INTERNATIONAL STANDARD

ISO 10303-112

First edition
2006-12-15

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

Part 112: Integrated application resource: Modelling commands for the exchange of procedurally represented 2D CAD models

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

Partie 112: Ressources d'application intégrée: Commandes de modélisation pour l'échange de modèles 2D CAD représentés en modes opératoires

Reference number
ISO 10303-112:2006(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Figures

Tables

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

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

Attention is drawn to the possibility that some of the elements of this part of ISO 10303 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 10303-112 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Subcommittee SC4, *Industrial data.*

ISO 10303 consists of a series of parts, under the general title *Industrial automation systems and integration — Product data representation and exchange.* The structure of ISO 10303 is described in ISO 10303-1.

Each part of ISO 10303 is a member of one of the following series: description methods, implementation methods, conformance testing methodology and framework, integrated generic resources, integrated application resources, application protocols, abstract test suites, application interpreted constructs, and application modules. This part is a member of the integrated application resources series. The integrated generic resources and the integrated application resources specify a single conceptual product data model.

A complete list of parts of ISO 10303 is available from the following URL:

http://www.tc184-sc4.org/titles/STEP_Titles.htm

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation of product information and for the exchange of product data. The objective is to provide a neutral mechanism capable of describing products throughout their life cycle. This mechanism is suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and as a basis for archiving.

This part of ISO 10303 is a member of the integrated application resources series. This part of ISO 10303 specifies the **procedural_sketch** schema.

This part of ISO 10303 provides general resources for the representation of modelling commands for the exchange of procedurally represented 2D CAD models. Procedural models have the advantage of being easy to edit, simply by changing values of parameters used as arguments of their constructional operations. Such models are said to embody design intent, in the sense that modifications to them conform to the method of creation used by their original creator, and they also comply with any constraints implied by the particular constructional operations used.

ISO 10303-55 specifies the resource constructs for the representation of models of the procedural or construction history type, defined in terms of the sequence of constructional operations used to build them. The mechanisms provided in ISO 10303-55 allow the use of entity data types specified in this part of ISO 10303 for representations of the operations.

The relationships of the schema in this part of ISO 10303 to other schemas that define the integrated resources of ISO 10303 are illustrated in Figure 1 using EXPRESS-G notation. EXPRESS-G is defined in ISO 10303-11:2004, Annex D. The schemas occurring in Figure 1 are components of ISO 10303 integrated resources, and they are specified in the following resource parts:

`measure_schema`	ISO 10303-41
`geometry_schema`	ISO 10303-42
`geometric_model_schema`	ISO 10303-42
`sketch_schema`	ISO 10303-108
`explicit_geometric_constraint_schema`	ISO 10303-108

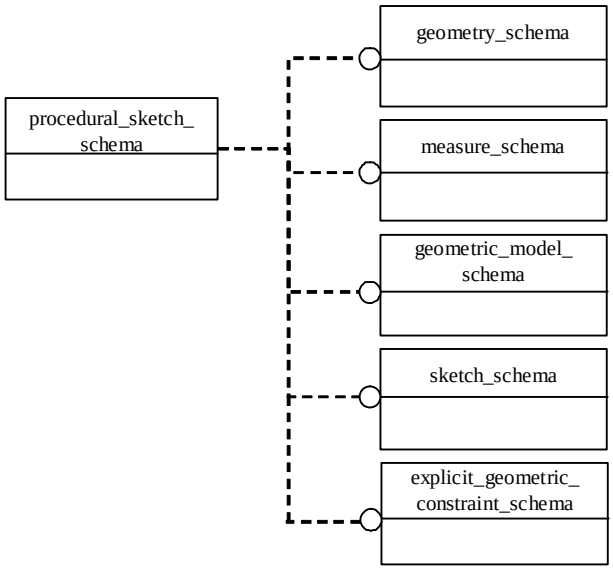


Figure 1 — Schema level diagram of relationships between the ISO 10303-112 schemas and other resource schemas

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

Part 112:
Integrated application resource: Modelling commands for the exchange of procedurally represented 2D CAD models

1 Scope

This part of ISO 10303 specifies the resource constructs for the representation of 2D modelling commands for use in the exchange of procedurally represented 2D CAD models. A procedural model is defined in terms of the sequence of 2D modelling operations used to build it.

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

— entities representing creation commands for geometric elements such as lines, arcs, chamfers and fillets;

— entities representing transformation operations such as rotations and translations.

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

— the identification of selected entities by interactive picking in a procedural model;

— the specification of sequences of modelling commands;

— query, deletion and modification commands (except in the case where the modification is a transformation);

— dimensioning and constraining commands for 2D modelling.

The first and second capabilities in the out of scope list above are provided by ISO 10303-55, and the fourth by ISO 10303-108. This part of ISO 10303 is designed to be used in conjunction with those parts of ISO 10303.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For updated references, the latest edition of the referenced document (including any amendments) applies

ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1) — Part 1: Specification of basic notation.*

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

ISO 10303-11:2004, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

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

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

ISO 10303-55, *Industrial automation systems and integration — Product data representation and exchange — Part 55: Integrated generic resource: Procedural and hybrid representation.*

ISO 10303-108, *Industrial automation systems and integration — Product data representation and exchange — Part 108: Integrated application resource: Parameterization and constraints for explicit geometric product models.*

3 Terms, definitions, and abbreviations

3.1 Terms defined in ISO 10303-1

For the purposes of this document, the following term defined in ISO 10303-1 applies.

— application.

3.2 Terms defined in ISO 10303-11

For the purposes of this document, the following terms defined in ISO 10303-11 apply.

— entity;

— entity data type;

— instance.

3.3 Terms defined in ISO 10303-55

For the purposes of this document, the following terms defined in ISO 10303-55 apply.

— explicit selected elements;

— procedural model.

3.4 Terms defined in ISO 10303-108

For the purposes of this document, the following terms defined in ISO 10303-108 apply.

— constraint;

— explicit constraint;

— sketch.

3.5 Other terms and definitions

For the purposes of this document, the following definition applies.

3.5.1
sketch command
command for the creation of one or more sketch elements.

3.6 Abbreviations

For the purposes of this part of ISO 10303 the following abbreviations apply:

CAD computer aided design

IR integrated resource (of ISO 10303)

4 Procedural sketch

4.1 Introduction

The following EXPRESS declaration begins the procedural sketch schema and identifies the necessary external references.

EXPRESS specification:

```
*)
SCHEMA procedural_sketch_schema;

REFERENCE FROM measure_schema                                -- ISO 10303-41
  (count_measure,
   length_measure,
   plane_angle_measure,
   positive_length_measure,
   positive_plane_angle_measure,
   positive_ratio_measure);

REFERENCE FROM geometry_schema                               -- ISO 10303-42
  (cartesian_point,
   circle,
   conic,
   cross_product,
   curve,
   direction,
   geometric_representation_item,
   line,
   point_on_curve,
   trimmed_curve);

REFERENCE FROM geometric_model_schema                        -- ISO 10303-42
  (geometric_curve_set
   geometric_set);

REFERENCE FROM sketch_schema                                 -- ISO 10303-108
  (sketch_element_select);
```

```
REFERENCE FROM explicit_geometric_constraint_schema        -- ISO 10303-108
  (near_point_relationship,
   non_negative_length_measure);
(*
```

NOTE 1 The schemas referenced above are specified in the following part of ISO 10303:

measure_schema	ISO 10303-41
geometry_schema	ISO 10303-42
geometric_model_schema	ISO 10303-42
sketch_schema	ISO 10303-108
explicit_geometric_constraint_schema	ISO 10303-108

NOTE 2 See annex D, Figures D.1 – D.16, for a graphical presentation of this schema.

4.2 Fundamental concepts and assumptions

The procedural sketch schema defines representations for a set of 2D modelling commands for the creation of a sketch. The intention is to provide a means for the exchange of procedural representations of sketches and other 2D geometric models.

NOTE 1 A common application of such a sketch is extrusion or rotation to generate a 3D model.

NOTE 2 The commands specified in this part of ISO 10303 are concerned purely with geometry. However, the capabilities of ISO 10303-108 may be used to parameterize or constrain elements of the created geometry.

4.2.1 Identification of selected entities

It is intended that **selected_items_in_procedural_shape_rep** as defined in ISO 10303-55 shall be used for the identification of elements selected by the user from the CAD system screen.

4.2.2 Representation of construction history

The sequence of operations used to create a 2D sketch model is referred to as a construction history or a procedural model. This part of ISO 10303 is intended to be used in conjunction with ISO 10303-55, which provides the mechanisms to capture the sequence of operations.

EXAMPLE To represent a construction history, **procedural_shape_representation_sequence** from ISO 10303-55 is used as follows.

```
#10=PROCEDURAL_SHAPE_REPRESENTATION_SEQUENCE('Example',(#33,#34,#13,#14,
#12),$,'sketch1');
#12=SKETCH_OPERATE_FILLET('Fillet1',#17,#18,2.,.T.);
#13=USER_SELECTED_SHAPE_ELEMENTS('',(#17));
#14=USER_SELECTED_SHAPE_ELEMENTS('',(#18));
#17=TRIMMED_CURVE('',#21,(#37),(#38),.T.,.CARTESIAN.);
#18=TRIMMED_CURVE('',#22,(#39),(#40),.T.,.CARTESIAN.);
#33=CREATE_LINE_SEGMENT_2_POINTS('line1',#37,#38);
#34=CREATE_LINE_SEGMENT_2_POINTS('line2',#39,#40);
```

NOTE 1 In the above example, the instance of **procedural_shape_representation_sequence** uses instances of the entity **geometric_representation_item** (defined in ISO 10303-42) as arguments. Because the entities of this part of ISO 10303 are subtypes of **geometric_representation_item**, they can be used for this purpose.

NOTE 2 The above example is a part of the example in clause E.1 of Annex E. Some instances such as #21, #22 are omitted.

4.2.3 Representation of explicit constraints in a 2D sketch

This part of ISO 10303 is intended to be used in conjunction with ISO 10303-108 to represent explicit constraints among sketch elements.

EXAMPLE To represent explicit constraints, resources of ISO 10303-108 are used as shown in the following example

```
#10=PROCEDURAL_SHAPE_REPRESENTATION_SEQUENCE('Example',(#35,#36,#15,#16,#11),$,'sketch
1');
#11=PARALLEL_GEOMETRIC_CONSTRAINT('constraint1','',(#23),(#24));
#15=USER_SELECTED_SHAPE_ELEMENTS('',(#19));
#16=USER_SELECTED_SHAPE_ELEMENTS('',(#20));
#19=TRIMMED_CURVE('',#23,(#41),(#42),.T.,.CARTESIAN.);
#20=TRIMMED_CURVE('',#24,(#43),( #44),.T.,.CARTESIAN.);
#23=LINE('',#41,#27);
#24=LINE('',#43,#28);
#35=CREATE_LINE_SEGMENT_2_POINTS('line3',#41,#42);
#36=CREATE_LINE_SEGMENT_2_POINTS('line4',#43,#44);
```

NOTE 1 In the above example, the instance of **parallel_geometric_constraint** from ISO 10303-108 uses two instances of line (as defined in ISO 10303-42) as arguments. According to the mechanism of ISO 10303-55, these instances represent the explicit geometry of instances #35 and #36. As a result, the instance of **parallel_geometric_constraint** effectively constrains the geometry created by instances #35 and #36.

NOTE 2 The above example is a part of the example in clause E.1 of Annex E. Some instances such as #41, #43 are omitted.

4.3 Procedural sketch type definitions

4.3.1 polygon_circle_type

A **polygon_circle_type** specifies the relationship between the polygon and the circle used to create it.

EXPRESS specification:

```
*)
TYPE polygon_circle_type = ENUMERATION OF
  (inscribed,
   circumscribed);
END_TYPE;
(*
```

Enumerated item definitions:

inscribed: the polygon to be created is inscribed in the circle.

circumscribed: the polygon to be created is circumscribed about the circle.

4.3.2 circle_or_circular_arc

A **circle_or_circular_arc** selects a circle or circular arc. A circular arc is represented by the **trimmed_curve** entity from ISO 10303-42, with the restriction that its basis curve shall be of type **circle** as defined in ISO 10303-42.

EXPRESS specification:

```
*)
TYPE circle_or_circular_arc = SELECT
  (circle,
   trimmed_curve);
WHERE
  WR1: circular_type(SELF);
END_TYPE;
(*
```

Formal propositions:

WR1: If the type of the selected instance is **trimmed_curve** then its attribute **basis_curve** shall be of type **circle**.

4.3.3 line_or_trimmed_line

A **line_or_trimmed_line** selects a line or trimmed line. A trimmed line is represented by the **trimmed_curve** entity from ISO 10303-42, with the restriction that its basis curve shall be of type **line** as defined in ISO 10303-42.

EXPRESS specification:

```
*)
TYPE line_or_trimmed_line = SELECT
  (line,
   trimmed_curve);
WHERE
  WR1: linear_type(SELF);
END_TYPE;
(*
```

Formal propositions:

WR1: If the type of the selected instance is **trimmed_curve** then its attribute **basis_curve** shall be of type **line**.

4.3.4 rotation_direction

A **rotation_direction** specifies the sense of rotation.

EXPRESS specification:

```
*)
TYPE rotation_direction = ENUMERATION OF
  (cw,
   ccw);
END_TYPE;
(*
```

Enumerated item definitions:

cw: the sense of rotation is clockwise.

ccw: the sense of rotation is counterclockwise.

4.3.5 ps_sketch_element_select

A **ps_sketch_element_select** selects a type of sketch or sketch transformation.

EXPRESS specification:

```
*)
TYPE ps_sketch_element_select = SELECT BASED_ON sketch_element_select WITH
  (sketch_create_curve_element,
   sketch_create_pattern_element,
   sketch_operate_transform);
END_TYPE;
(*
```

4.4 procedural_sketch entity definitions

4.4.1 sketch_command

A **sketch_command** represents the creation of a new sketch element or the transformation of an existing sketch element.

EXPRESS specification:

```
*)
ENTITY sketch_command
  ABSTRACT SUPERTYPE OF (ONEOF (sketch_create_curve_element,
  sketch_create_pattern_element, sketch_operate_transform));
END_ENTITY;
(*
```

4.4.2 sketch_create_curve_element

A **sketch_create_curve_element** is a type of **sketch_command** and **geometric_representation-_item** that specifies types of curve creation.

EXPRESS specification:

```
*)
ENTITY sketch_create_curve_element
  ABSTRACT SUPERTYPE OF (ONEOF (create_line_segment, create_centreline,
  create_polyline, create_rectangle, create_polygon, create_circular_arc,
  create_circle, create_ellipse, create_spline, create_parabolic_arc,
  create_fillet, create_chamfer, create_divided_curve))
  SUBTYPE OF (sketch_command, geometric_representation_item);
WHERE
  WR1: SELF\geometric_representation_item.dim = 2;
END_ENTITY;
(*
```

Formal propositions:

WR1: The dimension of this entity shall be 2D.

4.4.3 create_line_segment

A **create_line_segment** is a type of **sketch_create_curve_element** that specifies types of line segment creation.

EXPRESS specification:

```
*)
ENTITY create_line_segment
  ABSTRACT SUPERTYPE OF (ONEOF (create_line_segment_2_points,
```

```
  create_line_segment_point_tangent, create_line_segment_2_tangents))
  SUBTYPE OF (sketch_create_curve_element);
END_ENTITY;
(*
```

4.4.4 create_line_segment_2_points

A **create_line_segment_2_points** is a type of **create_line_segment** that creates a line segment with two given end points.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_line_segment_2_points
  SUBTYPE OF (create_line_segment);
  start_point: cartesian_point;
  end_point        : cartesian_point;
WHERE
  WR1: distance_between_cartesian_points(start_point, end_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

start_point: the start point of the line segment.

end_point: the end point of the line segment.

Formal propositions:

WR1: The start point and the end point shall have distinct positions.

Figure 2 — create_line_segment_2_points

4.4.5 create_line_segment_point_tangent

A **create_line_segment_point_tangent** is a type of **create_line_segment** that creates a line segment which is starting from a given start point and is tangential to a given curve. The end point of the line segment is the point of tangency with the **tangent_curve**.

NOTE The point and the curve referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
```



```
ENTITY create_line_segment_point_tangent
  SUBTYPE OF (create_line_segment);
  start_point                   : cartesian_point;
  tangent_curve                 : curve;
  tangent_curve_near_point: near_point_relationship;
WHERE
  WR1: tangent_curve_near_point\representation_item_relationship.
       relating_representation_item :=: tangent_curve;
END_ENTITY;
(*
```

Attribute definitions:

start_point: the start point of the line segment.

tangent_curve: the curve tangential to the line.

tangent_near_point: the **near_point_relationship** instance giving the approximate location of an intended tangency to the curve.

Formal propositions:

WR1: The associated curve for the specified near point shall correspond to the tangent curve.

4.4.6 create_line_segment_2_tangents

A **create_line_segment_2_tangents** is a type of **create_line_segment** that creates a line segment tangential to 2 given curves. The start and end points of the line segment are the points of tangency with the first and second curves, respectively.

NOTE The first and second curves referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_line_segment_2_tangents
  SUBTYPE OF (create_line_segment);
  first_curve       : curve;
  first_near_point        : near_point_relationship;
  second_curve            : curve;
  second_near_point : near_point_relationship;
WHERE
  WR1: first_curve <> second_curve;
  WR2: first_near_point\representation_item_relationship.
       relating_representation_item :=: first_curve;
  WR3: second_near_point\representation_item_relationship.
       relating_representation_item :=: second_curve;
END_ENTITY;
(*
```

Attribute definitions:

first_curve: the first curve tangential to the line segment.

first_near_point: the first **near_point_relationship** instance giving the approximate location of an intended tangency to the first curve.

second_curve: the second curve tangential to the line segment.

second_near_point: the second **near_point_relationship** instance giving the approximate location of an intended tangency to the second curve.

Formal propositions:

WR1: The first curve shall be different from the second curve.

WR2: The associated curve for the specified first near point shall correspond to the first curve.

WR3: The associated curve for the specified second near point shall correspond to the second curve.

4.4.7 create_centreline

A **create_centreline** is a type of **sketch_create_curve_element** that creates a centreline passing through two points in the sketch plane. A centreline is an auxiliary geometric element intended for use as an axis of rotation or symmetry.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_centreline
  SUBTYPE OF (sketch_create_curve_element);
  first_ref_point         : cartesian_point;
  second_ref_point        : cartesian_point;
WHERE
  WR1: distance_between_cartesian_points(first_ref_point, second_ref_point)
       <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

first_ref_point: the first reference point on the centreline.

second_ref_point: the second reference point on the centreline.

Formal propositions:

WR1: The first reference point and the second reference point shall have distinct positions.

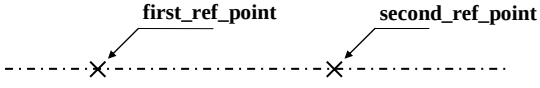


Figure 3 — create_centreline

4.4.8 create_polyline

A **create_polyline** is a type of **sketch_create_curve_element** that creates a polyline connecting a sequence of points on a plane.

