Reference number of working document: **ISO/TC 184/SC1/WG7 N 322**

Version description: **V5 - February 2002**

Date: 2002-02-06

Reference number of document: **ISO/FDIS 14649-10**

Committee identification: ISO/TC 184/SC 1/WG 7

Secretariat: DIN

Industrial automation systems and integration

Physical device control

ISO 14649 Data model for Computerized Numerical Controllers

Part 10: General Process Data

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: Draft International standard
Document subtype:
Document stage: (20) Preparation
Document language: E

Copyright notice

Information on technical or structural contents are available at the following addresses:

Convener:	Friedrich Glantschnig
	AMT Consulting
Address:	Höhenweg 33a
	CH-5417 Untersiggenthal
Telephone:	+41 (056) 288 2042
FAX;	+41 (056) 288 2043
E-mail:	fglantschnig@swissonline.ch

Owners of this part of documents,

	Yong Tak Hyun/Jochen Wolf	Dr. Chiaki Sakomoto
	WZL, RWTH-Aachen	Komatsu Engineering Co. LZD
Adress:	Werkzeugmaschinenlabor	3-20-1 Nakase
	RWTH	Kawasaki-Ward
	D-52074 Aachen	J-210-0818 Kawasaki City
Telephone:	+49 (0241) 8028226	+89 (44) 288 8782
FAX:	+49 (0241) 8022293	+89 (44) 288 8777
E-mail:	j.wolf@wzl.rwth-aachen.de	chiaki_sakamoto@KEG.komatsu.co.jp

Contents

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 Standard ISO 14649-10 was prepared by Technical Committee ISO/TC 184, *TC Industrial automation systems and integration*, Subcommittee SC 1, *SC Physical device control*.

ISO 14649 consists of the following parts, under the general title “Data model for Computerized Numerical Controllers”. A three phase release of the standard will take place, as described in section 7.1.2 of ISO 14649-1 .

Part 1:	Overview and fundamental principles,	published as DIS in Phase 1
Part 2:	Language Bindings, Fundamentals,	will be published in Phase 3
Part 3:	Language Binding in Java,	will be published in Phase 3
Part 9:	Glossary,	will be published in Phase 3
Part 10:	General Process Data,	published as DIS in Phase 1
Part 11	Process Data for Milling,	published as DIS in Phase 1
Part 12	Process Data for Turning,	will be published in Phase 3
Part 13	Process Data for EDM,	will be published in Phase 3
Part 50	AIM schema of General Process Data,	will be published in Phase 2
Part 51	AIM of Process Data for Milling,	will be published in Phase 2
Part 52	AIM of Process Data for Turning,	will be published in Phase 3
Part 53	AIM of Process Data for EDM,	will be published in Phase 3
Part 111	Tools for Milling,	published as DIS in Phase 1

Gaps in the numbering were left to allow further additions. Parts 2 and 3 are for language bindings according to ISO 10303 methods. Part 10 is the ISO 10303 Application Reference Model (ARM) for process-independent data. ISO 10303 ARMs for specific technologies are added after part 10. Part 50 is the ISO 10303 Application Interpreted Model (AIM) for process-independent data. ISO 10303 AIMs for specific technologies are added after Part 50.

ISO 14649 is harmonized with ISO 10303 in the common field of Product Data over the whole life cycle. Figure 1 of ISO 14649-1 shows the different fields of standardization between ISO 14649, ISO10303 and CNC manufacturers with respect to implementation and software development.

This part of ISO 14649, General process data, has several annexes:

Normative annexes:

Annex A:EXPRESS expanded listing

Annex B:Short names of entities

Annex C:Implementation method specific requirements

Informative annexes:

Annex D:EXPRESS-G figures

Introduction

Modern manufacturing enterprises are built from facilities spread around the globe, which contain equipment from hundreds of different manufacturers. Immense volumes of product information must be transferred between the various facilities and machines. Today's digital communications standards have solved the problem of reliably transferring information across global networks. For mechanical parts, the description of product data has been standardized by ISO 10303. This leads to the possibility of using standard data throughout the entire process chain in the manufacturing enterprise. Impediments to realizing this principle are the data formats used at the machine level. Most computer numerical control (CNC) machines are programmed in the ISO 6983 "G and M code" language. Programs are typically generated by computer-aided manufacturing (CAM) systems that use computer-aided design (CAD) information. However, ISO 6983 limits program portability for three reasons. First, the language focuses on programming the tool center path with respect to machine axes, rather than the machining process with respect to the part. Second, the standard defines the syntax of program statements, but in most cases leaves the semantics ambiguous. Third, vendors usually supplement the language with extensions that are not covered in the limited scope of ISO 6983.

ISO 14649 is a new model of data transfer between CAD/CAM systems and CNC machines, which replaces ISO 6983. It remedies the shortcomings of ISO 6983 by specifying machining processes rather than machine tool motion, using the object-oriented concept of Workingsteps. Workingsteps correspond to high-level machining features and associated process parameters. CNCs are responsible for translating Workingsteps to axis motion and tool operation. A major benefit of ISO 14649 is its use of existing data models from ISO 10303. As ISO 14649 provides a comprehensive model of the manufacturing process, it can also be used as the basis for a bi- and multi-directional data exchange between all other information technology systems.

ISO 14649 represents an object oriented, information and context preserving approach for NC-programming, that supersedes data reduction to simple switching instructions or linear and circular movements. As it is object- and feature oriented and describes the machining operations executed on the workpiece, and not machine dependent axis motions, it will be running on different machine tools or controllers. This compatibility will spare all data adaptations by postprocessors, if the new data model is correctly implemented on the NC-controllers. If old NC programs in ISO 6983 are to be used on such controllers, the corresponding interpreters shall be able to process the different NC program types in parallel.

