters.

EXAMPLE 14 – IP1 prohibits linefeed and carriage return characters in **presentable_text**.

7.2.3 font_select

The **font_select** is used for the definition of **text_literals** and **character_glyph_symbols**. It selects between different sources for text fonts.

EXPRESS specification:

```
*)
TYPE font_select = SELECT
  (pre_defined_text_font,
   externally_defined_text_font);
END_TYPE;
(*
```

7.3 Presentation resource schema entity definitions

7.3.1 character_glyph_symbol

A **character_glyph_symbol** contains the geometric representation of a character.

EXPRESS specification:

```
*)
ENTITY character_glyph_symbol
  SUBTYPE OF (symbol_representation);
  character_box  : planar_extent;
  baseline_ratio : ratio_measure;
DERIVE
  box_height : length_measure := character_box.size_in_y;
WHERE
  WR1: {0.0 <= baseline_ratio <= 1.0};
  WR2: item_in_context(SELF.character_box,
                       SELF\representation.context_of_items);
  WR3: 'MEASURE_SCHEMA.POSITIVE_LENGTH_MEASURE'
       IN TYPEOF (SELF.box_height);
END_ENTITY;
(*
```

Attribute definitions:

character_box: a rectangular box defining the extent of a character glyph. The **character_box** can be slanted by the **text_style** to produce slanted character glyphs.

baseline_ratio: the location of the baseline of the character glyph relative to the **character_box**. When character glyphs are composed to form a text literal, the baselines of adjacent glyphs are aligned. The character baseline is parallel to the x axis of the **character_box**. The specified value indicates the distance between the x axis of the **character_box** and the baseline as ratio of **box_height**.

NOTE – x axis and extent of the character box are defined by **planar_extent**, clause 7.3.17.

EXAMPLE 15 – A **baseline_ratio** of 0.0 specifies that the baseline of the character glyph and x axis are identical. A **baseline_ratio** of 0.5 specifies that the baseline of the character glyph divides the **character_box** in the middle of its height.

box_height: the height of the **character_box**.

Formal propositions:

WR1: The **baseline_ratio** ranges between 0.0 and 1.0.

WR2: The **character_box** shall be in the context of the **text_symbol**.

WR2: The **box_height** shall be a **positive_length_measure**.

NOTE – Figure 25 illustrates the types and definition of character glyph symbols.

7.3.2 character_glyph_symbol_stroke

A **character_glyph_symbol_stroke** is a **character_glyph_symbol** where the geometry of the glyph is described by a set of curves.

EXPRESS specification:

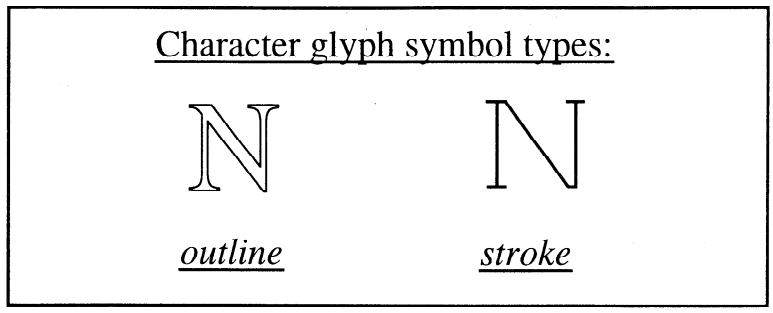
```
*)
ENTITY character_glyph_symbol_stroke
  SUBTYPE OF (character_glyph_symbol);
  strokes : SET [1:?] OF curve;
WHERE
  WR1: SELF.strokes <= SELF\representation.items;
END_ENTITY;
(*
```

Attribute definitions:

strokes: the set of **curves** that define the geometry of the character glyph.

Formal propositions:

WR1: All the curves making up the character glyph shall be contained in the set of items.



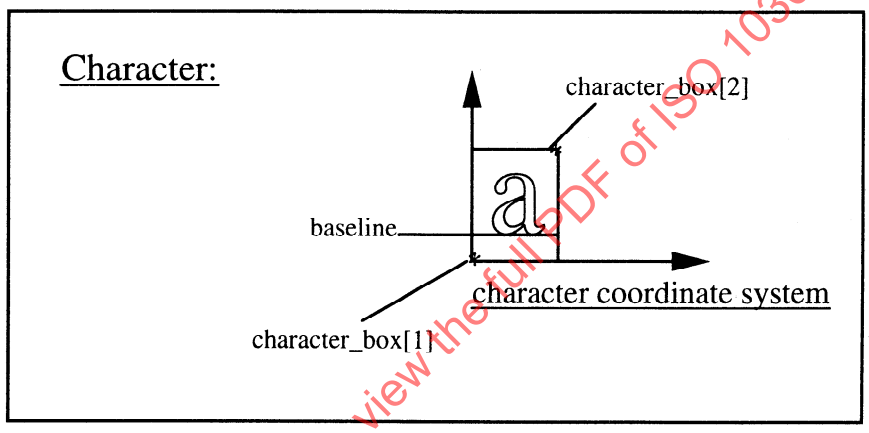


Figure 25 – Character glyph symbols

7.3.3 character_glyph_symbol_outline

A **character_glyph_symbol_outline** is a **character_glyph_symbol** where the geometry of the glyph is described by a set of **annotation_fill_area**s.

EXPRESS specification:

```
*)
ENTITY character_glyph_symbol_outline
  SUBTYPE OF (character_glyph_symbol);
  outlines : SET [1:?] OF annotation_fill_area;
WHERE
  WR1: SELF.outlines <= SELF\representation.items;
END_ENTITY;
(*
```

Attribute definitions:

outlines: the set of **annotation_fill_area**s that define the geometry of the character glyph.

Formal propositions:

WR1: All the fill areas making up the character glyph shall be contained in the set of items.

7.3.4 character_glyph_font_usage

A **character_glyph_font_usage** specifies the participation of a **character_glyph_symbol** in a **text_font**.

EXPRESS specification:

```
*)
ENTITY character_glyph_font_usage;
  character : character_glyph_symbol;
  font      : text_font;
END_ENTITY;
(*
```

Attribute definitions:

character: the **character_glyph_symbol** that is part of the font.

font: the **text_font** to which the **character_glyph_symbol** is assigned.

7.3.5 text_font

A **text_font** is the identification of a specific character font.

EXPRESS specification:

```
*)
ENTITY text_font;
  id          : identifier;
  name        : label;
  description : text;
INVERSE
  glyphs : SET [1:?] OF character_glyph_font_usage FOR font;
END_ENTITY;
(*
```

Attribute definitions:

id: the identification of the **text_font**.

name: the word, or group of words, by which the **text_font** is referred to.

description: text that relates the nature of the **text_font**.

glyphs: the set of **character_glyph_font_usage**s that identify the **character_glyph_symbol**s that participate in the **text_font**.