NOTE All the points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_polyline
  SUBTYPE OF (sketch_create_curve_element);
  points     : LIST [3:?] OF cartesian_point;
WHERE
  WR1: distinct_points(points);
END_ENTITY;
(*
```

Attribute definitions:

points: the sequence of input points.

Formal propositions:

WR1: Any two successive points of the **points** list shall have distinct positions.

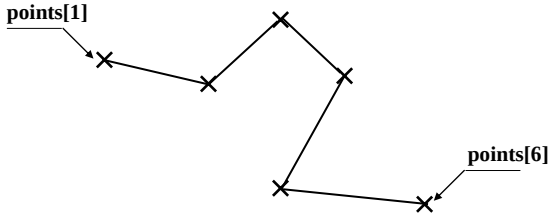


Figure 4 — create_polyline

4.4.9 create_rectangle

A **create_rectangle** is a type of **sketch_create_curve_element** that creates a rectangle in terms of three points. The first two points define the width and the rotation angle of the rectangle. The perpendicular distance of the third point from the line joining the first two points defines the height of the rectangle.

NOTE 1 The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_rectangle
  SUBTYPE OF (sketch_create_curve_element);
```

```
  first_point       : cartesian_point;
  second_point      : cartesian_point;
  third_point       : cartesian_point;
WHERE
  WR1: non_collinear_2d_points (first_point, second_point, third_point);
END_ENTITY;
(*
```

Attribute definitions:

first_point: the first point defining the rectangle.

second_point: the second point, which together with the first point defines the length and direction of one side of the rectangle.

third_point: the third point, which together with the first two points defines the height of the rectangle.

Formal propositions:

WR1: The three points shall not be collinear.

NOTE 2 The use in **WR1** of function **non_collinear_2d_points** defined in clause 4.5.2 ensures that no two of the defining points are coincident.

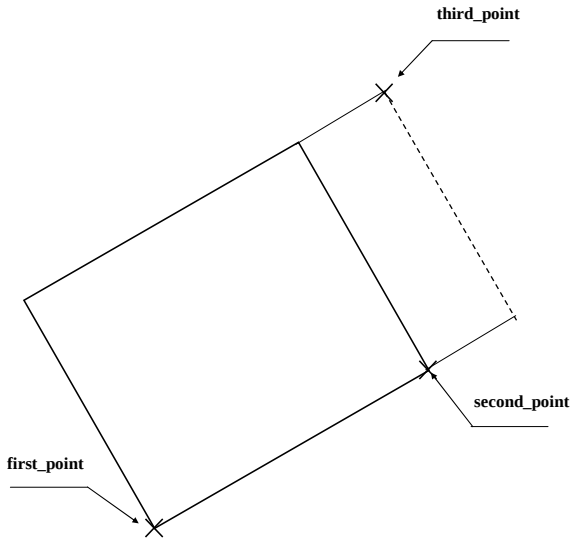


Figure 5 — create_rectangle

4.4.10 create_polygon

A **create_polygon** is a type of **sketch_create_curve_element** that creates a regular n-sided polygon which is either inscribed in a circle or circumscribed about a circle, the choice being indicated by the value of the attribute **circle_type**.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_polygon
```



```
  SUBTYPE OF (sketch_create_curve_element);
  polygon_sides           : count_measure;
  centre_point            : cartesian_point;
  start_point             : cartesian_point;
  circle_type             : polygon_circle_type;
  circle_radius           : positive_length_measure;
WHERE
  WR1: distance_between_cartesian_points(centre_point, start_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

polygon_sides: the number of sides of the polygon.

centre_point: the centre point of the polygon.

start_point: the start point of the polygon.

circle_type: the enumeration value that defines the relationship (inscribed or circumscribed) between the polygon and the circle.

circle_radius: the radius of the circle

Formal propositions:

WR1: The centre point and the start point shall have distinct positions.

4.4.11 create_circular_arc

A **create_circular_arc** is a type of **sketch_create_curve_element** that specifies types of circular arc creation.

EXPRESS specification:

```
*)
ENTITY create_circular_arc
  ABSTRACT SUPERTYPE OF (ONEOF (create_circular_arc_concentric,
  create_circular_arc_3_tangents, create_circular_arc_centre_ends,
  create_circular_arc_start_centre_angle,
  create_circular_arc_start_centre_length,
  create_circular_arc_start_end_angle,
  create_circular_arc_start_end_direction,
  create_circular_arc_start_end_radius, create_circular_arc_3_points,
  create_circular_arc_angles))
  SUBTYPE OF (sketch_create_curve_element);
END_ENTITY;
(*
```

4.4.12 create_circular_arc_concentric

A **create_circular_arc_concentric** is a type of **create_circular_arc** that creates a circular arc with the specified start and end points and is concentric with a given **circle_or_circular_arc**.

NOTE The two points and the curve referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_concentric
  SUBTYPE OF (create_circular_arc);
  reference_curve               : circle_or_circular_arc;
  start_point                   : cartesian_point;
  end_point                     : cartesian_point;
  arc_rotation_direction        : rotation_direction;
WHERE
  WR1: distance_between_cartesian_points(start_point, end_point) <> 0.0;
  WR2: distance_between_cartesian_points(start_point,
       centre_of_circle_or_circular_arc(reference_curve)) =
       distance_between_cartesian_points (end_point,
       centre_of_circle_or_circular_arc(reference_curve));
END_ENTITY;
(*
```

Attribute definitions:

reference_curve: the entity used to determine the centre point of the arc.

start_point: the start point of the arc.

end_point: the end point of the arc.

arc_rotation_direction: the rotation direction of the arc.

Formal propositions:

WR1: The start point and the end point shall have distinct positions.

WR2: The **start_point** and the **end_point** shall be equidistant from the centre-point of the **reference_curve**.

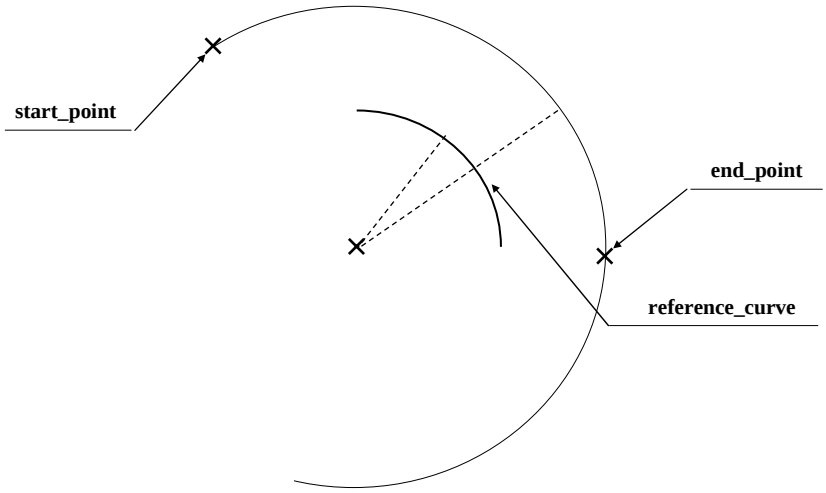


Figure 6 — create_circular_arc_concentric

4.4.13 create_circular_arc_3_tangents

A **create_circular_arc_3_tangents** is a type of **create_circular_arc** that creates a circular arc tangential to 3 given curves. The start and end points of the arc are the points of tangency with the first and third curves, respectively.

NOTE The three curves and the three points involved in the referenced near-point relationships shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_3_tangents
  SUBTYPE OF (create_circular_arc);
  first_curve             : curve;
  first_near_point        : near_point_relationship;
  second_curve            : curve;
  second_near_point       : near_point_relationship;
  third_curve             : curve;
  third_near_point        : near_point_relationship;
WHERE
  WR1: first_curve <> second_curve;
  WR2: second_curve <> third_curve;
  WR3: third_curve <> first_curve;
  WR4: first_near_point\representation_item_relationship.
       relating_representation_item :=: first_curve;
  WR5: second_near_point\representation_item_relationship.
       relating_representation_item :=: second_curve;
  WR6: third_near_point\representation_item_relationship.
       relating_representation_item :=: third_curve;
END_ENTITY;
(*
```

Attribute definitions:

first_curve: the first curve tangential to the arc.

first_near_point: the first **near_point_relationship** instance giving the approximate location of an intended tangency to the first curve.

second_curve: the second curve tangential to the arc.

second_near_point: the second **near_point_relationship** instance giving the approximate location of an intended tangency to the second curve.

third_curve: the third curve tangential to the arc.

third_near_point: the third **near_point_relationship** instance giving the approximate location of an intended tangency to the third curve.

Formal propositions:

WR1: The first curve shall be different from the second curve.

WR2: The second curve shall be different from the third curve.

WR3: The third curve shall be different from the first curve.

WR4: The associated curve for the specified first near point shall correspond to the first curve.

WR5: The associated curve for the specified second near point shall correspond to the second curve.

WR6: The associated curve for the specified third near point shall correspond to the third curve.

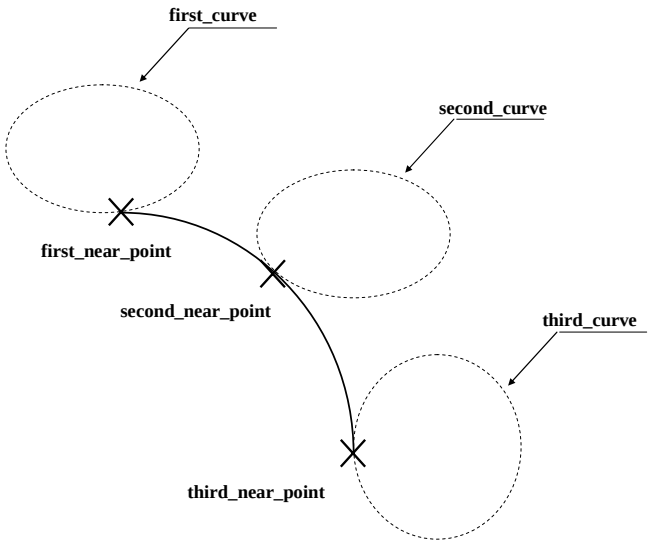


Figure 7 — create_circular_arc_3_tangents

4.4.14 create_circular_arc_centre_ends

A **create_circular_arc_centre_ends** is a type of **create_circular_arc** that creates a circular arc in terms of a centre point and two end points. As this information alone defines two arcs, the attribute **arc_rotation_direction** is provided to specify which arc is selected.

NOTE The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_centre_ends
  SUBTYPE OF (create_circular_arc);
  centre_point                  : cartesian_point;
  start_point                   : cartesian_point;
  end_point                     : cartesian_point;
  arc_rotation_direction        : rotation_direction;
WHERE
  WR1: three_distinct_points(start_point, end_point, centre_point);
  WR2: distance_between_cartesian_points (centre_point, start_point) =
       distance_between_cartesian_points (centre_point, end_point);
END_ENTITY;
(*
```

Attribute definitions:

centre_point: the centre point of the arc.

start_point: the start point of the arc.

end_point: the end point of the arc.

arc_rotation_direction: the rotation direction of the arc.

Formal propositions:

WR1: The start point, the end point and the centre point shall have distinct positions.

WR2: The start point and the end point shall be equidistant from the centre point.

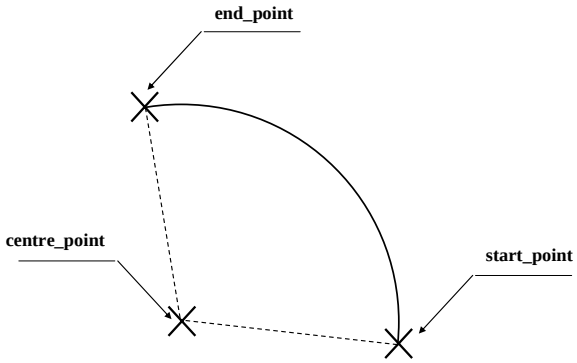


Figure 8 — create_circular_arc_centre_ends

4.4.15 create_circular_arc_start_centre_angle

A **create_circular_arc_start_centre_angle** is a type of **create_circular_arc** that creates a circular arc in terms of a start point, a centre point and an angle of end position. As this information alone defines two arcs, the attribute **arc_rotation_direction** is provided to specify which arc is selected.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_start_centre_angle
  SUBTYPE OF (create_circular_arc);
  centre_point                  : cartesian_point;
  start_point                   : cartesian_point;
  end_angle                     : positive_plane_angle_measure;
  arc_rotation_direction        : rotation_direction;
WHERE
  WR1: distance_between_cartesian_points(centre_point, start_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

centre_point: the centre point of the arc.

start_point: the start point of the arc.

end_angle: the end angle of the arc.

arc_rotation_direction: the rotation direction of the arc.

Formal propositions:

WR1: The centre point and the start point shall have distinct positions.

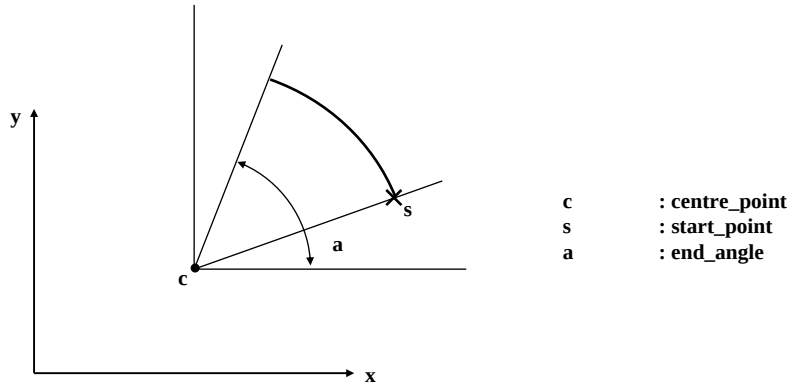


Figure 9 — create_circular_arc_start_centre_angle

4.4.16 create_circular_arc_start_centre_length

A **create_circular_arc_start_centre_length** is a type of **create_circular_arc** that creates a circular arc in terms of a start point, a centre point and a chord length. As this information alone defines two arcs, the attribute **arc_rotation_direction** is provided to specify which arc is selected.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_start_centre_length
  SUBTYPE OF (create_circular_arc);
  centre_point                  : cartesian_point;
  start_point                   : cartesian_point;
  chord_length                  : positive_length_measure;
  arc_rotation_direction        : rotation_direction;
WHERE
  WR1: distance_between_cartesian_points(centre_point, start_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

centre_point: the centre point of the arc.

start_point: the start point of the arc.

chord_length: length of chord of the arc.

arc_rotation_direction: the rotation direction of the arc.

Formal propositions:

WR1: The centre point and the start point shall have distinct positions.

4.4.17 create_circular_arc_start_end_angle

A **create_circular_arc_start_end_angle** is a type of **create_circular_arc** that creates a circular arc in terms of a start point, an end point and an included angle for the arc. As this information alone defines two arcs, the attribute **arc_rotation_direction** is provided to specify which arc is selected.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_start_end_angle
  SUBTYPE OF (create_circular_arc);
  start_point                  : cartesian_point;
  end_point                    : cartesian_point;
  arc_angle                    : positive_plane_angle_measure;
  arc_rotation_direction       : rotation_direction;
WHERE
  WR1: distance_between_cartesian_points(start_point, end_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

start_point: the start point of the arc.

end_point: the end point of the arc.

arc_angle: the included angle of the arc.

arc_rotation_direction: the rotation direction of the arc.

Formal propositions:

WR1: The start point and the end point shall have distinct positions.

4.4.18 create_circular_arc_start_end_direction

A **create_circular arc_startenddirection** is a type of **create_circular_arc** that creates a circular arc in terms of a start point, an end point and a tangential direction at the start point of the arc.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_start_end_direction
  SUBTYPE OF (create_circular_arc);
  start_point             : cartesian_point;
  end_point         : cartesian_point;
  start_direction         : direction;
WHERE
  WR1: distance_between_cartesian_points(start_point, end_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

start_point: the start point of the arc.

end_point: the end point of the arc.

start_direction: tangential direction of the start point of the arc.

Formal propositions:

WR1: The start point and the end point shall have distinct positions.

4.4.19 create_circular_arc_start_end_radius

A **create_circular_arc_start_end_radius** creates a circular arc in terms of a start point, an end point and a radius. As this information alone defines two arcs, the attribute **arc_rotation_direction** is provided to specify which arc is selected.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_start_end_radius
  SUBTYPE OF (create_circular_arc);
  start_point                   : cartesian_point;
  end_point                     : cartesian_point;
  radius                        : positive_length_measure;
  arc_rotation_direction        : rotation_direction;
WHERE
  WR1: distance_between_cartesian_points(start_point, end_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

start_point: the start point of the arc.

end_point: the end point of the arc.

radius: the radius of the arc.

arc_rotation_direction: the rotation direction of the arc.

Formal propositions:

WR1: The start point and the end point shall have distinct positions.

4.4.20 create_circular_arc_3_points

A **create_circular_arc_3_points** is a type of **create_circular_arc** that creates a circular arc passing through three given points.

NOTE 1 The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:


```
*)
ENTITY create_circular_arc_3_points
  SUBTYPE OF (create_circular_arc);
  first_point             : cartesian_point;
  second_point            : cartesian_point;
  third_point             : cartesian_point;
WHERE
  WR1: non_collinear_2d_points (first_point, second_point, third_point);
END_ENTITY;
(*
```

Attribute definitions:

first_point: the start point of the arc.

second_point: an intermediate point on the arc.

third_point: the end point of the arc.

Formal propositions:

WR1: The three points shall not be collinear.

NOTE 2 The use in **WR1** of function **non_collinear_2d_points** defined in clause 4.5.2 ensures that no two of the defining points are coincident.

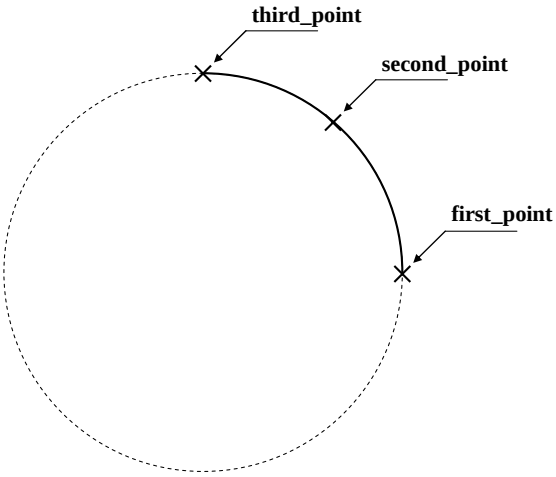


Figure 10 — create_circular_arc_3_points

4.4.21 create_circular_arc_angles

A **create_circular_arc_angles** is a type of **create_circular_arc** that creates a circular arc in terms of a centre point, a start angle, an end angle and a radius. The start angle and the end angle are measured counterclockwise relative to the horizontal direction (1,0).

NOTE The point referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circular_arc_angles
  SUBTYPE OF (create_circular_arc);
  centre_point            : cartesian_point;
  radius                  : positive_length_measure;
  start_angle             : plane_angle_measure;
  end_angle               : plane_angle_measure;
WHERE
  WR1: start_angle <> end_angle;
END_ENTITY;
(*
```

Attribute definitions:

centre_point: the centre point of the arc.

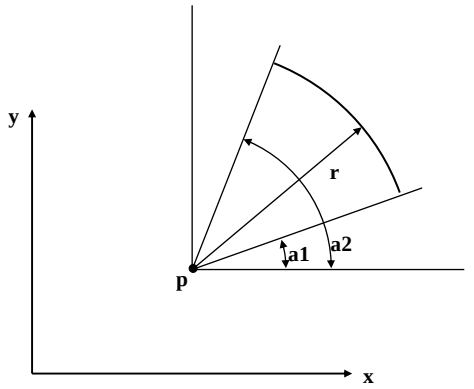
radius: the radius of the arc.

start_angle: the start angle of the arc.

end_angle: the end angle of the arc.

Formal propositions:

WR1: The start angle shall be different from the end angle.



p : centre_point
a1 : start_angle
a2 : end_angle
r : radius

Figure 11 — create_circular_arc_angles

4.4.22 create_circle

A **create_circle** is a type of **sketch_create_curve_element** that specifies types of circle creation.

NOTE The point referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circle
  ABSTRACT SUPERTYPE OF (ONEOF (create_circle_centre_point,
  create_circle_concentric, create_circle_3_tangents,
  create_circle_2_points, create_circle_3_points))
  SUBTYPE OF (sketch_create_curve_element);
END_ENTITY;
(*
```

4.4.23 create_circle_centre_point

A **create_circle_centre_point** is a type of **create_circle** that creates a circle in terms of a centre point and a radius.

NOTE The point referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circle_centre_point
  SUBTYPE OF (create_circle);
  centre_point            : cartesian_point;
  radius                  : positive_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

centre_point: the centre point of the circle.

radius: the radius of the circle.

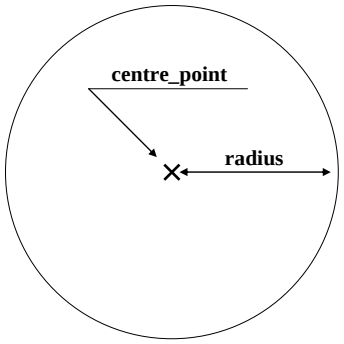


Figure 12 — create_circle_centre_point

4.4.24 create_circle_concentric

A **create_circle_concentric** is a type of **create_circle** that creates a circle that has a specified radius and the same centre point as a reference circle or circular arc.

NOTE The reference element referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circle_concentric
  SUBTYPE OF (create_circle);
  reference_element       : circle_or_circular_arc;
  radius                  : positive_length_measure;
END_ENTITY;
(*
```

Attribute definitions:

reference_element: the circle or circular arc whose centre point defines the centre of the circle to create.

radius: the radius of the circle.

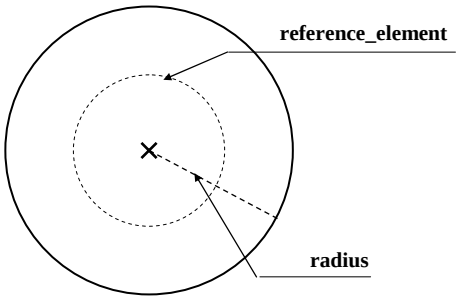


Figure 13 — create_circle_concentric

4.4.25 create_circle_3_tangents

A **create_circle_3_tangents** is a type of **create_circle** that creates a circle tangential to three given entities.

NOTE The three curves and the three points involved in the referenced near-point relationships shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circle_3_tangents
  SUBTYPE OF (create_circle);
  first_curve             : curve;
  first_near_point        : near_point_relationship;
  second_curve            : curve;
  second_near_point       : near_point_relationship;
  third_curve             : curve;
  third_near_point        : near_point_relationship;
WHERE
  WR1: first_curve <> second_curve;
  WR2: second_curve <> third_curve;
  WR3: third_curve <> first_curve;
  WR4: first_near_point\representation_item_relationship.
       relating_representation_item :=: first_curve;
  WR5: second_near_point\representation_item_relationship.
       relating_representation_item :=: second_curve;
  WR6: third_near_point\representation_item_relationship.
       relating_representation_item :=: third_curve;
END_ENTITY;
(*
```

Attribute definitions:

first_curve: the first curve tangential to the circle.

first_near_point: the first **near_point_relationship** instance giving the approximate location of an intended tangency to the first curve.

second_curve: the second curve tangential to the circle.

second_near_point: the second **near_point_relationship** instance giving the approximate location of an intended tangency to the second curve.

third_curve: the third curve tangential to the circle.

third_near_point: the third **near_point_relationship** instance giving the approximate location of an intended tangency to the third curve.

Formal propositions:

WR1: The first curve shall be different from the second curve.

WR2: The second curve shall be different from the third curve.

WR3: The third curve shall be different from the first curve.

WR4: The associated curve for the specified first near point shall correspond to the first curve.

WR5: The associated curve for the specified second near point shall correspond to the second curve.

WR6: The associated curve for the specified third near point shall correspond to the third curve.

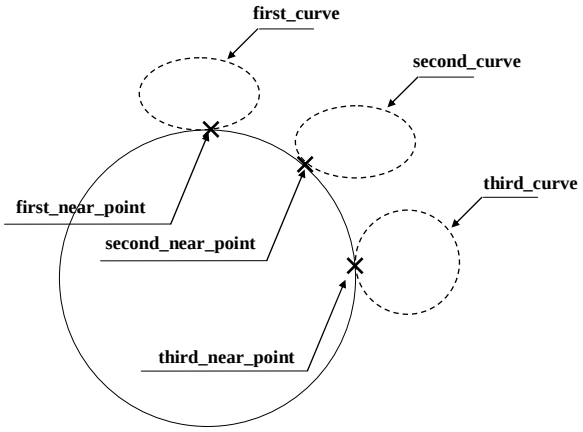


Figure 14 — create_circle_3_tangents

4.4.26 create_circle_2_points

A **create_circle_2_points** is a type of **create_circle** that creates a circle in terms of two points, which are interpreted as end points of a diameter of the circle.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circle_2_points
  SUBTYPE OF (create_circle);
```

```
  first_point             : cartesian_point;
  second_point            : cartesian_point;
DERIVE
  diameter: length_measure := distance_between_cartesian_points
  (first_point, second_point);
WHERE
  WR1: distance_between_cartesian_points(first_point, second_point) <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

first_point: the first point on the circle.

second_point: the second point on the circle.

diameter: the diameter of the circle.

Formal propositions:

WR1: The first point and the second point shall have distinct positions.

4.4.27 create_circle_3_points

A **create_circle_3_points** is a type of **create_circle** that creates a circle passing through three given points.

NOTE 1 The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_circle_3_points
  SUBTYPE OF (create_circle);
  first_point             : cartesian_point;
  second_point            : cartesian_point;
  third_point             : cartesian_point;
WHERE
  WR1: non_collinear_2d_points (first_point, second_point, third_point);
END_ENTITY;
(*
```

Attribute definitions:

first_point: the first point on the circle.

second_point: the second point on the circle.

third_point: the third point on the circle.

Formal propositions:

WR1: The three defining points shall not be collinear.

NOTE 2 The use in **WR1** of the function **non_collinear_2d_points** defined in clause 4.5.2 ensures that no two of the defining points are coincident.

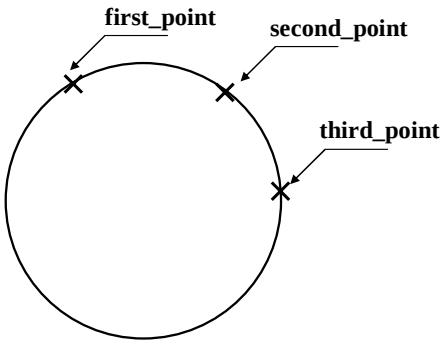


Figure 15 — create_circle_3_points

4.4.28 create_ellipse

A **create_ellipse** is a type of **sketch_create_curve_element** that specifies types of ellipse creation.

EXPRESS specification:

```
*)
ENTITY create_ellipse
  ABSTRACT SUPERTYPE OF (ONEOF (create_ellipse_3_points,
  create_ellipse_centre_point))
  SUBTYPE OF (sketch_create_curve_element);
END_ENTITY;
(*
```

4.4.29 create_ellipse_3_points

A **create_ellipse_3_points** is a type of **create_ellipse** that creates an ellipse in terms of three given points. The first two points are interpreted as end points of one axis of the created ellipse, and the third point is used to determine the length of the second semi-axis of the ellipse. The length of its projection onto the line defined by the first two points defines the second semi-axis length.

NOTE The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_ellipse_3_points
  SUBTYPE OF (create_ellipse);
  first_point             : cartesian_point;
  second_point            : cartesian_point;
  third_point             : cartesian_point;
WHERE
  WR1: non_collinear_2d_points(first_point, second_point, third_point);
END_ENTITY;
(*
```

Attribute definitions:

first_point: the first point, which determines one end of a semi-axis of the ellipse.

second_point: the second point, which determines the other end of the same semi-axis of the ellipse.

third_point: the third point, which determines the length of the other semi-axis of the ellipse.

Formal propositions:

WR1: The first point, the second point and the third point shall not be collinear.

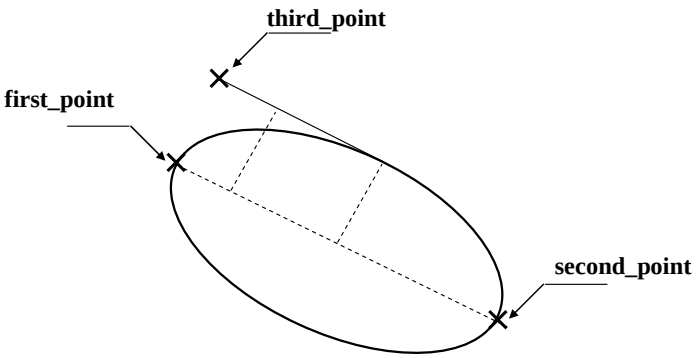


Figure 16 — create_ellipse_3_points

4.4.30 create_ellipse_centre_point

A **create_ellipse_centre_point** is a type of **create_ellipse** that creates an ellipse in terms of a centre point and two points lying on the ellipse. The line joining the centre point and the first point on the ellipse define one semi-axis of the ellipse. The second point is used to determine the length of the second semi-axis of the ellipse. The length of its projection onto the line defined by the first two points defines that length.

NOTE The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_ellipse_centre_point
  SUBTYPE OF (create_ellipse);
  centre_point            : cartesian_point;
  first_point             : cartesian_point;
  second_point            : cartesian_point;
WHERE
  WR1: non_collinear_2d_points(centre_point, first_point, second_point);
END_ENTITY;
(*
```

Attribute definitions:

centre_point: the centre point of the ellipse.

first_point: the point used, together with the centre point, to define the first semi-axis of the ellipse.

second_point: the point, which determines the length along the other semi-axis of the ellipse.

Formal propositions:

WR1: The centre point, the first point, and the second point shall not be collinear.

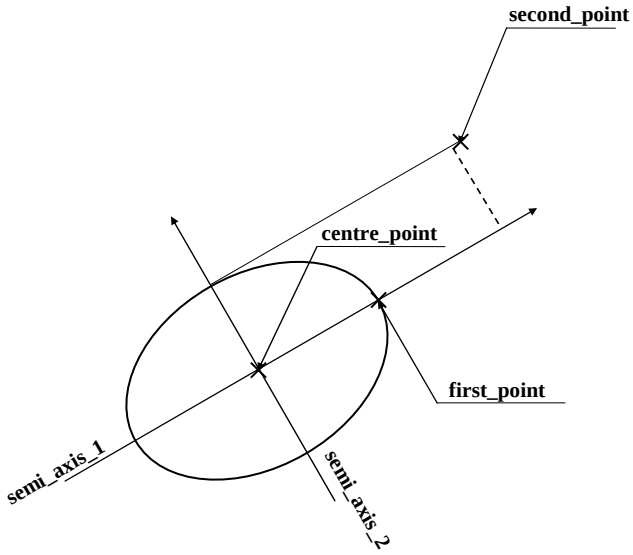


Figure 17 — create_ellipse_centre_point

4.4.31 create_spline

A **create_spline** is a type of **sketch_create_curve_element** that creates a cubic spline curve interpolating 4 or more points. A cubic spline curve is a piecewise polynomial of degree 3 with continuity of derivatives of order 2 at the common joints between segments. A Boolean attribute **closed_curve** is provided to distinguish between open and closed curves; in the closed case the curve has second order continuity across its end points, in the open case the curvature at the ends of the spline curve is zero.

NOTE All the points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_spline
  SUBTYPE OF (sketch_create_curve_element);
  points                  : LIST[4:?] OF cartesian_point;
  closed                  : BOOLEAN;
WHERE
  WR1: distinct_points(points);
END_ENTITY;
(*
```

Attribute definitions:

points: an ordered list of points through which the spline passes.

closed: TRUE if the curve is closed, and FALSE if the curve is open.

Formal propositions:

WR1: Any two successive points of the **points** list shall have distinct positions.

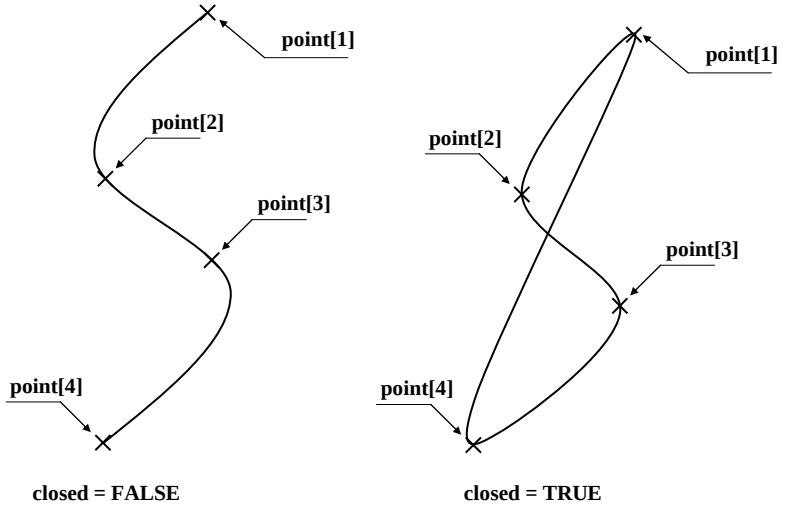


Figure 18 — create_spline

4.4.32 create_parabolic_arc

A **create_parabolic_arc** is a type of **sketch_create_curve_element** that creates a parabolic arc with two given end points, passing through a given intermediate (shoulder) point

NOTE The three points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_parabolic_arc
  SUBTYPE OF (sketch_create_curve_element);
  first_end_point          : cartesian_point;
  intermediate_point       : cartesian_point;
  second_end_point         : cartesian_point;
WHERE
  WR1: three_distinct_points(first_end_point, intermediate_point,
       second_end_point);
END_ENTITY;
(*
```

Attribute definitions:

first_end_point: the first end point of the parabolic arc.

intermediate_point: the intermediate point through which the parabolic arc passes.

second_end_point: the second end point of the parabolic arc.

Formal propositions:

WR1: The first end point, the intermediate point and the second end point shall have distinct positions.

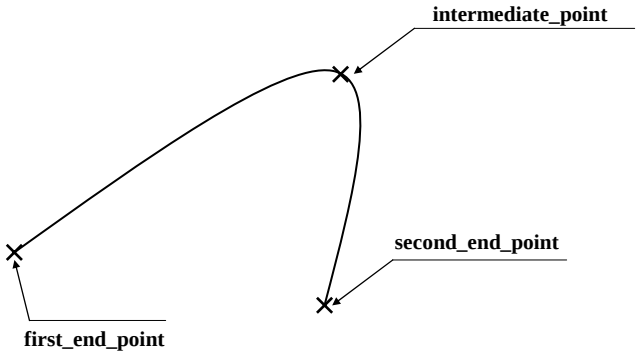


Figure 19 — create_parabolic_arc

4.4.33 create_fillet

A **create_fillet** is a type of **sketch_create_curve_element** that creates a fillet curve with a specified radius to smoothe the angle of intersection between two selected curves**.** The attributes **to_right_of_first_curve** and **to_right_of_second_curve** determine on which side of each of the filleted curves the fillet is to be created. The **near_point** is a point close to the filleted intersection.

NOTE 1 The two curves and the point involved in the referenced near-point relationship shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

NOTE 2 Although no directional sense is associated with instances of **sketch_create_curve_element**, the curves to be filleted by this operation have been selected by the user, and the selection mechanism provided in ISO 10303-55 ensures that explicit representations are available for them, including the required sense information.

The additional BOOLEAN-valued attribute **trim_option** determines whether or not the filleted curves are to be trimmed at fillet ends.

NOTE 3 The effect of the **trim_option** attribute is illustrated in Figure 20.

EXPRESS specification:

```
*)
ENTITY create_fillet
  SUBTYPE OF (sketch_create_curve_element);
  first_curve                 : curve;
  to_right_of_first_curve     : BOOLEAN;
  second_curve                : curve;
  to_right_of_second_curve    : BOOLEAN;
  radius                      : positive_length_measure;
  trim_option                 : BOOLEAN;
  near_point                  : cartesian_point;
WHERE
  WR1: first_curve <> second_curve;
```

```
END_ENTITY;
(*
```

Attribute definitions:

first_curve: the first of the two intersecting curves.

to_right_of_first_curve: TRUE if the fillet is positioned to the right of the first curve and FALSE if it is to the left.

second_curve: the second of the two intersecting curves.

to_right_of_second_curve: TRUE if the fillet is positioned to the right of the second curve and FALSE if it is to the left.

radius: the fillet radius.

trim_option: TRUE if the filleted curves are trimmed at fillet ends after the creation of the fillet and FALSE if the complete curves are retained.

near_point: a point close to the filleted intersection.

Formal propositions:

WR1: The first curve shall be different from the second curve.

Informal propositions:

IP1: **first_curve** and **second_curve** shall have at least one point of intersection.

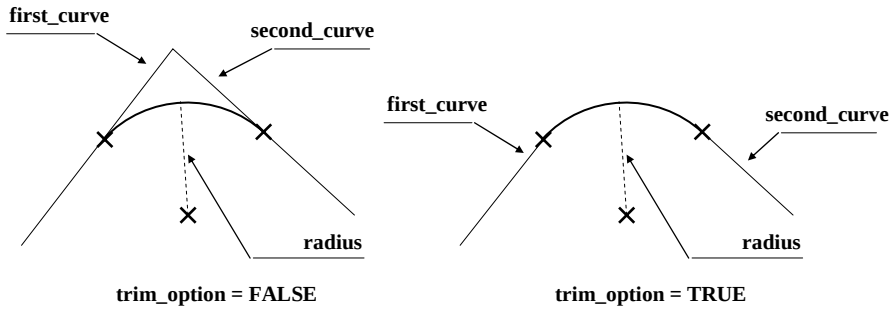


Figure 20 — create_fillet

4.4.34 create_chamfer

A **create_chamfer** is a type of **sketch_create_curve_element** that creates a chamfer with specified characteristics between two selected lines. The attributes **to_right_of_first_line** and **to_right_of_second_line** determine on which side of each of the chamfered lines the chamfer is to be created.