EXAMPLE 16 – Examples for character fonts are Courier 12, Times 10, Helvetica Bold 14, 0815, ABC.

7.3.6 text_font_family

A **text_font_family** is the identification of a related collection of **text_font**s.

EXPRESS specification:

```
*)
ENTITY text_font_family;
  id          : identifier;
  name        : label;
  description : text;
INVERSE
  fonts       : SET [1:?] OF text_font_in_family FOR family;
END_ENTITY;
(*
```

Attribute definitions:

id: the identification of the **text_font_family**.

name: the word, or group of words, by which the **text_font_family** is referred to.

description: text that relates the nature of the **text_font_family**.

fonts: the set of **text_font_in_family**s that identify the **text_font**s that participate in the **text_font_family**.

7.3.7 text_font_in_family

A **text_font_in_family** specifies the participation of a **text_font** in a **text_font_family**.

EXPRESS specification:

```
*)
ENTITY text_font_in_family;
  font   : text_font;
  family : text_font_family;
END_ENTITY;
(*
```

Attribute definitions:

font: the **text_font** that is part of the family.

family: the **text_font_family** to which the **text_font** is assigned.

EXAMPLE 17 – Some font families are Courier and Helvetica. Fonts in these families include Courier 12, Helvetica Bold 14, respectively.

7.3.8 externally_defined_text_font

An **externally_defined_text_font** is an external reference to a text font.

EXPRESS specification:

```
*)
ENTITY externally_defined_text_font
  SUBTYPE OF (externally_defined_item);
END_ENTITY;
(*
```

7.3.9 pre_defined_text_font

A **pre_defined_text_font** is a text font provided for the definition of an application–specific font.

NOTE – Application Resources or Application Protocols specify the use of this entity.

EXPRESS specification:

```
*)
ENTITY pre_defined_text_font
  SUBTYPE OF (pre_defined_item);
END_ENTITY;
(*
```

7.3.10 colour

A colour defines a basic appearance property of an element with respect to the light reflected by it.

EXPRESS specification:

```
*)
ENTITY colour;
END_ENTITY;
(*
```

7.3.11 colour_specification

A **colour_specification** contains a colour definition which refers directly to a specific colour space.

EXAMPLE 18 – Colour spaces are RGB, HLS, HSV, and CIE. More details about these colour spaces can be found in [12].

NOTE – Only RGB colours are supported by this part of ISO 10303. This entity provides for later extension.

EXPRESS specification:

```
*)
ENTITY colour_specification
  SUBTYPE OF (colour);
  name : label;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **colour_specification** is referred to.

7.3.12 colour_rgb

A colour_rgb defines a colour by specifying the intensity of red, green, and blue.

EXPRESS specification:

```
*)
ENTITY colour_rgb
  SUBTYPE OF (colour_specification);
  red   : REAL;
  green : REAL;
  blue  : REAL;
WHERE
  WR1: {0.0 <= red <= 1.0};
  WR2: {0.0 <= green <= 1.0};
  WR3: {0.0 <= blue <= 1.0};
END_ENTITY;
(*
```

Attribute definitions:

red: the intensity of the red colour component.

green: the intensity of the green colour component.

blue: the intensity of the blue colour component.

Formal propositions:

WR1: The intensity of the red colour component shall be between 0.0 and 1.0.

WR2: The intensity of the green component shall be between 0.0 and 1.0.

WR3: The intensity of the blue component shall be between 0.0 and 1.0.

7.3.13 colour_associated

The **colour_associated** entity defines a colour for the visualization of one-dimensional state variables to achieve a rendered picture. The colour at specific positions of a curve or surface is derived from the values of the state variables and the **colour_association_table**. The positions for which the colours are derived depend on the **shading_curve_method** or the **shading_surface_method** assigned to the curve or surface, respectively. The derived colours are interpolated according to the shading method appearance attribute.

NOTES

1 – State variables are often used to specify physical quantities.

2 – The interpolation of colours across curves and surfaces is described in 6.3.17 and 6.3.21.

EXPRESS specification:

```
*)
ENTITY colour_associated
  SUBTYPE OF (colour);
  name : colour;
  variable_to_be_shown : SET [1:?] OF REAL;
  mapping              : colour_association_table;
END_ENTITY;
(*
```

Attribute definitions:

name: the word, or group of words, by which the **colour_asscociated** is referred to.

variable_to_be_shown: the set of state variables are the one–dimensional values to be visualized by colours.

mapping: the **colour_association_table** used to derive the colour.

7.3.14 colour_association_table

A **colour_association_table** defines the mapping of a one–dimensional state variable space into a colour space.

The **colour_association_table** contains state variable values that are associated with a colour. When the value of a state variable is between two fixed state variable values, the colour shall be calculated by either staircase or linear interpolation.

EXPRESS specification:

```
*)
ENTITY colour_association_table;
  discrete_states_with_colours : LIST [1:?] OF state_variable_with_colour;
  interpolation_type           : staircase_or_linear;
END_ENTITY;
(*
```

Attribute definitions:

discrete_states_with_colours: a list of one–dimensional state variable values that are associated with colour.

interpolation_type: the method of interpolation to be used.

7.3.15 state_variable_with_colour

A **state_variable_with_colour** associates a single fixed state variable with a **colour_specification**.

EXPRESS specification:

```
*)
ENTITY state_variable_with_colour;
  state_variable    : REAL;
  associated_colour : colour_specification;
END_ENTITY;
(*
```

Attribute definitions:

state_variable: the value of a one–dimensional **state_variable**.

EXAMPLE 19 Physical state variables are temperature and stress components.

associated_colour: the **colour_specification** associated with the **state_variable**.

7.3.16 pre_defined_colour

A **pre_defined_colour** is provided to allow an application–specific colour definition.

NOTE – Application Resources or Application Protocols specify the use of this entity. The **pre_defined_colour** entity further enables Application Resources or Application Protocols to fix colour values or components of colour values for their particular uses.

EXPRESS specification:

```
*)
ENTITY pre_defined_colour
  SUBTYPE OF (pre_defined_item, colour);
END_ENTITY;
(*
```

7.3.17 planar_extent

A **planar_extent** specifies an extent in both directions of a two–dimensional coordinate system.

EXPRESS specification:

```
*)
ENTITY planar_extent
  SUBTYPE OF (geometric_representation_item);
   size_in_x : length_measure;
   size_in_y : length_measure;
END_ENTITY;
(*
```

Attribute definitions:

size_in_x: the extent in the x axis direction.

size_in_y: the extent in the y axis direction.

7.3.18 planar_box

A **planar_box** specifies an arbitrary rectangular box and its location in a two–dimensional cartesian coordinate system.

EXPRESS specification:

```
*)
ENTITY planar_box
  SUBTYPE OF (planar_extent);
  placement:  axis2_placement;
END_ENTITY;
(*
```

Attribute definitions:

placement: the position and orientation of the bottom–left corner of the box. The attributes of the supertype define the length of the sides of the box along the positive x and y axes.

7.3.19 presentation_scaled_placement

The **presentation_scaled_placement** is a **geometric_representation_item** provided to support the definition of **graphical_transformations**.

NOTE – **graphical_transformation** is defined in 4.4.12.

EXPRESS specification:

```
*)
ENTITY presentation_scaled_placement
  SUBTYPE OF (geometric_representation_item);
  placement : axis2_placement;
  scaling   : positive_ratio_measure;
END_ENTITY;
(*
```

Attribute definitions:

placement: a placement used to define rotation and translation.

scaling: the scaling that is part of the transformation.

EXPRESS specification:

```
*)
END_SCHEMA; -- presentation_resource_schema
(*
```

Annex A
(normative)

Short names of entities

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

Table A.1 – Short names of entities

Entity names	Short names
ANNOTATION_CURVE_OCCURRENCE	ANCROC
ANNOTATION_FILL_AREA	ANFLAR
ANNOTATION_FILL_AREA_OCCURRENCE	AFAO
ANNOTATION_OCCURRENCE	ANNOCC
ANNOTATION_OCCURRENCE_RELATIONSHIP	ANOCRL
ANNOTATION_POINT_OCCURRENCE	ANPNOC
ANNOTATION_SYMBOL	ANNSYM
ANNOTATION_SYMBOL_OCCURRENCE	ANSYOC
ANNOTATION_TABLE	ANNTBL
ANNOTATION_TABLE_OCCURRENCE	ANTBOC
ANNOTATION_TEXT	ANNTXT
ANNOTATION_TEXT_CHARACTER	ANTXCH
ANNOTATION_TEXT_OCCURRENCE	ANTXOC
ANNOTATION_TEXT_WITH_ASSOCIATED_CURVES	ATWAC
ANNOTATION_TEXT_WITH_BLANKING_BOX	ATWBB
ANNOTATION_TEXT_WITH_DELINEATION	ATWD
ANNOTATION_TEXT_WITH_EXTENT	ATWE
APPROXIMATION_TOLERANCE	APPTLR
APPROXIMATION_TOLERANCE_DEVIATION	APTLDV
APPROXIMATION_TOLERANCE_PARAMETER	APTLPR

Table A.1 (continued)

Entity names	Short names
AREA_DEPENDENT_ANNOTATION_REPRESENTATION	ADAR
AREA_IN_SET	ARINST
BACKGROUND_COLOUR	BCKGLR
CAMERA_IMAGE	CMRIMG
CAMERA_MODEL	CMRMDL
CAMERA_MODEL_D2	CMMDD2
CAMERA_MODEL_D2_SHAPE_CLIPPING	CMDSC
CAMERA_MODEL_D3	CMMDD3
CAMERA_MODEL_D3_MULTI_CLIPPING	CMDMC
CAMERA_MODEL_D3_WITH_HLHSR	CMDWH
CAMERA_MODEL_WITH_LIGHT_SOURCES	CMWLS