NOTE 1 The two lines referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

NOTE 2 Although no directional sense is associated with instances of **sketch_create_curve_element**, the lines to be filleted by this operation have been selected by the user, and the selection mechanism provided in ISO 10303-55 ensures that explicit representations are available for them, including the required sense information.

The additional BOOLEAN-valued attribute **trim_option** determines whether or not the chamfered lines are to be trimmed at chamfer ends.

NOTE 3 The effect of the **trim_option** attribute is illustrated in Figure 21.

The attributes **first_chamfer_length** and **second_chamfer_length** determine the geometry of the chamfer, as described in the attribute definitions.

EXPRESS specification:

```
*)
ENTITY create_chamfer
  SUBTYPE OF (sketch_create_curve_element);
  first_line                    : line;
  to_right_of_first_line        : BOOLEAN;
  second_line                   : line;
  to_right_of_second_line       : BOOLEAN;
  first_chamfer_length          : positive_length_measure;
  second_chamfer_length         : OPTIONAL positive_length_measure;
  trim_option                   : BOOLEAN;
WHERE
  WR1: first_line <> second_line;
  WR2: cross_product(first_line.dir, second_line.dir).magnitude <> 0.0;
END_ENTITY;
(*
```

Attribute definitions:

first_line: the first of the two intersecting lines.

to_right_of_first_line: TRUE if the chamfer is positioned at the right of the first line.

second_line: the second of the two intersecting lines.

to_right_of_second_line: TRUE if the chamfer is positioned at the right of the second line.

first_chamfer_length: the distance along the first chamfered line from one end point of the chamfer to the intersection point of the two lines.

second_chamfer_length: if specified, the distance along the second chamfered line from one end point of the chamfer to the intersection point of the two lines. If not specified the distance equal to **first_chamfer_length** is used.

trim_option: TRUE if the chamfered lines are trimmed at chamfer ends after the creation of the chamfer and FALSE otherwise.

Formal propositions:

WR1: The first line shall be different from the second line.

WR2: The two chamfered lines shall not be parallel.

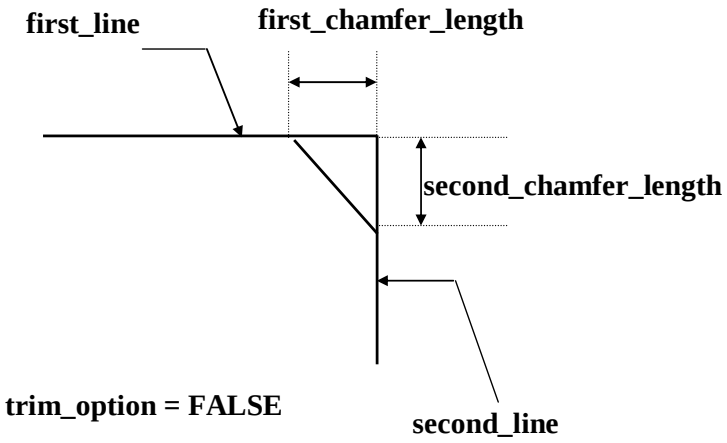


Figure 21 — create_chamfer

4.4.35 create_divided_curve

A **create_divided_curve** is a type of **sketch_create_curve_element** that divides a curve into two portions. It creates two separate curve instances. The point of division is specified as an instance of **point_on_curve** as defined in ISO 10303-42, and a WHERE rule is provided to ensure that the curve instance used in defining that point is the same as the curve instance being divided.

NOTE The curve and the point referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_divided_curve
  SUBTYPE OF (sketch_create_curve_element);
  selected_curve          : curve;
  division_point          : point_on_curve;
WHERE
  WR1: division_point.basis_curve :=: selected_curve;
END_ENTITY;
(*
```

Attribute definitions:

selected_curve: the target curve to be divided.

division_point: the point at which the curve is divided.

Formal propositions:

WR1: The instance referenced by the attribute **selected_curve** shall be same as the instance referenced by the attribute **basis_curve** of **division_point**.

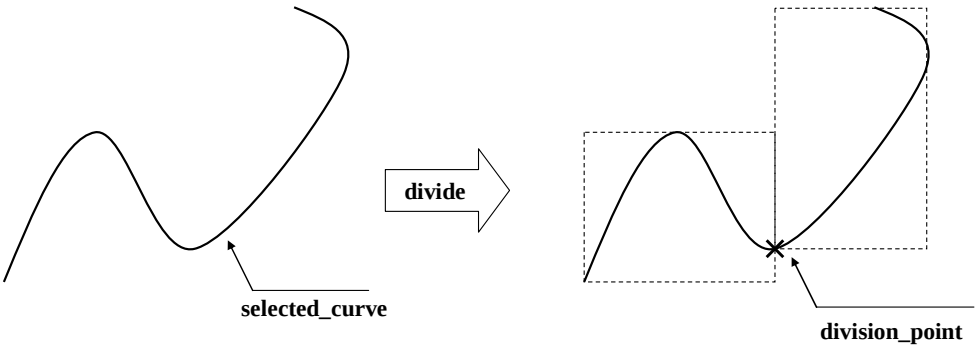


Figure 22 — sketch_divided_curve

4.4.36 sketch_operate_transform

A **sketch_operate_transform** is a type of **sketch_command** and **geometric_representation_item** that specifies types of sketch transformations. A transformation modifies the original geometry defined by the attribute **selected_objects**, by applying the specified transformation to it. The original geometry may be preserved as indicated by the value of the attribute **preserve_selected_objects**.

NOTE The geometric curve set referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY sketch_operate_transform
  ABSTRACT SUPERTYPE OF (ONEOF (sketch_transform_translate,
  sketch_transform_rotate, sketch_transform_mirror,
  sketch_transform_scale))
  SUBTYPE OF (sketch_command, geometric_representation_item);
  selected_objects              : geometric_curve_set;
  preserve_selected_objects     : BOOLEAN;
WHERE
  WR1: SELF\geometric_representation_item.dim = 2;
END_ENTITY;
(*
```

Attribute definitions:

selected_objects: the entities that are to be transformed.

preserve_selected_objects: TRUE if the selected objects are preserved in addition to the transformed geometry, and FALSE if only the transformed geometry is preserved.

Formal propositions:

WR1: The dimension of this entity shall be 2D.

4.4.37 sketch_transform_translate

A **sketch_transform_translate** is a type of **sketch_operate_transform** that applies a linear displacement to the selected objects in the plane of the sketch.

EXPRESS specification:

```
*)
ENTITY sketch_transform_translate
  SUBTYPE OF (sketch_operate_transform);
  displacement_x          : length_measure;
  displacement_y          : length_measure;
END_ENTITY;
(*
```

Attribute definitions:

displacement_x: the *x*-component of the displacement.

displacement_y: the *y*-component of the displacement.

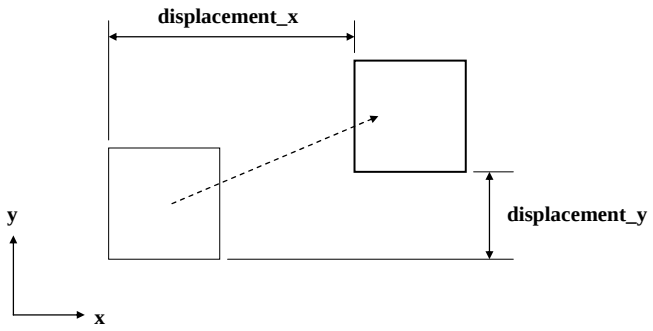


Figure 23 — sketch_transform_translate

4.4.38 sketch_transform_rotate

A **sketch_transform_rotate** is a type of **sketch_operate_transform** that rotates the selected objects in the plane of the sketch through a specified angle about a specified point.

NOTE The point referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY sketch_transform_rotate
  SUBTYPE OF (sketch_operate_transform);
  rotation_angle                : plane_angle_measure;
  rotation_centre               : cartesian_point;
  objects_rotation_direction    : OPTIONAL rotation_direction;
END_ENTITY;
(*
```

Attribute definitions:

rotation_angle: the magnitude of the rotation.

rotation_centre: the centre of the rotation.

objects_rotation_direction: the rotation direction of the entities to be rotated. The default direction is counter-clockwise.

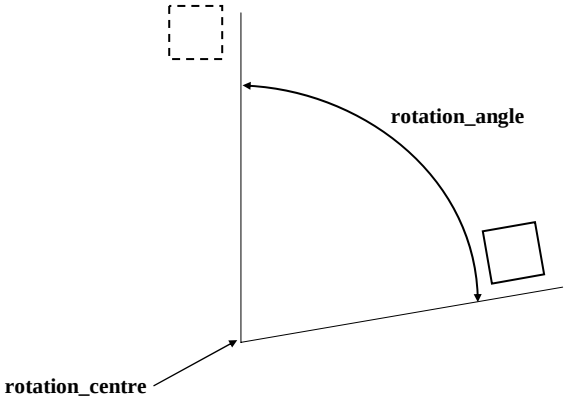


Figure 24 — sketch_transform_rotate

4.4.39 sketch_transform_mirror

A **sketch_transform_mirror** is a type of **sketch_operate_transform** that mirrors the selected objects about a specified line in the plane of the sketch.

NOTE The reference axis referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY sketch_transform_mirror
  SUBTYPE OF (sketch_operate_transform);
  reference_axis          : line_or_trimmed_line;
END_ENTITY;
(*
```

Attribute definitions:

reference_axis: the **line_or_trimmed_line** about which the selected objects are mirrored.

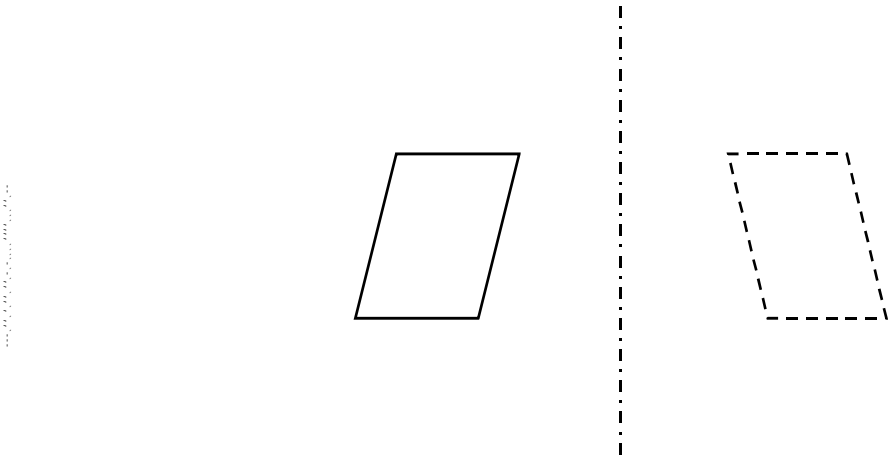


Figure 25 — sketch_transform_mirror

4.4.40 sketch_transform_scale

A **sketch_transform_scale** is a type of **sketch_operate_transform** that applies a scaling transformation to the selected objects, in the plane of the sketch. Provision is made for inhomogeneous scaling. The attribute **fixed_point** defines the fixed point of the transformation, i.e., the point in the sketch plane whose position is unchanged by the application of the transformation.

NOTE 1 The point referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY sketch_transform_scale
  SUBTYPE OF (sketch_operate_transform);
  scale_x                : positive_ratio_measure;
  scale_y                : positive_ratio_measure;
  fixed_point            : cartesian_point;
END_ENTITY;
(*
```

Attribute definitions:

scale_x: the scaling factor applied to the projected distance in the *x*-direction from the fixed point to the transformed point.

scale_y: the scaling factor applied to the projected distance in the *y*-direction from the fixed point to the transformed point.

fixed_point: the fixed point of the scaling transformation.

NOTE 2 The effect of scaling is illustrated in Figure 26. In the figure the *x* scaling factor is less than 1 and the *y* scaling factor greater than 1. The fixed point of the scaling is the origin.

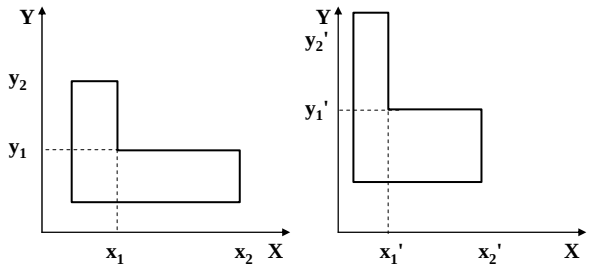


Figure 26 — sketch_transform_scale

4.4.41 sketch_create_pattern_element

A **sketch_create_pattern_element** is a type of **sketch_command** that specifies types of pattern creation. It is also a type of **geometric_curve_set** as defined in ISO 10303-42. The elements of the generated **geometric_curve_set** will contain a copy of **selected_objects.elements** together with transformed copies of **selected_objects.elements** corresponding to a specified pattern.

NOTE The **geometric_curve_set** referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY sketch_create_pattern_element
   ABSTRACT SUPERTYPE OF (ONEOF (create_pattern_rectangular,
   create_pattern_circular))
   SUBTYPE OF (sketch_command, geometric_curve_set);
   selected_objects        : geometric_curve_set;
WHERE
   WR1: NOT(have_pattern_element_in_geometric_curve_set(selected_objects));
   WR2: SELF\geometric_representation_item.dim = 2;
END_ENTITY;
(*
```

Attribute definitions:

selected_objects: the points and curves from which the pattern is to be created.

Formal propositions:

WR1: The attribute **selected_objects** shall not reference any instance of **sketch_create_pattern_-element**.

WR2: The dimension of this entity shall be 2D.

4.4.42 create_pattern_rectangular

A **create_pattern_rectangular** is a type of **sketch_create_pattern_element** that generates multiple copies of a selected set of sketch elements in locations defined by a specified rectangular pattern. The copied sketch elements retain their original orientation in the resulting pattern.

EXPRESS specification:

```
*)
ENTITY create_pattern_rectangular
  SUBTYPE OF (sketch_create_pattern_element);
  row_direction                 : direction;
  row_number                    : INTEGER;
  column_number                 : INTEGER;
  row_spacing                   : positive_length_measure;
  column_spacing                : positive_length_measure;
  column_augmentation_direction  : OPTIONAL rotation_direction;
WHERE
  WR1: (row_number > 0) AND (column_number > 0);
END_ENTITY;
(*
```

Attribute definitions:

row_direction: the row direction of the pattern.

row_number: the number of rows in the pattern.

column_number: the number of columns in the pattern.

row_spacing: the spacing between rows.

column_spacing: the spacing between columns.

column_augmentation_direction: the direction of column augmentation with respect to the **row_direction**. It shall be counterclockwise or clockwise. The default direction is counterclockwise.

Formal propositions:

WR1: The number of rows and columns shall be positive.

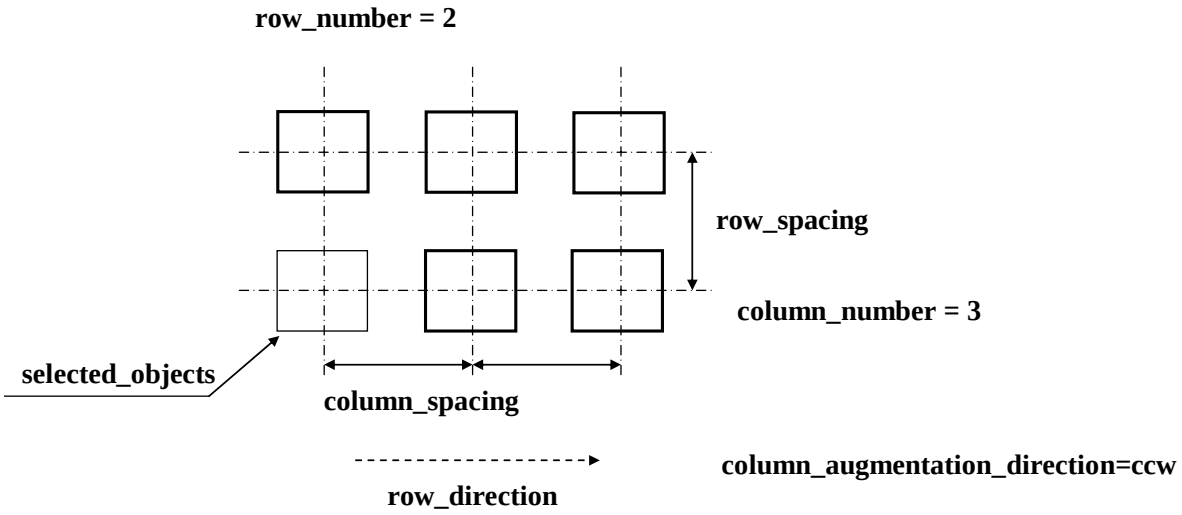


Figure 27 — create_pattern_rectangular

4.4.43 create_pattern_circular

A **create_pattern_circular** is a type of **sketch_create_pattern_element** that generates multiple copies of a selected set of sketch elements in locations defined by a specified circular pattern.

NOTE The two points referred to in the definition of this entity shall be explicitly represented, through the use of the ISO 10303-55 selection mechanism.

EXPRESS specification:

```
*)
ENTITY create_pattern_circular
  SUBTYPE OF (sketch_create_pattern_element);
  number_of_replicates    : INTEGER;
  angle_increment         : plane_angle_measure;
  centre_point            : cartesian_point;
  radial_alignment        : BOOLEAN;
  reference_point         : cartesian_point;
WHERE
  WR1: number_of_replicates > 0;
END_ENTITY;
(*
```

Attribute definitions:

number_of_replicates: the number of the elements in the pattern, not including the original.

angle_increment: the angular increment between successive replicates.

centre_point: the centre of the rotation.

radial_alignment: specifies whether elements are rotated or not. If TRUE, the replicates are individually rotated about the centre of rotation; if FALSE, each point in the pattern has the same translation as the reference point, so that the orientation of the original **selected_objects** is maintained.

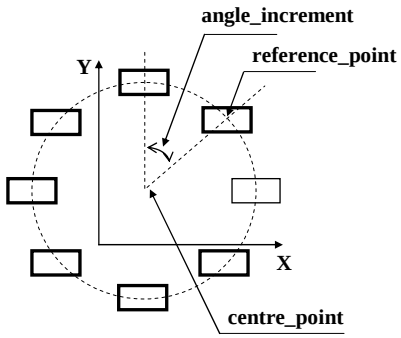
reference_point: a point associated with the **selected_objects**, whose distance from the **centre_point** determines the radius of the circular pattern. Each replicate has a corresponding reference point on the circle at successive multiples of **angle_increment** with respect to the line joining the **centre_point** to the original **reference_point**.

Formal propositions:

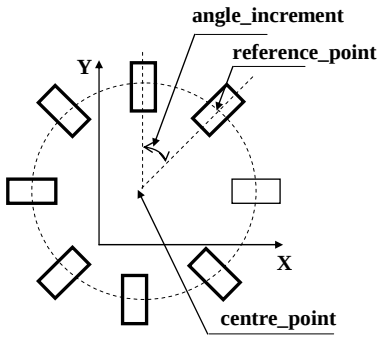
WR1: The number of the elements in the pattern shall be positive.

Informal Propositions

IP1: The product of (**number_of_replicates** + 1) and the value of **angle_increment** shall not exceed 360 degrees.



radial_alignment = FALSE



radial_alignment = TRUE

Figure 28 — create_pattern_circular

4.5 procedural_sketch function definitions

4.5.1 distance_between_cartesian_points

The function **distance_between_cartesian_points** returns a value of type **length_measure** which is the distance between two specified instances of **cartesian_point**.

EXPRESS specification:

```
*)
FUNCTION distance_between_cartesian_points(pt1, pt2 : cartesian_point) :
length_measure;

LOCAL
  temp_dis : length_measure;
END_LOCAL;

  temp_dis := SQRT ( ((pt1.coordinates[1]-pt2.coordinates[1])**2)
                   +((pt1.coordinates[2]-pt2.coordinates[2])**2) );

  RETURN(temp_dis);

END_FUNCTION;
(*
```

Argument definitions:

pt1: (input) the first point.

pt2: (input) the second point.

temp_dis: (output) The distance between the two points.

4.5.2 non_collinear_2d_points

The function **non_collinear_2d_points** checks whether three 2D **cartesian_point**s are collinear or not.

EXPRESS specification:

```
*)
FUNCTION non_collinear_2d_points(pt1, pt2, pt3 : cartesian_point) :
BOOLEAN;

  IF ((pt1.dim =2) AND (pt2.dim =2) AND (pt3.dim =2)) THEN
    IF (three_distinct_points(pt1, pt2, pt3)) THEN
      IF ((pt2.coordinates[2]-pt1.coordinates[2])*(pt3.coordinates[1]-
        pt1.coordinates[1]) <> (pt3.coordinates[2]-pt1.coordinates[2])*
        (pt2.coordinates[1]-pt1.coordinates[1])) THEN
        RETURN (TRUE);
      END_IF;
    END_IF;
  END_IF;

  RETURN (FALSE);
```



```
END_FUNCTION;
(*
```

Argument definitions:

pt1: (input) the first of three points to be tested for collinearity.

pt2: (input) the second of three points to be tested for collinearity.

pt3: (input) the third of three points to be tested for collinearity.

4.5.3 midpoint

The function **midpoint** returns the **cartesian_point** which is the midpoint of two specified **cartesian_point**s.

EXPRESS specification:

```
*)
FUNCTION midpoint(pt1, pt2 : cartesian_point) : cartesian_point;

LOCAL
  temp_pt : cartesian_point;
END_LOCAL;

  temp_pt.coordinates[1] := (pt1.coordinates[1] + pt2.coordinates[1]) / 2;
  temp_pt.coordinates[2] := (pt1.coordinates[2] + pt2.coordinates[2]) / 2;

  RETURN (temp_pt);

END_FUNCTION;
(*
```

Argument definitions:

pt1, pt2: (input) The two cartesian points whose midpoint is to be calculated.

temp_pt: (output) The midpoint of the two input cartesian points.

4.5.4 distinct_points

The function **distinct_points** checks whether any instance of **cartesian_point** in a given list of three or more such instances has the same position as the next instance in the list.

EXPRESS specification:

```
*)
FUNCTION distinct_points(pts : LIST[3:?] OF cartesian_point) : BOOLEAN;

  REPEAT index := LOINDEX(pts) TO HIINDEX(pts) - 1;
    IF (distance_between_cartesian_points(pts[index], pts[index + 1])
    = 0.0) THEN
      RETURN (FALSE);
    END_IF;
  END_REPEAT;

  RETURN (TRUE);
```

```
END_FUNCTION;
(*
```

Argument definitions:

pts: (input) The given list of cartesian points.

4.5.5 circular_type

The function **circular_type** checks whether any instance of the select data type **circle_or_circular_arc** is a **circle** or a **trimmed_curve** referencing a **circle**.

EXPRESS specification:

```
*)
FUNCTION circular_type(tr : circle_or_circular_arc) : BOOLEAN;

  IF ('GEOMETRY_SCHEMA.TRIMMED_CURVE' IN TYPEOF(tr)) THEN
    IF ('GEOMETRY_SCHEMA.CIRCLE' IN TYPEOF
    (tr\trimmed_curve.basis_curve)) THEN
      RETURN (TRUE);
    ELSE
      RETURN (FALSE);
    END_IF;
  END_IF;

  RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

tr: (input) The given circle or circular arc.

4.5.6 linear_type

The function **linear_type** checks whether any instance of the select data type **line_or_trimmed_line** is a **line** or a **trimmed_curve** referencing a **line**.

EXPRESS specification:

```
*)
FUNCTION linear_type(tr : line_or_trimmed_line) : BOOLEAN;

  IF ('GEOMETRY_SCHEMA.TRIMMED_CURVE' IN TYPEOF(tr)) THEN
    IF ('GEOMETRY_SCHEMA.LINE' IN TYPEOF
    (tr\trimmed_curve.basis_curve)) THEN
      RETURN (TRUE);
    ELSE
      RETURN (FALSE);
    END_IF;
  END_IF;

  RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

tr: (input) The given line or trimmed line.

4.5.7 centre_of_circle_or_circular_arc

The function **centre_of_circle_or_circular_arc** returns the **cartesian_point** which is the centre point of a **circle_or_circular_arc** instance.

EXPRESS specification:

```
*)
FUNCTION centre_of_circle_or_circular_arc(coca :  circle_or_circular_arc) :
cartesian_point;

  IF ('GEOMETRY_SCHEMA.TRIMMED_CURVE' IN TYPEOF(coca)) THEN
    RETURN (coca\trimmed_curve.basis_curve\circle\conic.position.location);
  ELSE
    RETURN (coca\circle\conic.position.location);
  END_IF;

END_FUNCTION;
(*
```

Argument definitions:

coca: (input) The given circle or circular arc

4.5.8 have_pattern_elements_in_geometric_curve_set

The function have**_pattern_elements_in_geometric_curve_set** checks whether a **geometric_curve-_set** contains any instance of type **sketch_create_pattern_element**.

EXPRESS specification:

```
*)
FUNCTION have_pattern_element_in_geometric_curve_set
(cs : geometric_curve_set) : BOOLEAN;

  REPEAT index := LOINDEX(cs\geometric_set.elements)
         TO HIINDEX(cs\geometric_set.elements);
    IF ('PROCEDURAL_SKETCH_SCHEMA.SKETCH_CREATE_PATTERN_ELEMENT' IN
       TYPEOF(cs\geometric_set.elements[index])) THEN
       RETURN (TRUE);
    END_IF;
  END_REPEAT;

  RETURN (FALSE);

END_FUNCTION;
(*
```

Argument definitions:

cs: (input) The **geometric_curve_set** to be tested.

4.5.9 three_distinct_points

The function **three_distinct_points** checks whether no two of any set of three cartesian points have the same position.

EXPRESS specification:

```
*)
FUNCTION three_distinct_points(pt1, pt2, pt3 : cartesian_point) : BOOLEAN;

  IF ((distance_between_cartesian_points(pt1, pt2) <> 0.0) AND
      (distance_between_cartesian_points(pt2, pt3) <> 0.0) AND
      (distance_between_cartesian_points(pt3, pt1) <> 0.0)) THEN
    RETURN (TRUE);
  END_IF;

  RETURN (FALSE);

END_FUNCTION;
(*
```

Argument definitions:

pt1, pt2, pt3: (input) The given cartesian points to be checked.

EXPRESS specification:

```
*)
END_SCHEMA; -- procedural_sketch_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 data type names	Short names
create_chamfer	**CRTCHM**
create_centreline	**CRTCNT**
create_circle	**CRTCRC**
create_circle_centre_point	**CCCP**
create_circle_concentric	**CRCRCN**
create_circle_2_points	**CC2P**
create_circle_3_points	**CC3P**
create_circle_3_tangents	**CC3T**
create_circular_arc	**CRCRAR**
create_circular_arc_angles	**CCAA**
create_circular_arc_centre_ends	**CCACE**
create_circular_arc_concentric	**CCAC**
create_circular_arc_start_centre_angle	**CCASCA**
create_circular_arc_start_centre_length	**CCASCL**
create_circular_arc_start_end_angle	**CCASEA**
create_circular_arc_start_end_direction	**CCASED**
create_circular_arc_start_end_radius	**CCASER**
create_circular_arc_3_points	**CCA3P**
create_circular_arc_3_tangents	**CCA3T**
create_divided_curve	**CRDVCR**
create_ellipse	**CRTELL**
create_ellipse_centre_point	**CECP**
create_ellipse_3_points	**CE3P**
create_fillet	**CRTFLL**
create_line_segment	**CRLNSG**
create_line_segment_point_tangent	**CLSPT**
create_line_segment_2_points	**CLS2P**
create_line_segment_2_tangents	**CLS2T**
create_parabolic_arc	**CRPRAR**
create_pattern_circular	**CRPTCR**
create_pattern_rectangular	**CRPTRC**
create_polygon	**CRT0**
create_polyline	**CRTPLY**
create_rectangle	**CRTRCT**
create_spline	**CRTSPL**
sketch_command	**SKTCMM**

Table A.1 (*continued*)

Entity data type names	Short names
sketch_create_curve_element	**SCCE**
sketch_create_pattern_element	**SCPE**
sketch_operate_transform	**SKOPTR**
sketch_transform_mirror	**SKTRMR**
sketch_transform_rotate	**SKTRRT**
sketch_transform_scale	**SKTRSC**
sketch_transform_translate	**SKTRTR**

Annex B
(normative)

Information object registration

B.1 Document identification

To provide for unambiguous identification of an information object in an open system, the object identifier

```
{ iso standard 10303 part(112) version(1) }
```

is assigned to this part of ISO 10303. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

B.2 Schema identification

To provide for unambiguous identification of the procedural-sketch-parametric schema in an open information system, the object identifier