CAMERA_USAGE	CMRUSG
CHARACTER_GLYPH_FONT_USAGE	CGFU
CHARACTER_GLYPH_STYLE_OUTLINE	CGO
CHARACTER_GLYPH_STYLE_OUTLINE_WITH_CHARACTERISTICS	CGSOWC
CHARACTER_GLYPH_STYLE_STROKE	CGS
CHARACTER_GLYPH_SYMBOL	CHGLSY
CHARACTER_GLYPH_SYMBOL_OUTLINE	CGSO
CHARACTER_GLYPH_SYMBOL_STROKE	CGSS
COLOUR	COLOUR
COLOUR_ASSOCIATED	CLRASS
COLOUR_ASSOCIATION_TABLE	CLASTB
COLOUR_RGB	CLRRGB
COLOUR_SPECIFICATION	CLRSPC
COMPOSITE_TEXT	CMPTXT
COMPOSITE_TEXT_WITH_ASSOCIATED_CURVES	CTWAC
COMPOSITE_TEXT_WITH_BLANKING_BOX	CTWBB

Table A.1 (continued)

Entity names	Short names
COMPOSITE_TEXT_WITH_DELINEATION	CTWD
COMPOSITE_TEXT_WITH_EXTENT	CTWE
CONTEXT_DEPENDENT_INVISIBILITY	CNDPIN
CONTEXT_DEPENDENT_OVER_RIDING_STYLED_ITEM	CDORSI
CURVE_STYLE	CRVSTY
CURVE_STYLE_CURVE_PATTERN	CSCP
CURVE_STYLE_CURVE_PATTERN_SET	CSCPS
CURVE_STYLE_FONT	CRSTFN
CURVE_STYLE_FONT_AND_SCALING	CSFAS
CURVE_STYLE_FONT_PATTERN	CSFP
CURVE_STYLE_RENDERING	CRSTRN
CURVE_STYLE_WIDE	CRSTWD
CURVE_STYLE_WITH_ENDS_AND_CORNERS	CSWEAC
CURVE_STYLE_WITH_EXTENSION	CSWE
DEFINED_CHARACTER_GLYPH	DFCHGL
DEFINED_SYMBOL	DFNSYM
DEFINED_TABLE	DFNTBL
EXTERNALLY_DEFINED_CHARACTER_GLYPH	EDCG
EXTERNALLY_DEFINED_CURVE_FONT	EDCF
EXTERNALLY_DEFINED_HATCH_STYLE	EDHS
EXTERNALLY_DEFINED_STYLE	EXDFST
EXTERNALLY_DEFINED_SYMBOL	EXDFSY
EXTERNALLY_DEFINED_TEXT_FONT	EDTF
EXTERNALLY_DEFINED_TILE	EXDFTL
EXTERNALLY_DEFINED_TILE_STYLE	EDTS
FILL_AREA_STYLE	FLARST
FILL_AREA_STYLE_COLOUR	FASC

Table A.1 (continued)

Entity names	Short names
FILL_AREA_STYLE_HATCHING	FASH
FILL_AREA_STYLE_TILES	FAST
FILL_AREA_STYLE_TILE_COLOURED_REGION	FASTCR
FILL_AREA_STYLE_TILE_CURVE_WITH_STYLE	FASTCW
FILL_AREA_STYLE_TILE_SYMBOL_WITH_STYLE	FASTSW
GRAPHICAL_TRANSFORMATION	GRPTRN
INVISIBILITY	INVSBL
LIGHT_SOURCE	LGHSRC
LIGHT_SOURCE_AMBIENT	LGSRAM
LIGHT_SOURCE_DIRECTIONAL	LGSRDR
LIGHT_SOURCE_POSITIONAL	LGSRPS
LIGHT_SOURCE_SPOT	LGSRSP
OCCLUSION_PRECEDENCE	OCCPRC
ONE_DIRECTION_REPEAT_FACTOR	ODRF
OVER_RIDING_STYLED_ITEM	ORSI
PLANAR_BOX	PLNBX
PLANAR_EXTENT	PLNEXT
POINT_STYLE	PNTSTY
PRESENTATION_AREA	PRSAR
PRESENTATION_LAYER_ASSIGNMENT	PRLYAS
PRESENTATION_LAYER_USAGE	PRLYUS
PRESENTATION_REPRESENTATION	PRSRPR
PRESENTATION_REPRESENTATION_RELATIONSHIP	PRRPRL
PRESENTATION_SCALED_PLACEMENT	PRSCPL