```
{  iso standard 10303 part(112) version(1) schema(1)
          procedural-sketch-schema(1)  }
```

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

Annex C
(informative)

Computer interpretable listings

This annex references a listing of the EXPRESS entity data type names and corresponding short names as specified in this part of ISO 10303. It also references a listing of each EXPRESS schema specified in this part of ISO 10303, without comments or other explanatory text. These listings are available in computer-interpretable form, and can be found at the following URLs:

Short names:

`http://www.tc184-sc4.org/Short_Names/`

EXPRESS:

`http://www.tc184-sc4.org/EXPRESS/`

If there is difficulty accessing these sites contact ISO Central Secretariat or contact the ISO TC184/SC4 Secretariat directly at `sc4sec@tc184-sc4.org`.

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

Annex D
(informative)

EXPRESS-G diagrams

The diagrams in this annex correspond to the EXPRESS schemas specified in this part of ISO 10303. The diagrams use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in ISO 10303-11:2004, Annex D.

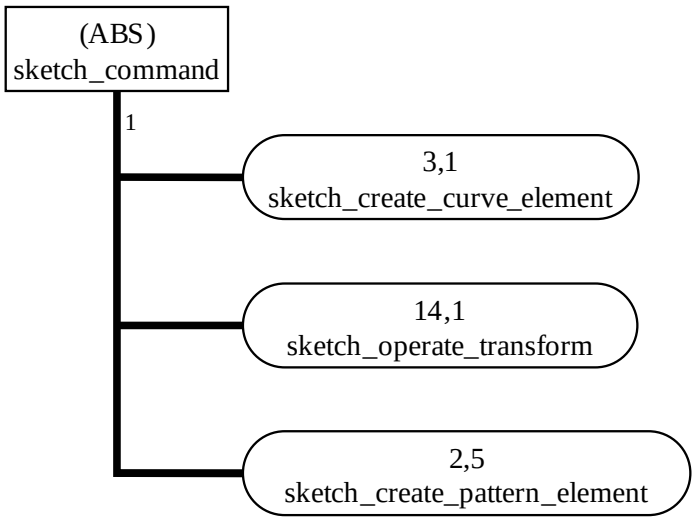


Figure D. 1 — procedural_sketch_schema EXPRESS-G diagram 1 of 16

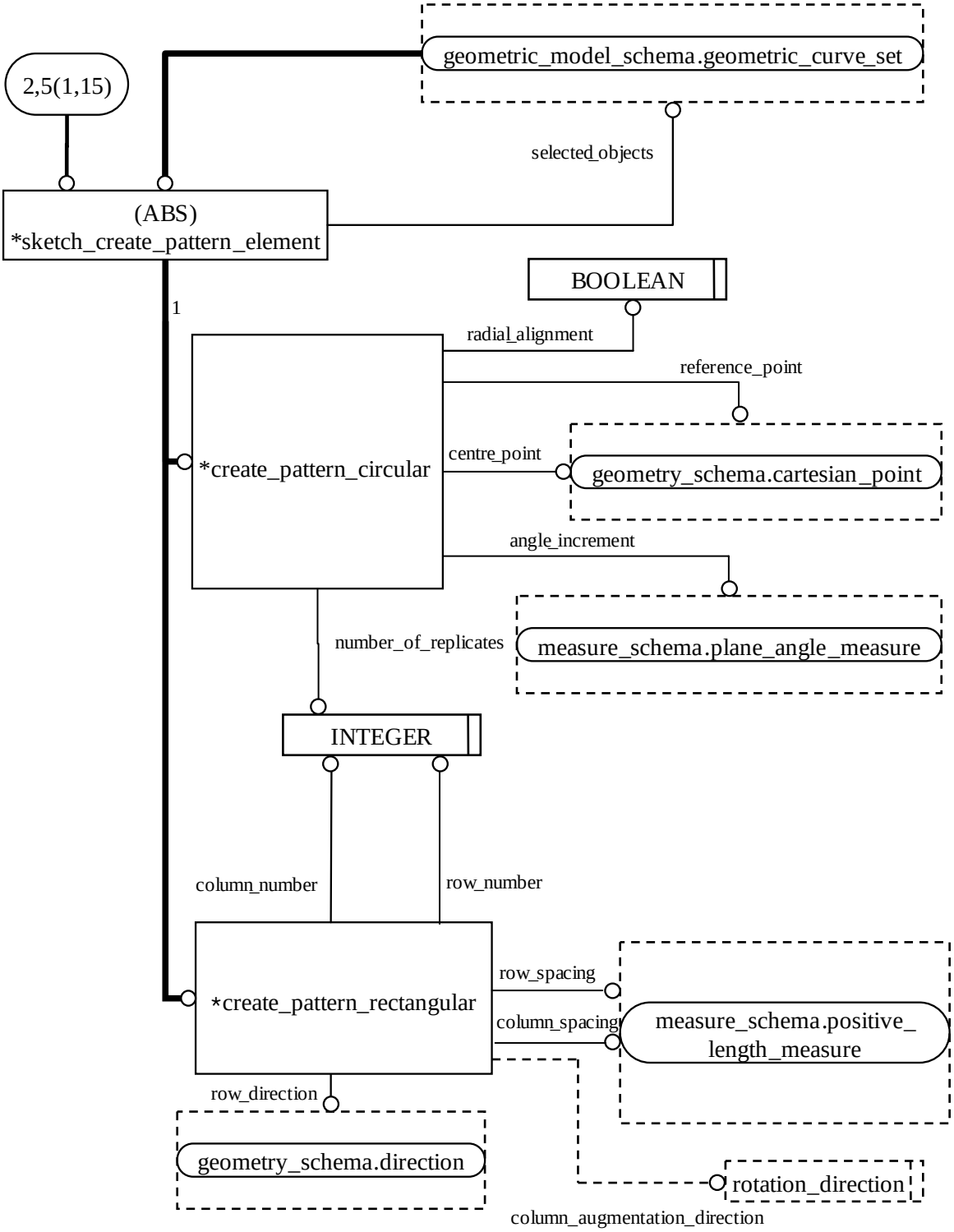


Figure D. 2 — procedural_sketch_schema EXPRESS-G diagram 2 of 16

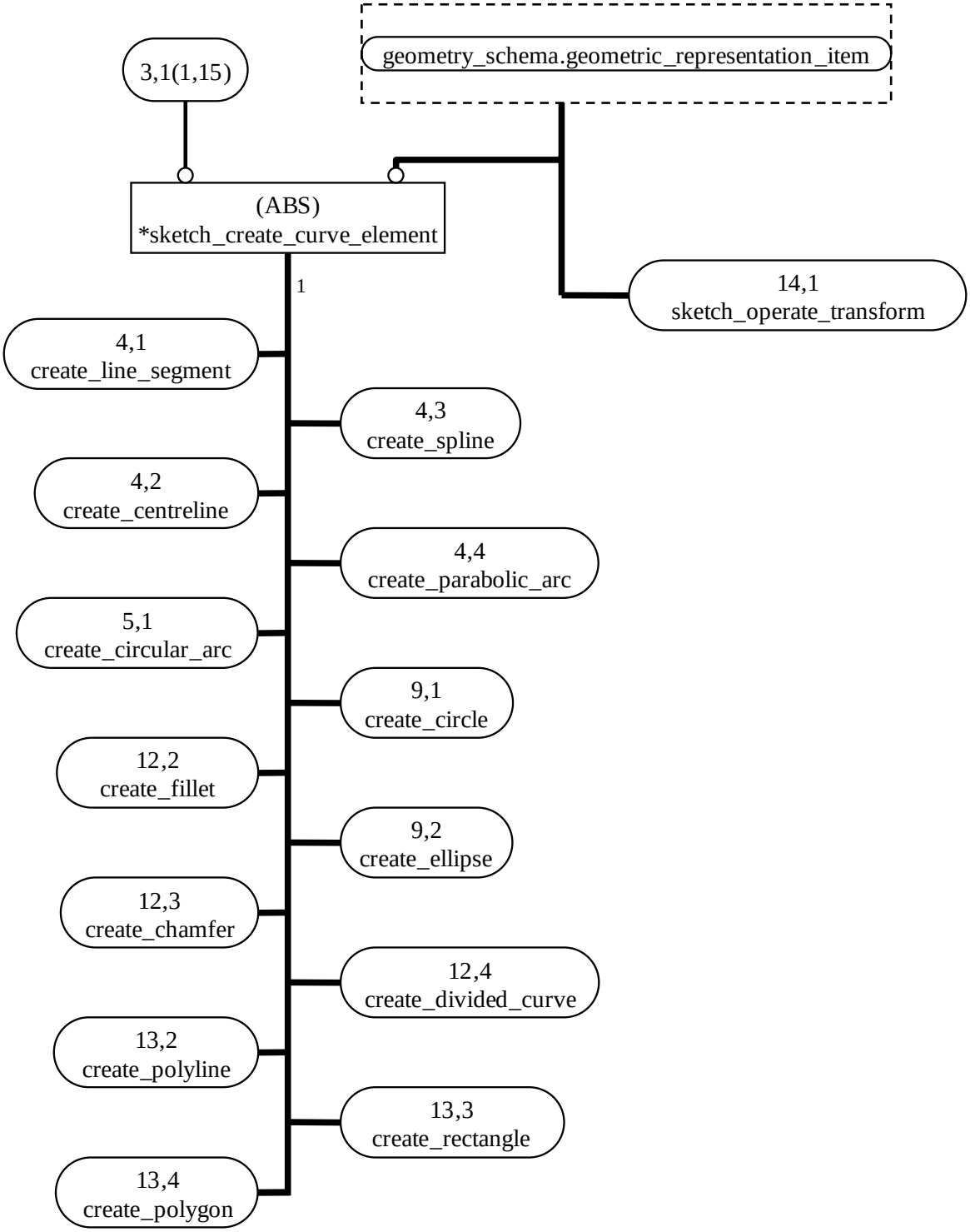


Figure D. 3 — procedural_sketch_schema EXPRESS-G diagram 3 of 16

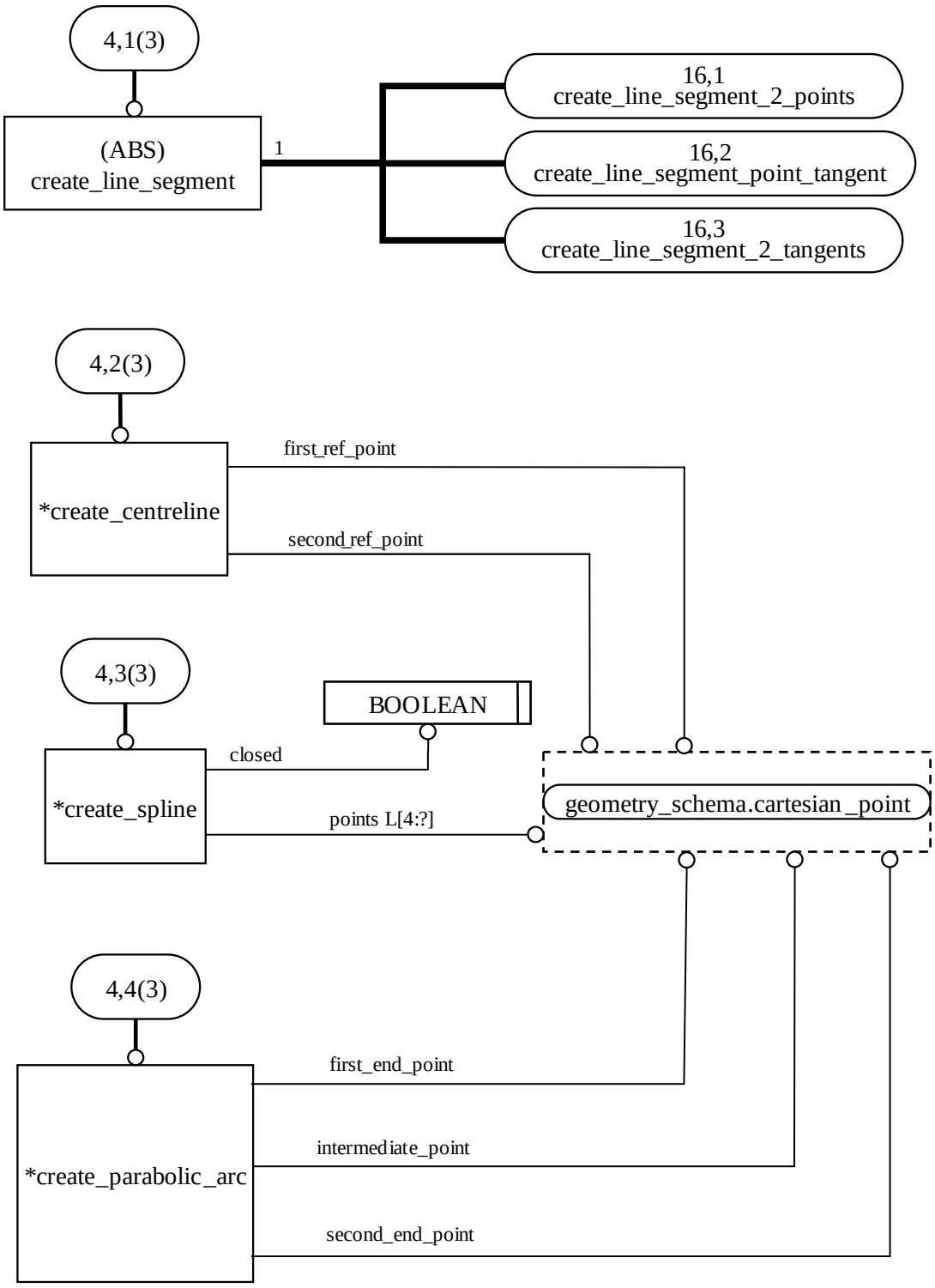


Figure D. 4 — procedural_sketch_schema EXPRESS-G diagram 4 of 16

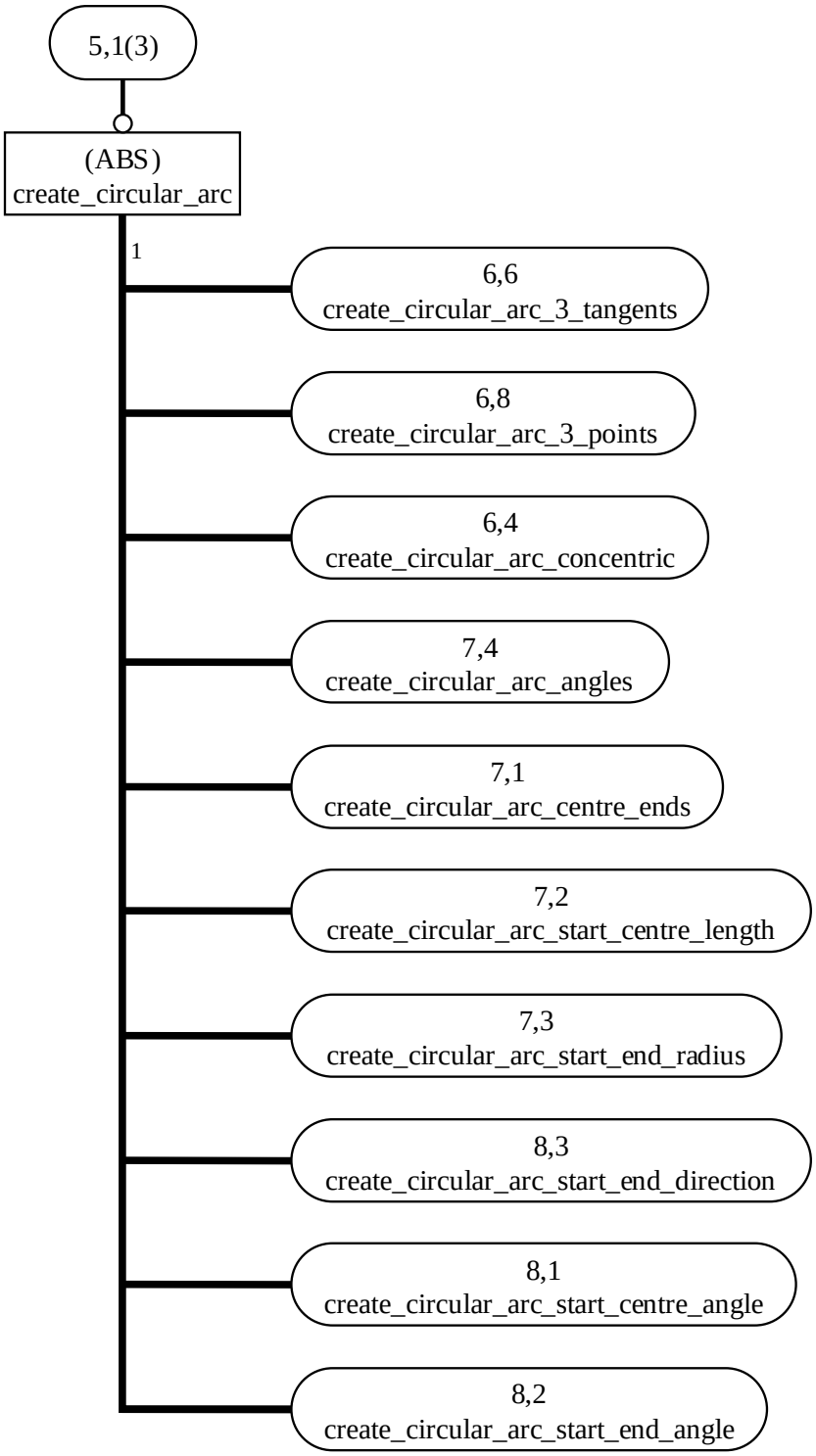


Figure D. 5 — procedural_sketch_schema EXPRESS-G diagram 5 of 16