PRESENTATION_SET	PRSST
PRESENTATION_SIZE	PRSSZ
PRESENTATION_STYLE_ASSIGNMENT	PRSTAS

Table A.1 (continued)

Entity names	Short names
PRESENTATION_STYLE_BY_CONTEXT	PSBC
PRESENTATION_VIEW	PRSVW
PRESENTED_ITEM	PRSITM
PRESENTED_ITEM_REPRESENTATION	PRITRP
PRE_DEFINED_CHARACTER_GLYPH	PDCG
PRE_DEFINED_CHARACTER_SPACING	PDCS
PRE_DEFINED_COLOUR	PRDFCL
PRE_DEFINED_CURVE_FONT	PDCF
PRE_DEFINED_HATCH_STYLE	PDHS
PRE_DEFINED_MARKER	PRDFMR
PRE_DEFINED_PRESENTATION_STYLE	PDPS
PRE_DEFINED_SIZE	PRDFSZ
PRE_DEFINED_SURFACE_SIDE_STYLE	PDSSS
PRE_DEFINED_SYMBOL	PRDFSY
PRE_DEFINED_TEXT_FONT	PDTF
PRE_DEFINED_TILE	PRDFTL
PRE_DEFINED_TILE_STYLE	PDTS
PRODUCT_DATA_REPRESENTATION_VIEW	PDRV
REPRESENTATION_ITEM_DEPENDENT_LAYER_ASSIGNMENT	RIDLA
STATE_VARIABLE_WITH_COLOUR	SVWC
STYLED_ITEM	STYITM
SURFACE_RENDERING_PROPERTIES	SRRNPR
SURFACE_SIDE_STYLE	SRSDST
SURFACE_STYLE_BOUNDARY	SRSTBN
SURFACE_STYLE_CONTROL_GRID	SSCG
SURFACE_STYLE_FILL_AREA	SSFA
SURFACE_STYLE_PARAMETER_LINE	SSPL

Table A.1 (continued)

Entity names	Short names
SURFACE_STYLE_REFLECTANCE_AMBIENT	SSRA
SURFACE_STYLE_REFLECTANCE_AMBIENT_DIFFUSE	SSRAD
SURFACE_STYLE_REFLECTANCE_AMBIENT_DIFFUSE_SPECULAR	SSRADS
SURFACE_STYLE_RENDERING	SRSTRN
SURFACE_STYLE_RENDERING_WITH_PROPERTIES	SSRWP
SURFACE_STYLE_SEGMENTATION_CURVE	SSSC
SURFACE_STYLE_SILHOUETTE	SRSTSL
SURFACE_STYLE_TRANSPARENT	SRSTTR
SURFACE_STYLE_USAGE	SRSTUS
SYMBOL_COLOUR	SYMCLR
SYMBOL_ELEMENT_STYLE	SYELST
SYMBOL_REPRESENTATION	SYMRPR
SYMBOL_REPRESENTATION_MAP	SYRPMP
SYMBOL_REPRESENTATION_RELATIONSHIP	SYRPRL
SYMBOL_REPRESENTATION_WITH_BLANKING_BOX	SRWBB
SYMBOL_STYLE	SYMSTY
SYMBOL_TARGET	SYMTRG
TABLE_RECORD_FIELD_REPRESENTATION	TRFR
TABLE_RECORD_FIELD_REPRESENTATION_WITH_CLIPPING_BOX	TRFRWC
TABLE_RECORD_REPRESENTATION	TBRCRP
TABLE_REPRESENTATION	TBLRPR
TABLE_REPRESENTATION_RELATIONSHIP	TBRPRL
TABLE_TEXT_RELATIONSHIP	TBTXRL
TEXT_FONT	TXTFNT
TEXT_FONT_FAMILY	TXFNFM
TEXT_FONT_IN_FAMILY	TFIF
TEXT_LITERAL	TXTLTR

Table A.1 (concluded)

Entity names	Short names
TEXT_LITERAL_WITH_ASSOCIATED_CURVES	TLWAC
TEXT_LITERAL_WITH_BLANKING_BOX	TLWBB
TEXT_LITERAL_WITH_DELINEATION	TLWD
TEXT_LITERAL_WITH_EXTENT	TLWE
TEXT_STRING_REPRESENTATION	TXSTRP
TEXT_STYLE	TXTSTY
TEXT_STYLE_FOR_DEFINED_FONT	TSFDF