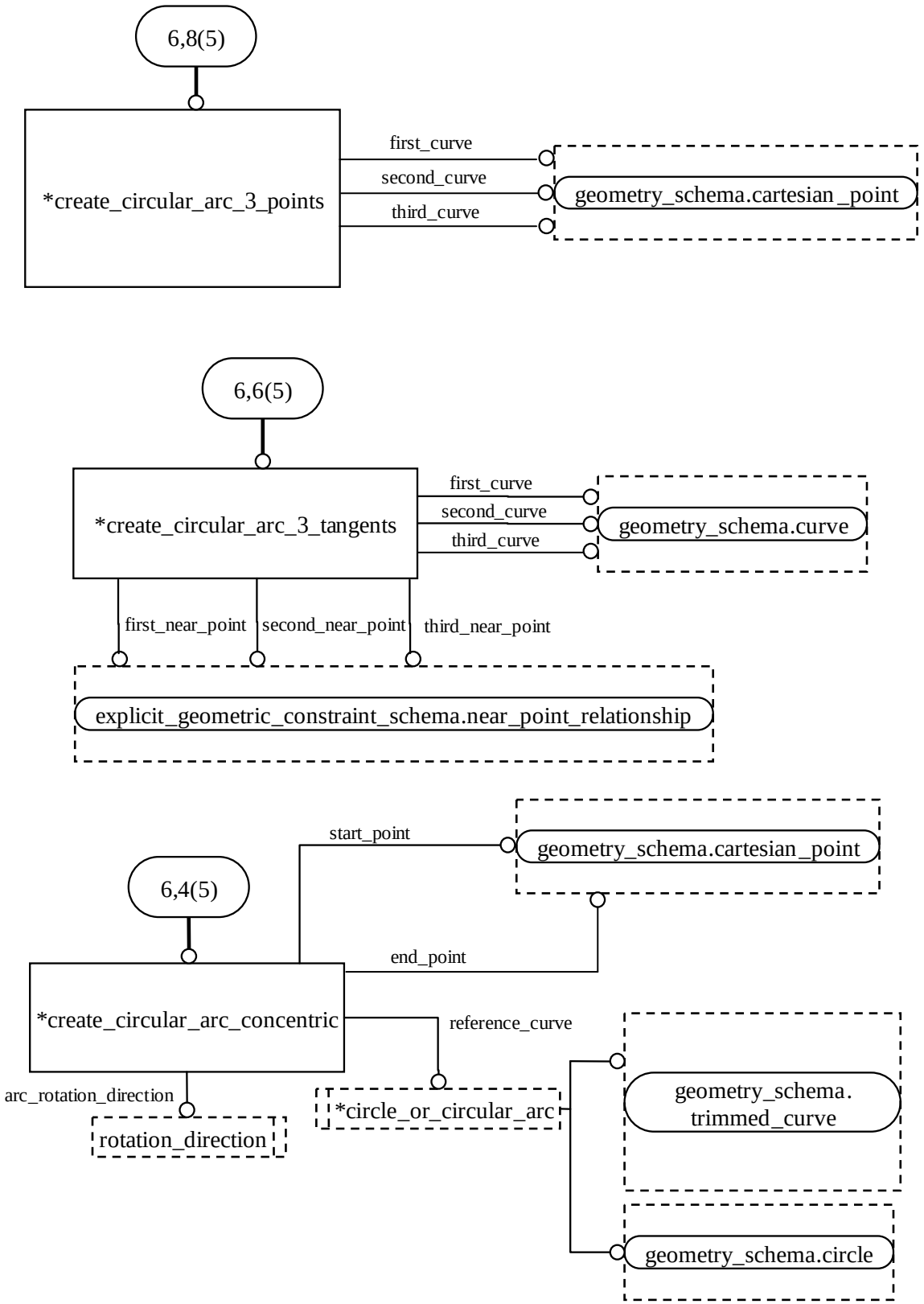


Figure D. 6 — procedural_sketch_schema EXPRESS-G diagram 6 of 16

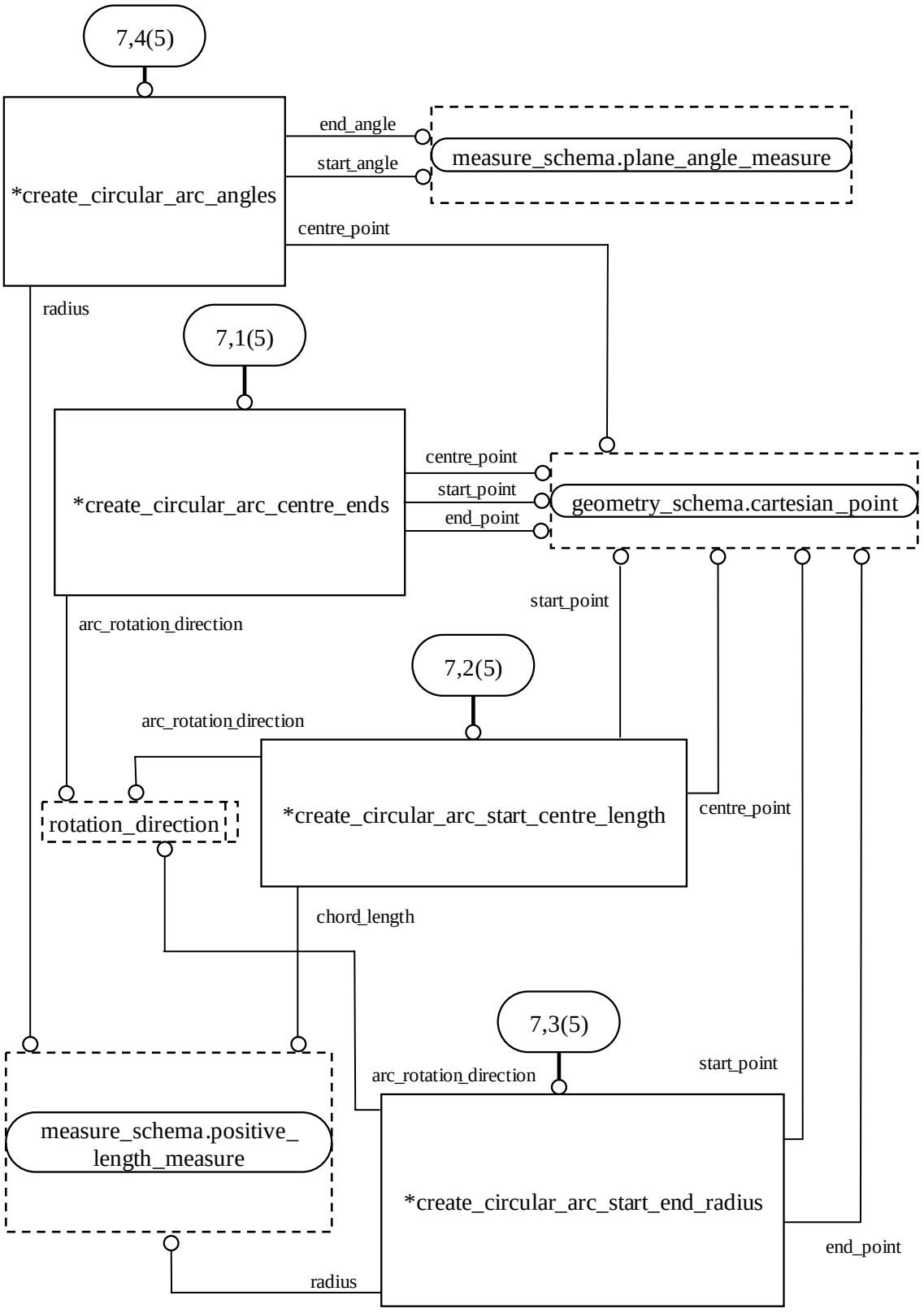


Figure D. 7 — procedural_sketch_schema EXPRESS-G diagram 7 of 16

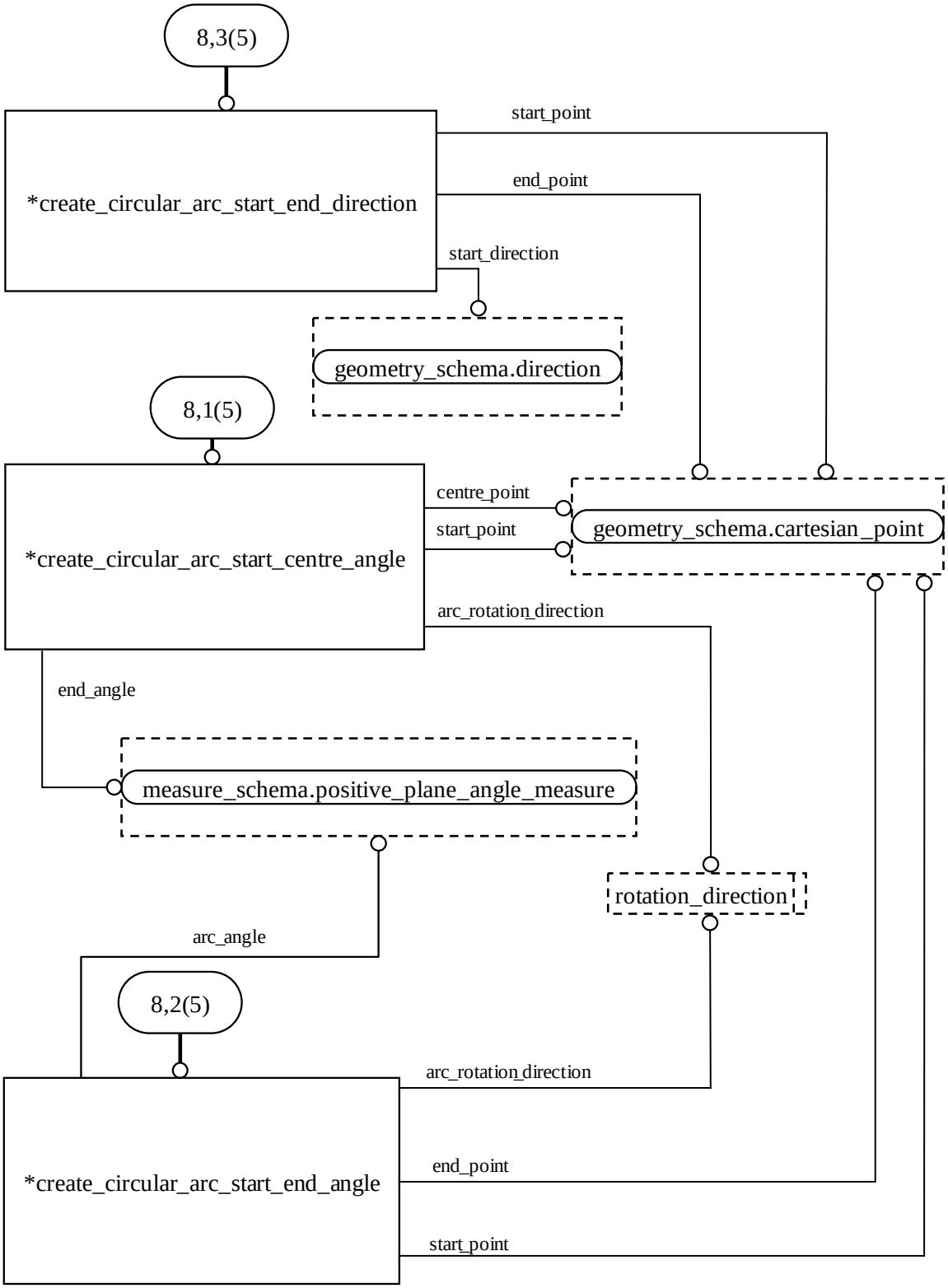


Figure D. 8 — procedural_sketch_schema EXPRESS-G diagram 8 of 16

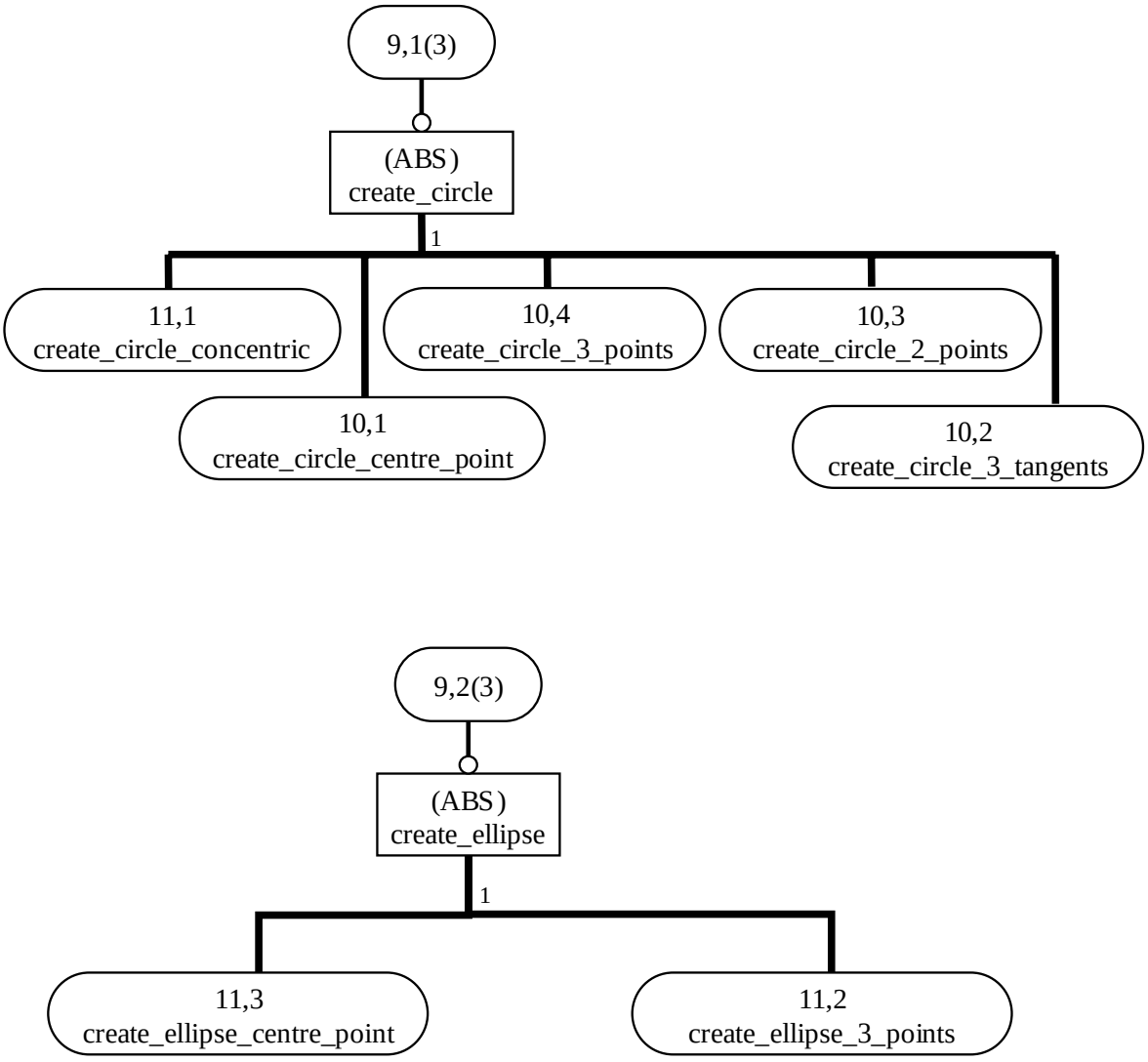


Figure D. 9 — procedural_sketch_schema EXPRESS-G diagram 9 of 16

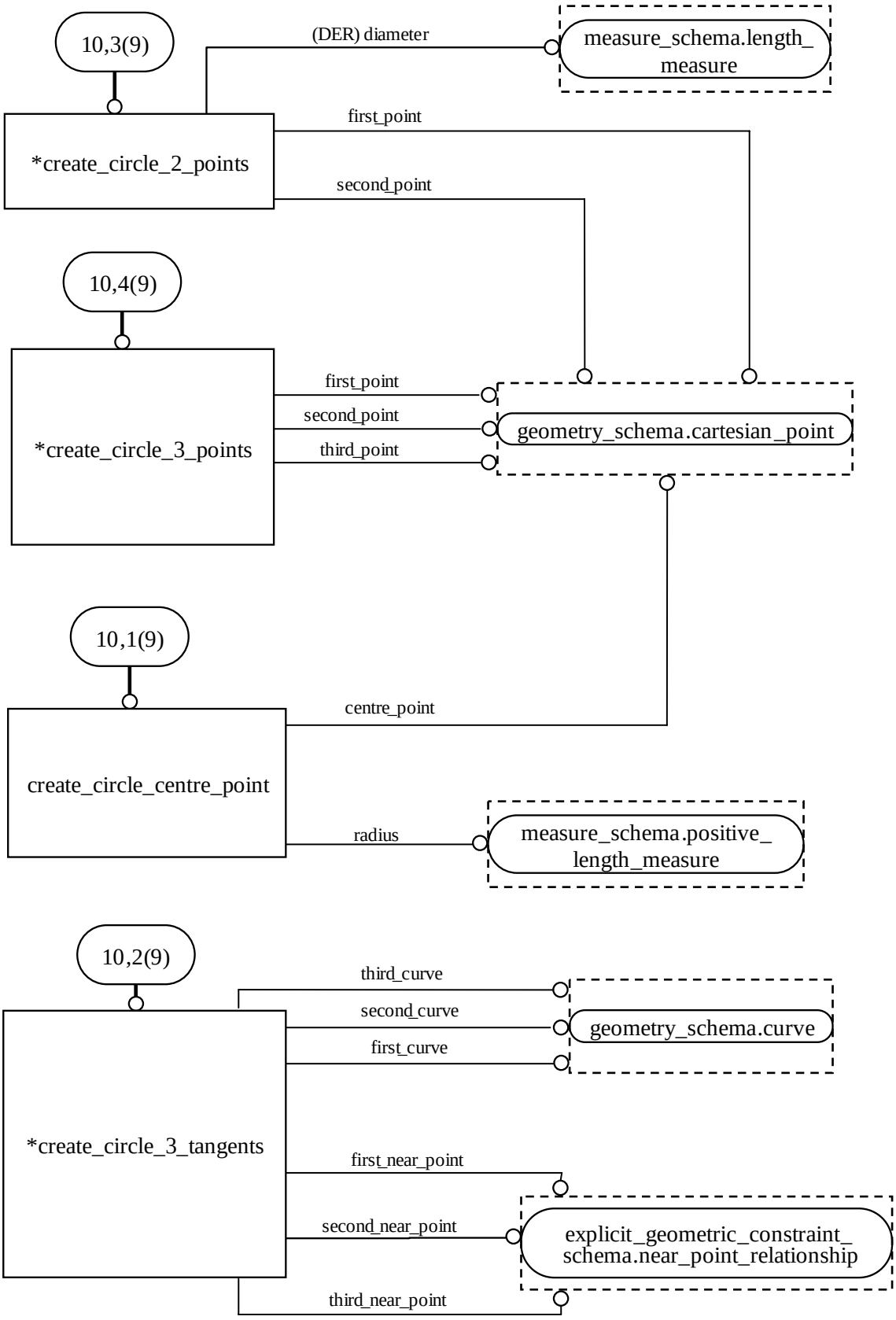


Figure D. 10 — procedural_sketch_schema EXPRESS-G diagram 10 of 16

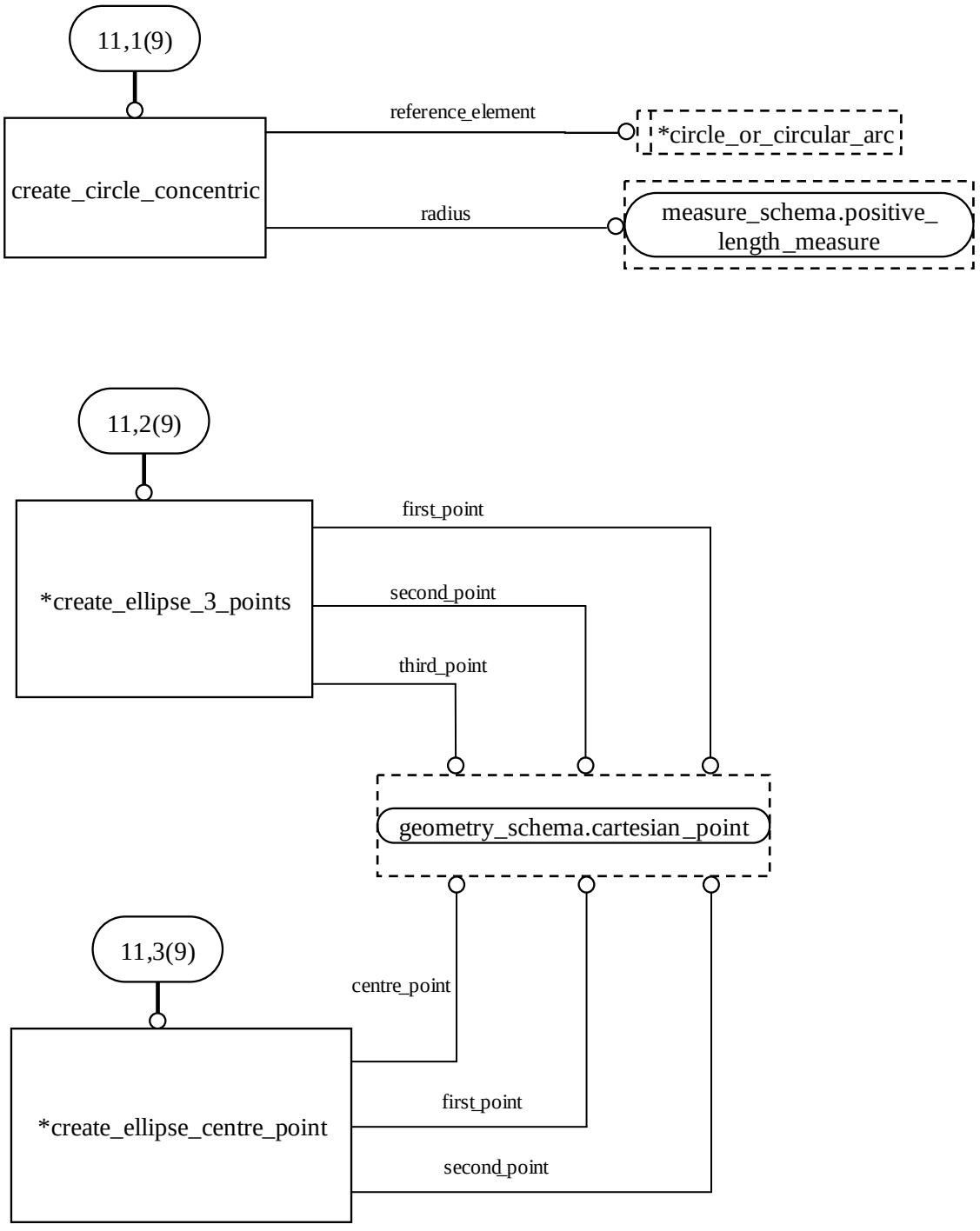


Figure D. 11 — procedural_sketch_schema EXPRESS-G diagram 11 of 16

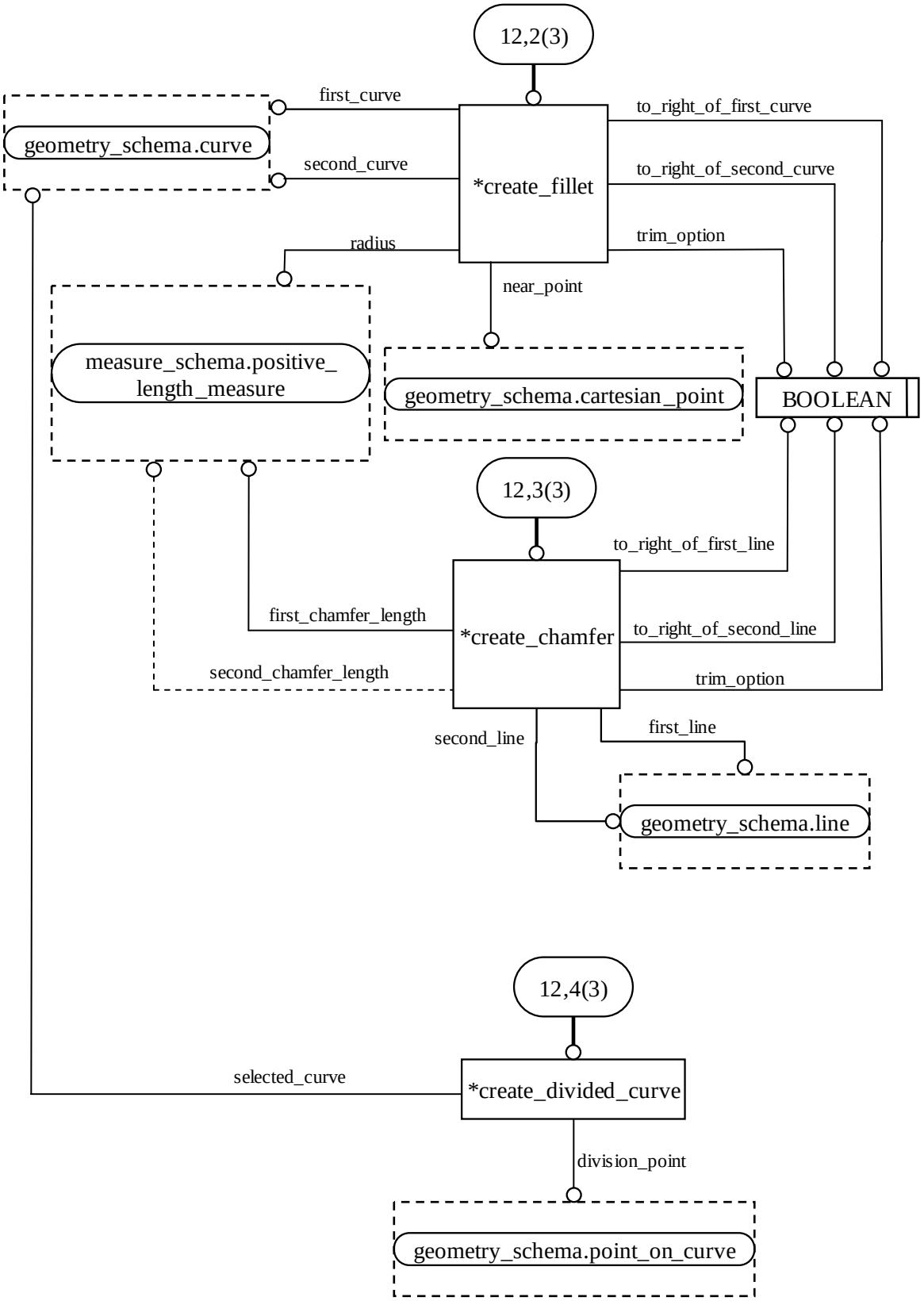


Figure D. 12 — procedural_sketch_schema EXPRESS-G diagram 12 of 16

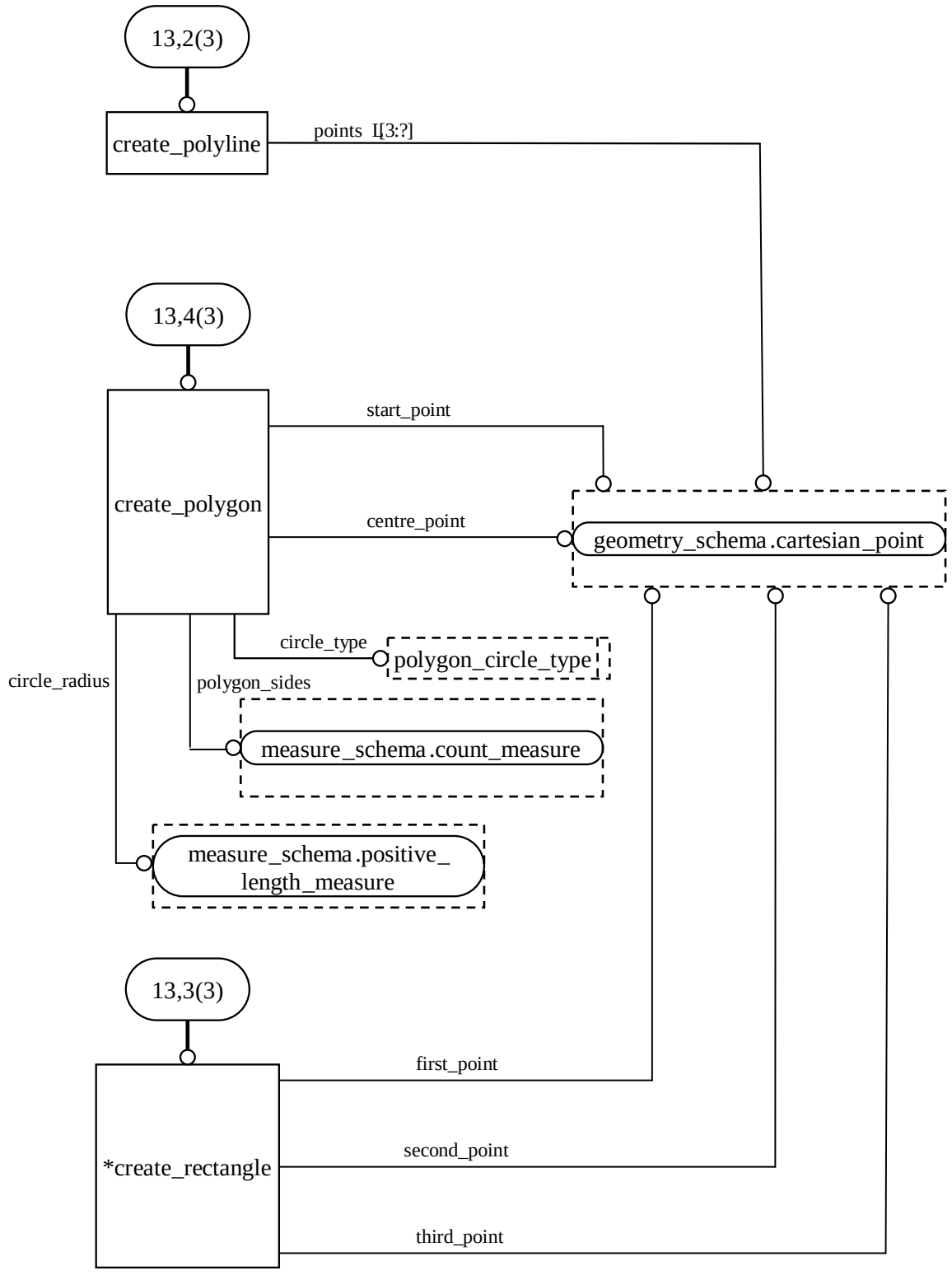


Figure D. 13 — procedural_sketch_schema EXPRESS-G diagram 13 of 16

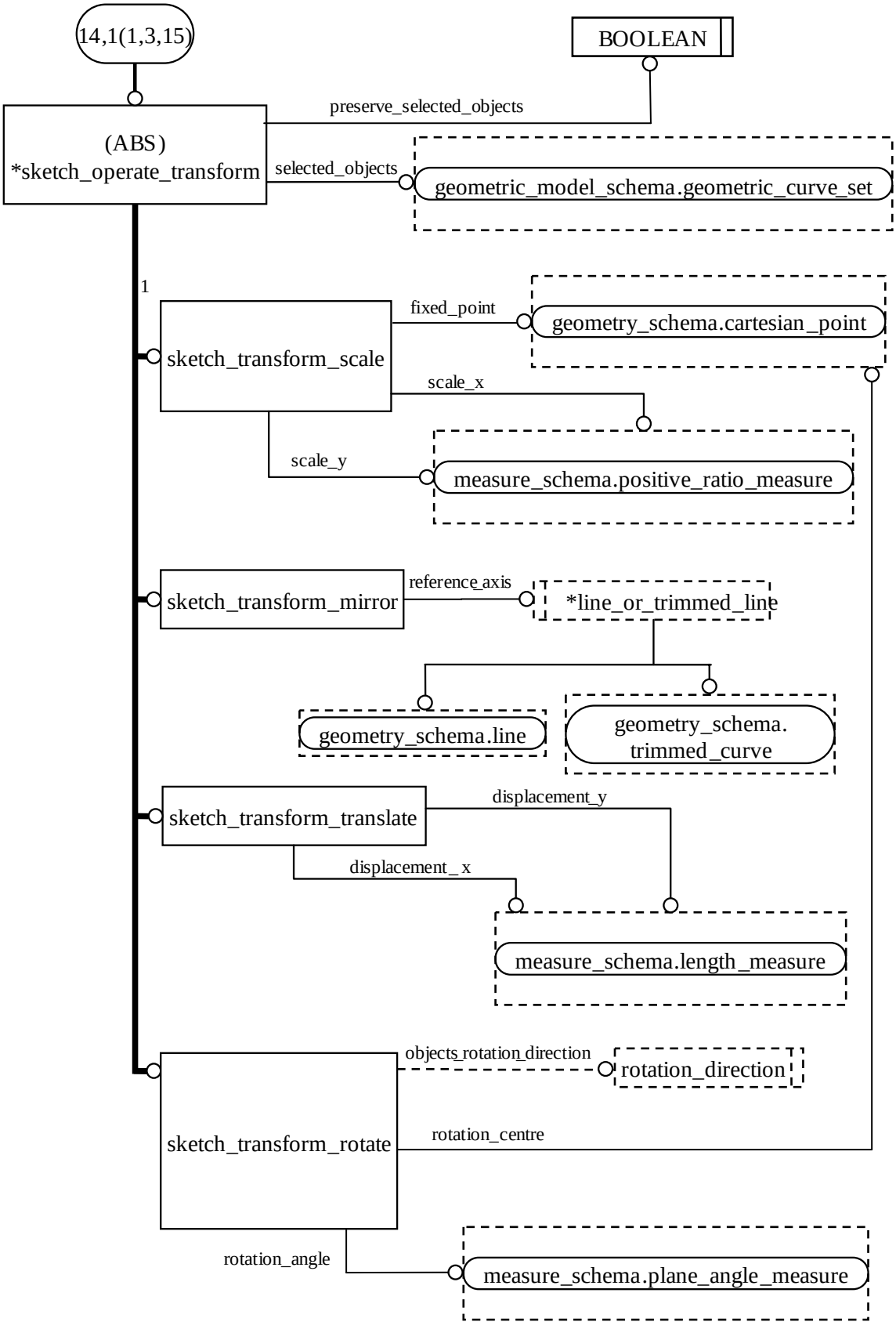


Figure D. 14 — procedural_sketch_schema EXPRESS-G diagram 14 of 16

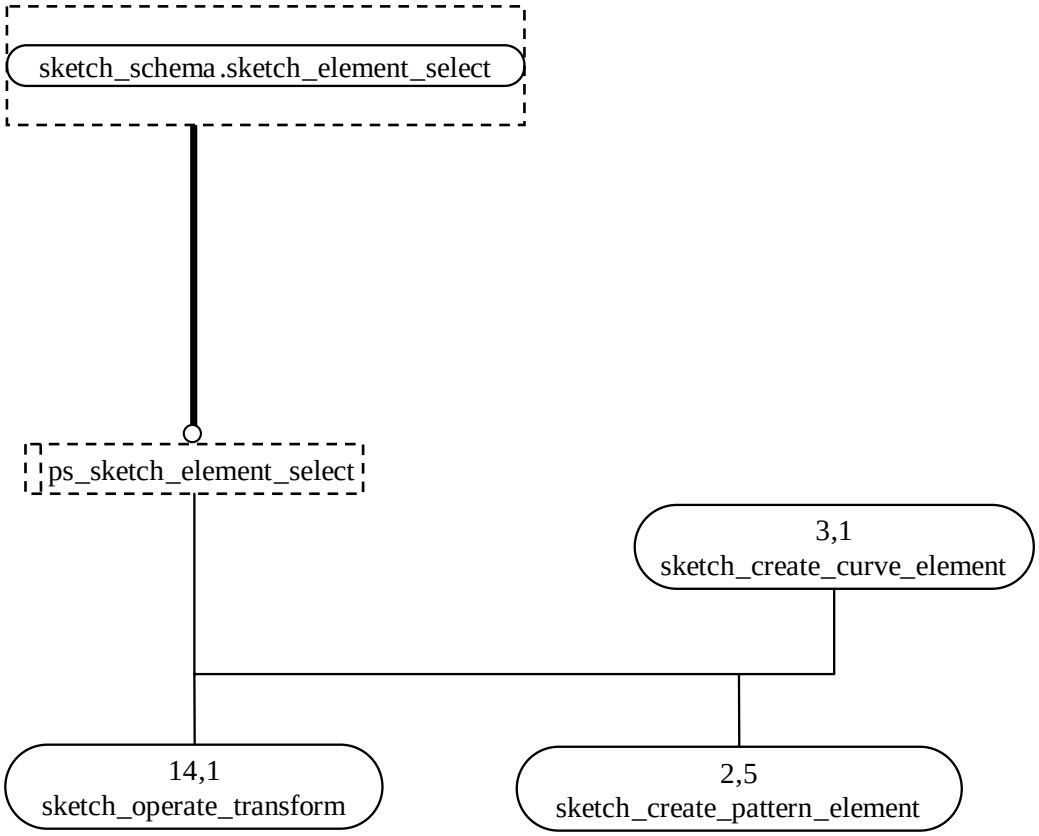


Figure D. 15 — procedural_sketch_schema EXPRESS-G diagram 15 of 16

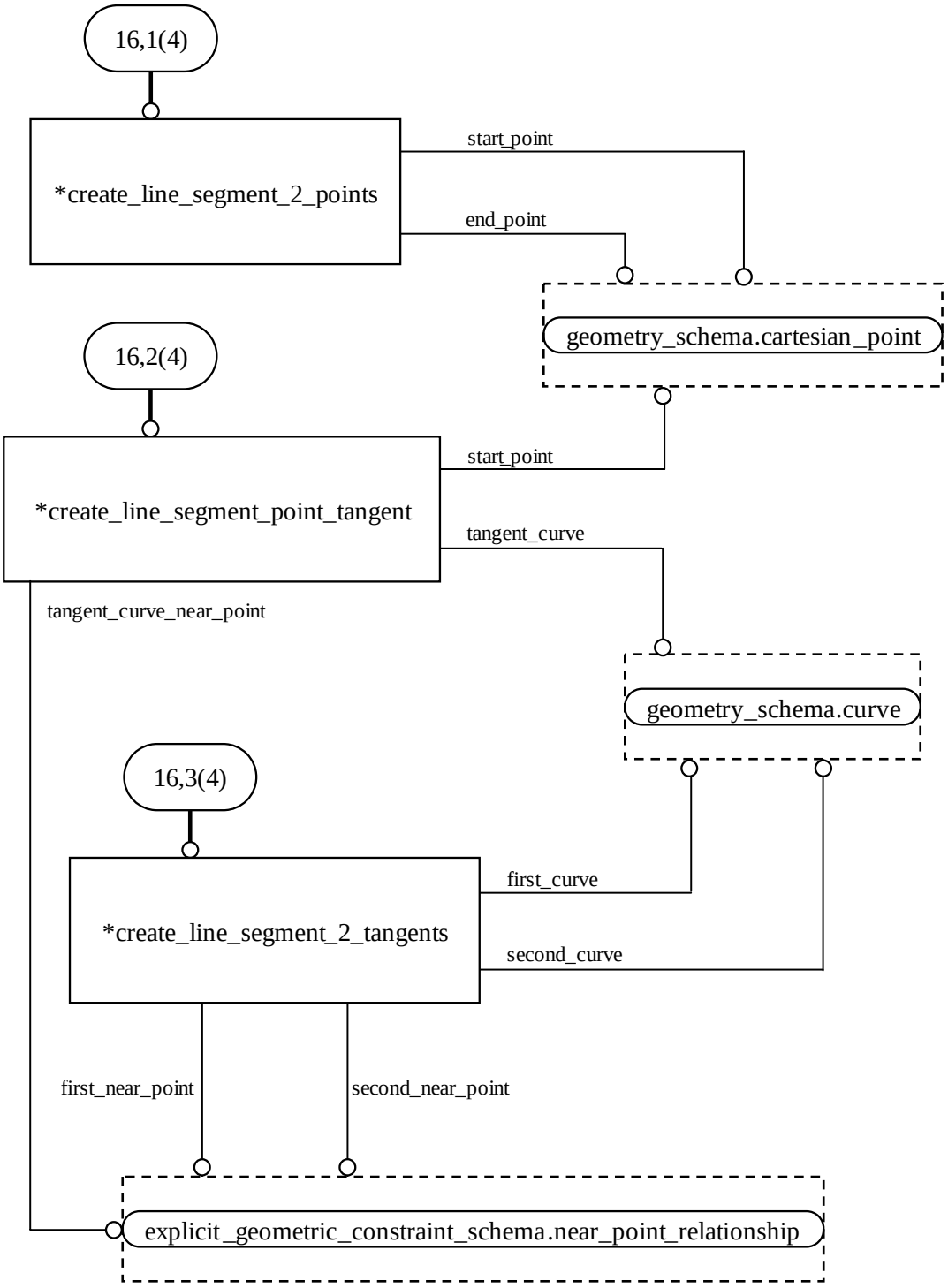


Figure D. 16 — procedural_sketch_schema EXPRESS-G diagram 16 of 16

Annex E
(informative)

Example of intended usage of this part of ISO 10303

This annex contains an example illustrating the intended use of the capabilities defined in this part of ISO 10303 for modelling the shape of Figure E.1. The harmonization with the related parts of ISO 10303, namely ISO 10303-42, ISO 10303-55, and ISO 10303-108, is also illustrated.

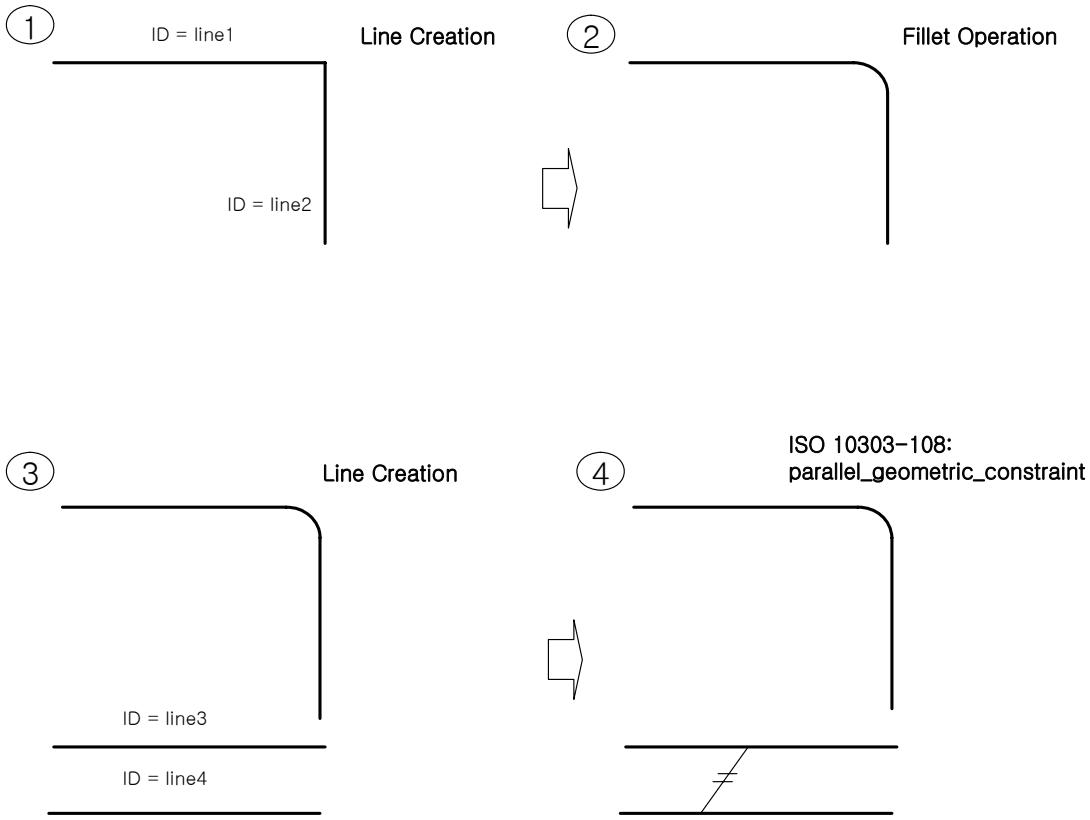


Figure E. 1 — Example

E.1 Example