TEXT_STYLE_WITH_BOX_CHARACTERISTICS	TSWBC
TEXT_STYLE_WITH_JUSTIFICATION	TSWJ
TEXT_STYLE_WITH_MIRROR	TSWM
TEXT_STYLE_WITH_SPACING	TSWS
TWO_DIRECTION_REPEAT_FACTOR	TDRF
VIEW_DEPENDENT_ANNOTATION_REPRESENTATION	VDAR
VIEW_VOLUME	VWVLM

Annex B
(normative)

Information object registration

B.1 Document identification

In order to provide for unambiguous identification of an information object in an open system, the object identifier

{ iso standard 10303 part(46) version(1) }

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO 8824-1, and is described in ISO 10303-1.

B.2 Schema identification

B.2.1 presentation_organisation_schema identification

In order to provide for unambiguous identification of the presentation_organisation_schema in an open system, the object identifier

{ iso standard 10303 part(46) version(1) object(1) presentation-organisation-schema(1) }

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

B.2.2 presentation_definition_schema identification

In order to provide for unambiguous identification of the presentation_definition_schema in an open system, the object identifier

{ iso standard 10303 part(46) version(1) object(1) presentation-definition-schema(2) }

is assigned to the presentation_definition_schema schema (see clause 5). The meaning of this value is defined in ISO 8824-1, and is described in ISO 10303-1.

B.2.3 presentation_appearance_schema identification

In order to provide for unambiguous identification of the presentation_appearance_schema in an open system, the object identifier

{ iso standard 10303 part(46) version(1) object(1) presentation-appearance-schema(3) }

is assigned to the presentation_appearance_schema schema (see clause 6). The meaning of this value is defined in ISO 8824-1, and is described in ISO 10303-1.

B.2.4 presentation_resource_schema identification

In order to provide for unambiguous identification of the presentation_resource_schema in an open system, the object identifier

{ iso standard 10303 part(46) version(1) object(1) presentation-resource-schema(4) }

is assigned to the presentation_resource_schema schema (see clause 7). The meaning of this value is defined in ISO 8824-1, and is described in ISO 10303-1.