```
ISO-10303-21;
HEADER;
/* Generated by software containing ST-Developer
 * from STEP Tools, Inc. (www.steptools.com)
 */

FILE_DESCRIPTION(
/* description */ (''),
/* implementation_level */ '2;1');

FILE_NAME(
/* name */ 'Example',
/* time_stamp */ '2004-02-27T16:32:19+09:00',
/* author */ ('Byungchul Kim'),
/* organization */ ('KAIST iCAD Lab.'),
/* preprocessor_version */ 'ST-DEVELOPER v8',
```

```
/* originating_system */ '',
/* authorisation */ '');

FILE_SCHEMA (('GEOMETRY_SCHEMA','REPRESENTATION_SCHEMA','PROCEDURAL_SKETCH',
'PROCEDURAL_SHAPE_MODEL_SCHEMA','EXPLICIT_GEOMETRIC_CONSTRAINT_SCHEMA'));
ENDSEC;

DATA;
#10=PROCEDURAL_SHAPE_REPRESENTATION_SEQUENCE('Example',(#33,#34,#13,#14,
#12,#35,#36,#15,#16,#11),$,'sketch1');
#11=PARALLEL_GEOMETRIC_CONSTRAINT('constraint1','',(#23),(#24));
#12=CREATE_FILLET('Fillet1',#17,#18,2.,.T.);
#13=USER_SELECTED_SHAPE_ELEMENTS('',(#17));
#14=USER_SELECTED_SHAPE_ELEMENTS('',(#18));
#15=USER_SELECTED_SHAPE_ELEMENTS('',(#19));
#16=USER_SELECTED_SHAPE_ELEMENTS('',(#20));
#17=TRIMMED_CURVE('',#21,(#37),(#38),.T.,.CARTESIAN.);
#18=TRIMMED_CURVE('',#22,(#39),(#40),.T.,.CARTESIAN.);
#19=TRIMMED_CURVE('',#23,(#41),(#42),.T.,.CARTESIAN.);
#20=TRIMMED_CURVE('',#24,(#43),(#44),.T.,.CARTESIAN.);
#21=LINE('',#37,#25);
#22=LINE('',#39,#26);
#23=LINE('',#41,#27);
#24=LINE('',#43,#28);
#25=VECTOR('',#29,10.);
#26=VECTOR('',#30,7.);
#27=VECTOR('',#31,10.);
#28=VECTOR('',#32,10.);
#29=DIRECTION('',(1.,0.,0.));
#30=DIRECTION('',(0.,-1.,0.));
#31=DIRECTION('',(1.,0.,0.));
#32=DIRECTION('',(1.,0.,0.));
#33=CREATE_LINE_SEGMENT_2_POINTS('line1',#37,#38);
#34=CREATE_LINE_SEGMENT_2_POINTS('line2',#39,#40);
#35=CREATE_LINE_SEGMENT_2_POINTS('line3',#41,#42);
#36=CREATE_LINE_SEGMENT_2_POINTS('line4',#43,#44);
#37=CARTESIAN_POINT('point1',(0.,10.,0.));
#38=CARTESIAN_POINT('point2',(10.,10.,0.));
#39=CARTESIAN_POINT('point3',(10.,10.,0.));
#40=CARTESIAN_POINT('point4',(10.,3.,0.));
#41=CARTESIAN_POINT('point5',(0.,1.,0.));
#42=CARTESIAN_POINT('point6',(10.,1.,0.));
#43=CARTESIAN_POINT('point7',(0.,0.,0.));
#44=CARTESIAN_POINT('point8',(10.,0.,0.));
ENDSEC;
END-ISO-10303-21;
```

In this example, #10 represents the sequence of Figure E.1 in terms of the **procedural_shape_representation_sequence** instance defined in ISO 10303-55. The line1 and the line2 in Figure E.1 are created by the instances #33 and #34. The instances #13 and #14 select two curves represented by #17 and #18, respectively. The instances #17 and #18 are the curves created as a result of the instances #33 and #34, respectively. The instance #12 creates a fillet curve between #17 and #18.

Similarly, the instances #35 and #36 creates two line instances. The instances #15 and #16 select the instances #19 and #20 which created as a result of #instances #35 and #36, respectively. The instance #11 makes the selected instances parallel.

Index

ICS 25.040.40

Price based on 71 pages