Annex C
(informative)

Computer–interpretable listings

This annex provides a listing of the short names and a listing of the *EXPRESS* specified in this part of ISO 10303. No text or annotation is included. This annex is provided only in computer-interpretable form.

NOTE – The information provided on this diskette is informative; the normative text is that contained in the body of this part of ISO 10303.

Annex D
(informative)
Technical discussions

D.1 Symbols used in reflectance equations

Table D.1 is taken from ISO/IEC 9592–4, Annex E.

Table D.1 – PHIGS PLUS annex E : Variable definition and their sources

Symbol	Description	Data Type	Source
$\vec{L}_d$	light source definition	3×R	(1)
L_c	light source colour	COLRV	(1)
$\vec{L}_p$	light source position	3×R	(1)
L_e	light source concentration exponent	R	(1)
C_1, C_2	attenuation coefficients	R	(1)
A_s	spread angle	R	(1)
L_a	light attenuation	R	(2)
$\vec{O}_p$	object position	3×R	(3)
O_d	object diffuse colour	COLRV	(4)
O_s	object specular colour	COLRV	(5)
O_e	object specular exponent	R	(5)
K_a	ambient reflection coefficient	R	(5)
K_d	diffuse reflection coefficient	R	(5)
K_s	specular reflection coefficient	R	(5)
$\vec{V}_e$	unit vector from object to eye point	3×R	(2)
$\vec{V}_r$	unit reflection vector from object	3×R	(2)
$\vec{V}_l$	unit vector from object to light source	3×R	(2)
$\vec{V}_n$	unit normal vector to object	NORM	(3)
C_a	ambient contribution from light source	COLRV	(2)
C_d	diffuse contribution from light source	COLRV	(2)
C_s	specular contribution from light source	COLRV	(2)

Source legend:

1: light source representation
2: calculated
3: explicit or derived from object geometry
4: colour table, direct colour, vertex colour, back interior colour
5: surface or back area properties

D.2 Suggested reflectance equations

The following equations are taken from ISO/IEC 9592–4. Annex E:

The result of the reflectance equation evaluated at the point on a primitive is a single colour that is the sum of the individual components, $C_a + C_d + C_s$, of all the currently active light sources:

$\sum_{i=1}^{N}(C_{a_i} + C_{d_i} + C_{s_i})$; N = total number of the light sources

For ambient light sources

$$\begin{aligned} C_a &= K_a L_c O_d \\ C_d &= 0 \\ C_s &= 0 \end{aligned}$$

For directional light sources

$$\begin{aligned} C_a &= 0 \\ C_d &= K_d L_c O_d (\vec{V}_n \cdot \vec{V}_l) \\ C_s &= K_s O_s L_c (\vec{V}_e \cdot \vec{V}_r)^{O_e} \end{aligned}$$

For positional light sources

$$\begin{aligned} C_a &= 0 \\ C_d &= K_d O_d L_c (\vec{V}_n \cdot \vec{V}_l) L_a \\ C_s &= K_s O_s L_c (\vec{V}_e \cdot \vec{V}_r)^{O_e} L_a \end{aligned}$$

For one spot light source
(Contributions from one spot light sources will be zero if $\vec{V}_l$ is outside the cone of influence of the light source.)

$$\begin{aligned} C_a &= 0 \\ C_d &= K_d O_d L_c (\vec{V}_n \cdot \vec{V}_l)(\vec{L}_d \cdot (-\vec{V}_l))^{L_e} L_a \\ C_s &= K_s O_s L_c (\vec{V}_e \cdot \vec{V}_r)^{O_e} (\vec{L}_d \cdot (-\vec{V}_l))^{L_e} L_a \end{aligned}$$

Light attenuation may be calculated as follows:

$$L_a = \frac{1}{C_1 + C_2 \, ||\vec{O}_p - \vec{L}_p||}$$

Light reflection vector may be calculated as follows:

$$\vec{V}_r = 2(\vec{V}_n \cdot \vec{V}_l)\vec{V}_n - \vec{V}_l$$

In all cases, a dot product resulting in a negative value will be replaced by 0.

Annex E
(informative)

EXPRESS–G diagrams

Figures E.1 through E.42 correspond to the *EXPRESS* listing given in annex A. The figures use the *EXPRESS-G* graphical notation for the *EXPRESS* language. *EXPRESS-G* is defined in annex D of ISO 10303-11.

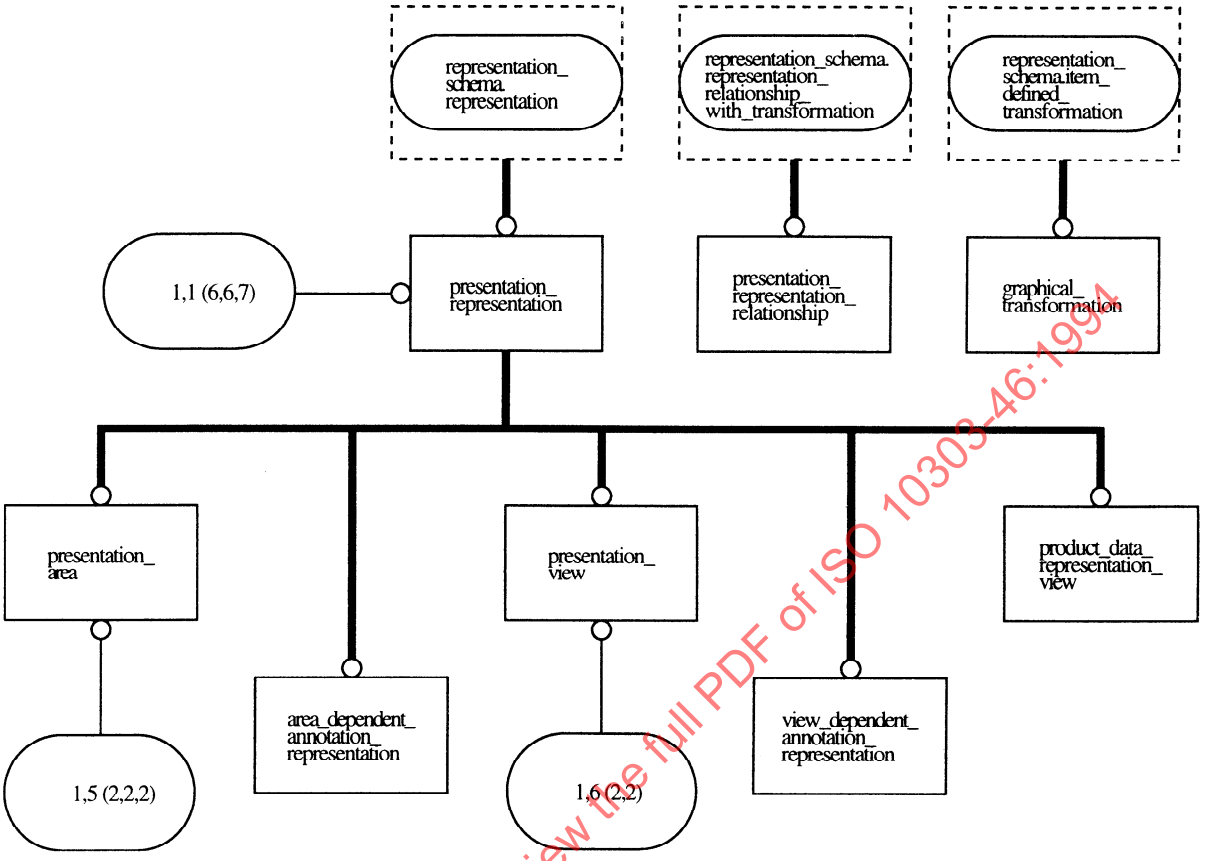


Figure E.1 – presentation_organisation_schema – EXPRESS–G diagram 1 of 7

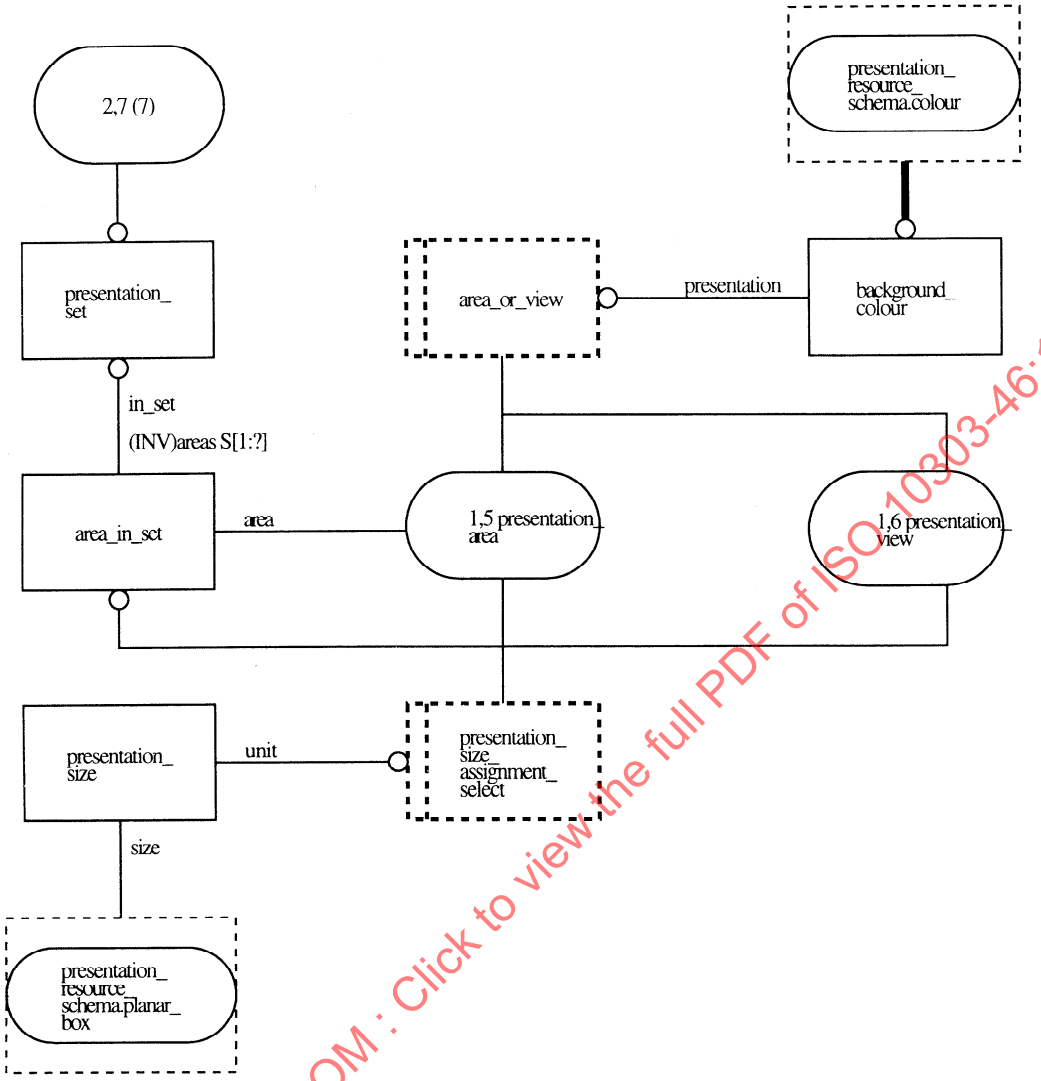


Figure E.2 – presentation_organisation_schema – EXPRESS–G diagram 2 of 7

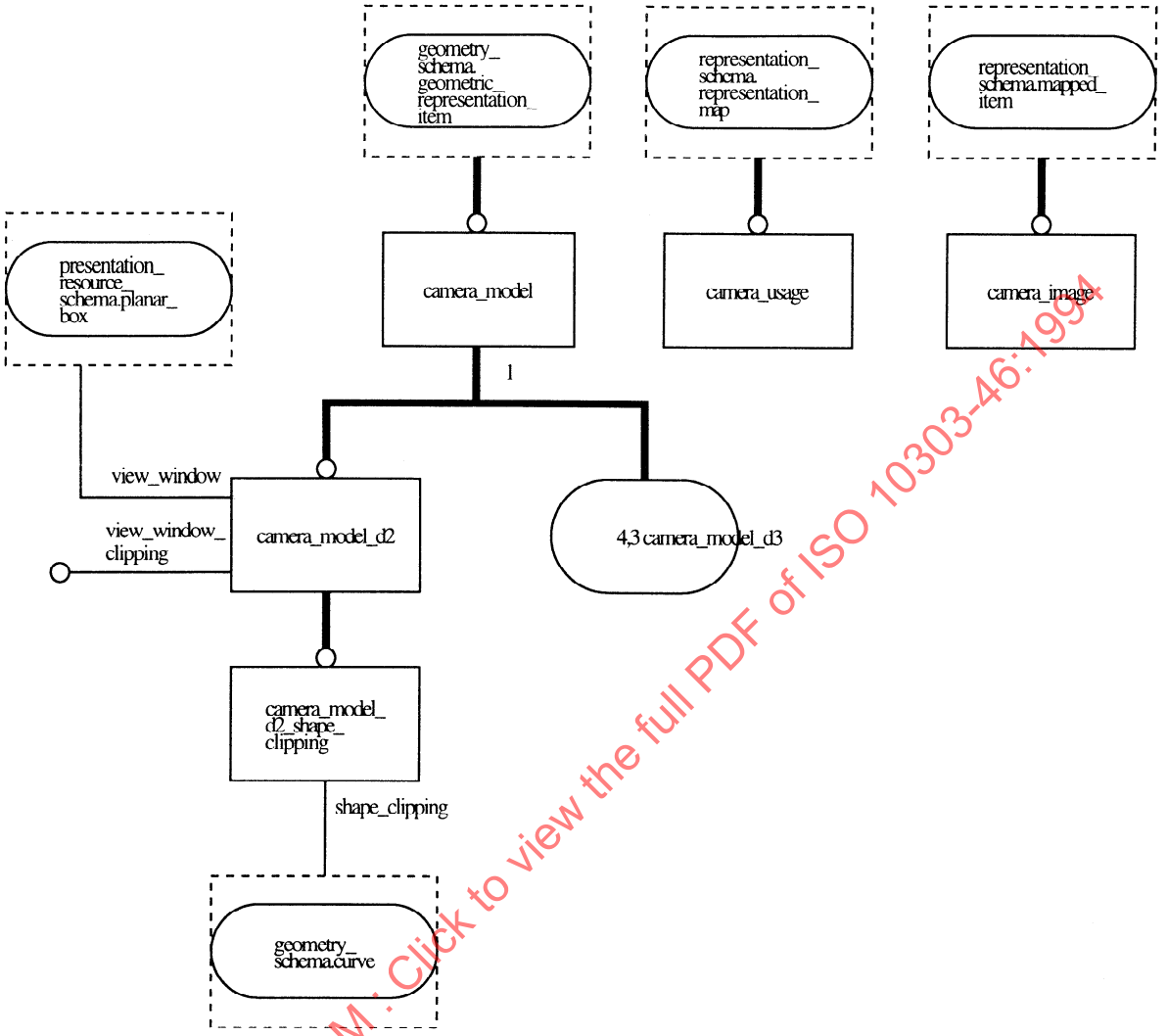


Figure E.3 – presentation_organisation_schema – EXPRESS–G diagram 3 of 7

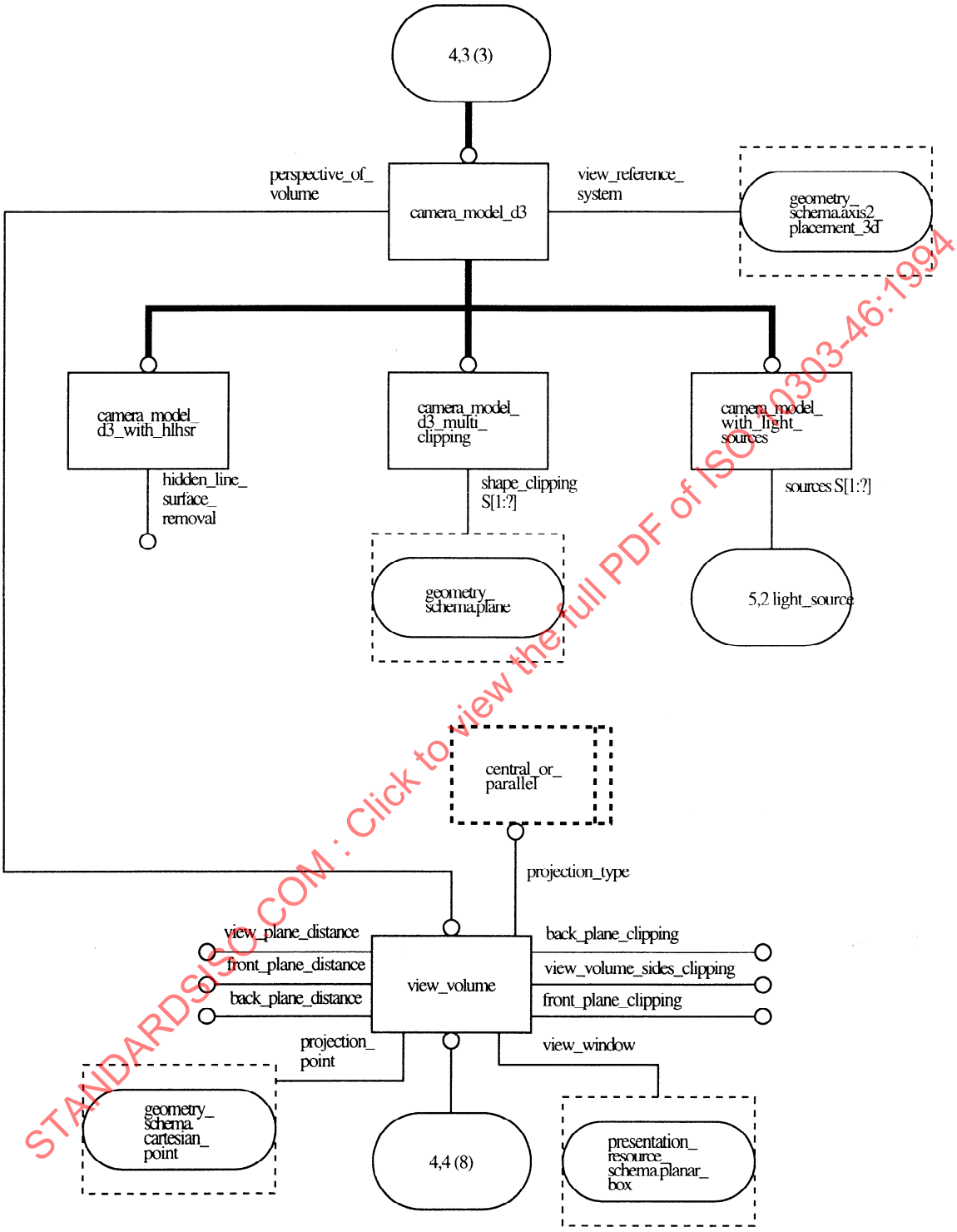


Figure E.4 – presentation_organisation_schema – EXPRESS–G diagram 4 of 7

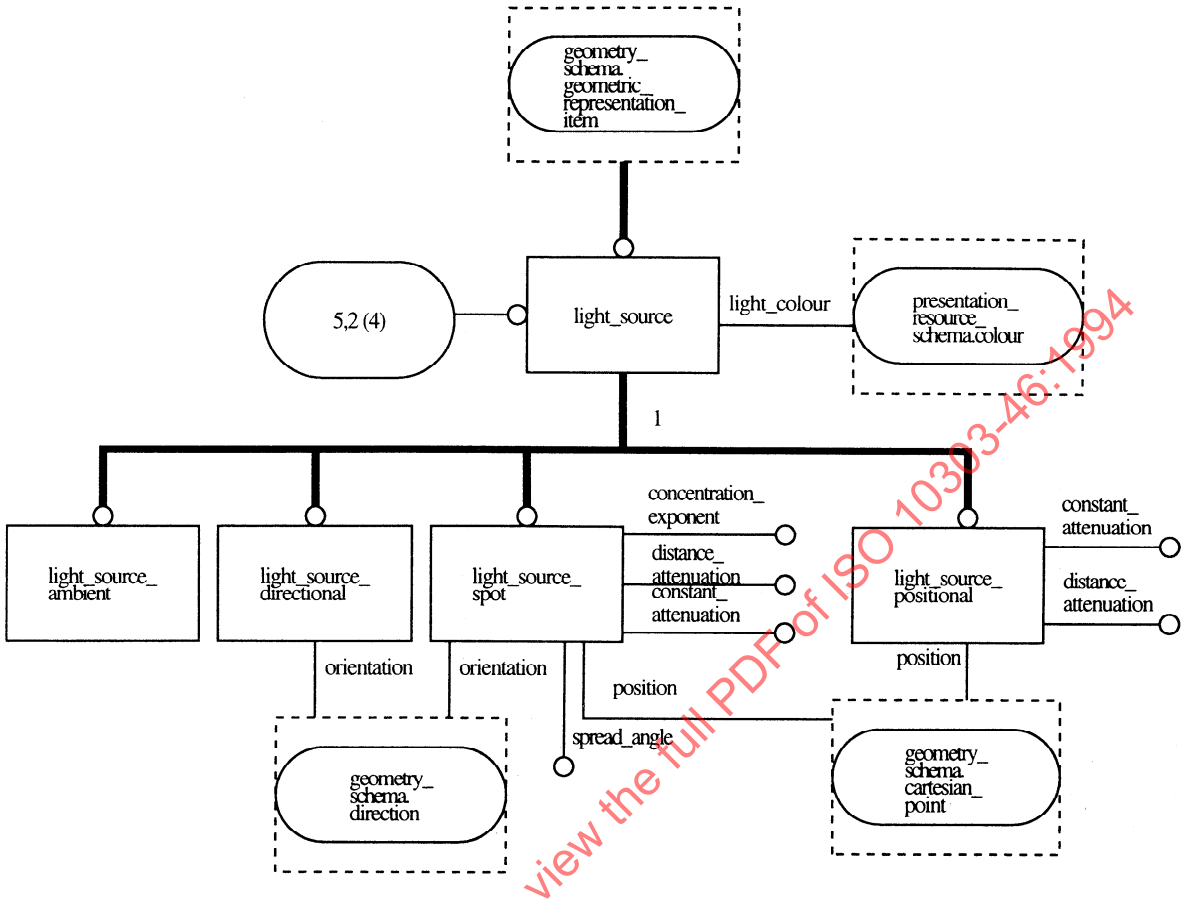


Figure E.5 – presentation_organisation_schema – EXPRESS–G diagram 5 of 7

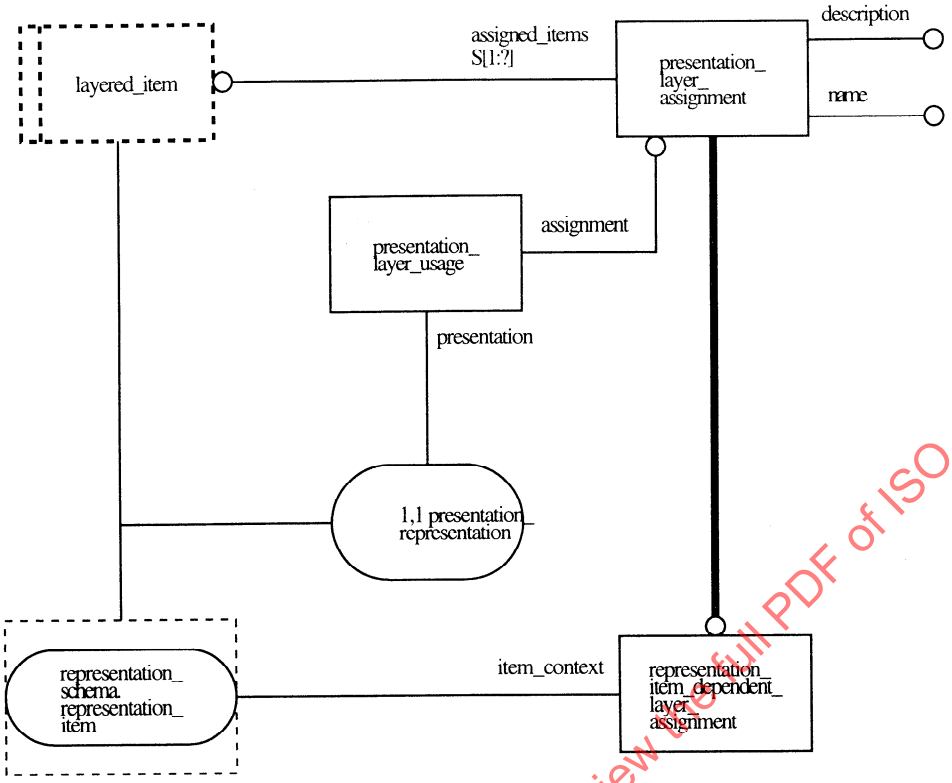


Figure E.6 – presentation_organisation_schema – EXPRESS–G diagram 6 of 7

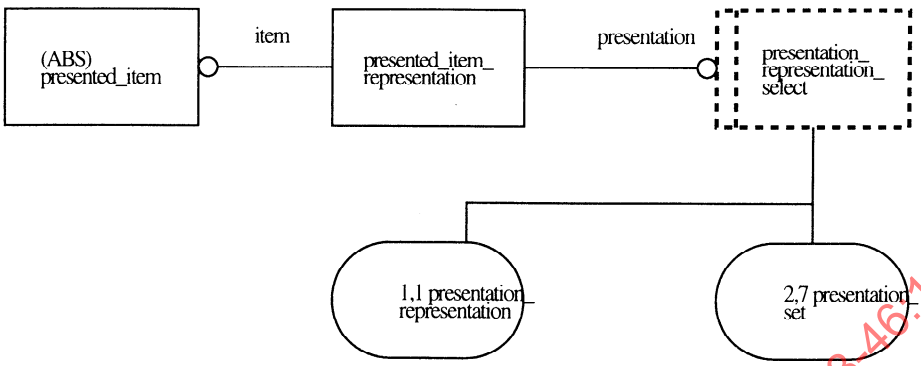


Figure E.7 – presentation_organisation_schema – EXPRESS–G diagram 7 of 7