ISO TC184/SC1/WG7 envisions a gradual evolution from ISO 6983 programming to portable feature-based programming. Early adopters of ISO 14649 will certainly support data input of legacy "G and M codes" manually or through programs, just as modern controllers support both command-line interfaces and graphical user interfaces. This will likely be made easier as open-architecture controllers become more prevalent. Therefore, ISO 14649 does not include legacy program statements, which would otherwise dilute the effectiveness of the standard.

Industrial automation systems and integration
Physical device control
Data model for computerized numerical controllers

Part 10: General process data

1 Scope

This part of ISO 14649 specifies the process data which is generally needed for NC-programming within all machining technologies. These data elements describe the interface between a computerised numerical controller and the programming system (i.e. CAM system or shopfloor programming system). On the programming system, the programme for the numerical controller is created. This programme includes geometric and technological information. It can be described using this part of ISO 14649 together with the technology-specific parts (ISO 14649-11, etc.). This part of ISO 14649 provides the control structures for the sequence of programme execution, mainly the sequence of working steps and associated machine functions.

The “machining_schema” defined in this part of ISO 14649 contains the definition of data types which are generally relevant for different technologies (e.g. milling, turning, grinding). The features for non-milling technologies like turning, EDM, etc. will be introduced when the technology specific parts like ISO 14649-12 for turning, ISO 14649-13 for EDM, and ISO 14649-14 for contour cutting of wood and glass are published. It includes the definition of the workpiece, a feature catalogue containing features which might be referenced by several technologies, the general executables and the basis for an operation definition. Not included in this schema are geometric items and representations, which are referenced from ISO 10303’s generic resources, and the technology-specific definitions, which are defined in separate parts of ISO 14649.

This part of ISO 14649 cannot stand alone. An implementation needs in addition at least one technology-specific part (e.g. ISO 14649-11 for milling, ISO 14649-12 for turning).

Additionally, the schema uses machining features similar to ISO 10303-224 and ISO 10303-214. The description of process data is done using the EXPRESS language as defined in ISO 10303-11. The encoding of the data is done using ISO 10303-21.

2 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this part of ISO 14649. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 14649 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document, released before this document, applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO 286-1:1988, ISO system of limits and fits - Part 1: Bases of tolerances, deviations, and fits.

ISO 286-2:1988, ISO system of limits and fits - Part 2: Tables of standard tolerance grades and limit deviations for holes and shafts.

ISO 841 Physical devices - Axis and motion nomenclature

ISO 2806 Physical devices - Vocabulary for NC-machines

ISO 10303 Part1 Overview and fundamental principles

ISO 10303 Part11 Description methods: The EXPRESS language reference manual

ISO 10303 Part21 Implementation methods: Clear text encoding of exchange structure

ISO 10303 Part41 Fundamentals of product description and support

ISO 10303 Part42 Geometric and topological representation

ISO 10303 Part43 Representation structures

ISO 10303 Part203 Application protocol: Configuration controlled designs

ISO 10303-214:2001, Industrial automation systems and integration - Product data representation and exchange - Part 214: Application protocol: automotive mechanical design process.

ISO 10303-224:2001, Industrial automation systems and integration - Product data representation and exchange - Part 224: Application protocol: Mechanical product definition for process planning using machining feature.

ISO 10303-514:1999, Industrial automation systems and integration - Product data representation and exchange - Part 224: Application interpreted construct: Advanced boundary representation.

3 Terms and definitions

For the purposes of this part of ISO 14649 the following terms definitions and abbreviations are used:

2D: Geometry in a xy-plane, where all the geometrie's points have only x and y coordinates.

2½D machining: Machining of a prismatic part. Typically, the workpiece is processed in several layers which are located perpendicular to the tool axis. In the EXPRESS listing of ISO14649, the term "two5D" is used for entity and attribute names.

3D: Geometry in three-dimensional space, where all points have x, y, and z coordinates.

freeform machining:Machining of freeform surfaces. For this kind of machining, the tool has to move in at least three axes simultaneously while processing the workpiece. Sometimes five-axes milling machines are used to reach an optimised angle between the tool and the workpiece surface.

CAM Computer Aided Manufacturing

CNC Computer Numerical Control

EDM Electrical Discharge Machining

EXPRESS: The language described in ISO10303-11.

EXPRESS-G: The graphic representation of the EXPRESS language as described in ISO10303-11.

Feature: A geometric entity of a workpiece which has semantic significance. In the context of ISO 14649, manufacturing features are used.

SI International system of units

4 General process data

4.1 Header and references

The following listing gives the header and the list of entities which are referenced within this schema.

```
SCHEMA machining_schema;

REFERENCE FROM approval_schema (* ISO 10303-41 *) (
  approval
);

REFERENCE FROM date_time_schema (* ISO 10303-41 *) (
  date_and_time
);

REFERENCE FROM measure_schema (* ISO 10303-41 *) (
  length_measure,
  parameter_value,
  plane_angle_measure,
  positive_length_measure,
  positive_ratio_measure,
  time_measure
);

REFERENCE FROM person_organization_schema (* ISO 10303-41 *) (
  person, address
);

REFERENCE FROM support_resource_schema (* ISO 10303-41 *) (
  identifier,
  label,
  text
);

REFERENCE  FROM geometry_schema (* ISO 10303-42 *) (
  axis2_placement_3d,
  b_spline_curve,
  bounded_curve,
  bounded_pcurve,
  bounded_surface,
  cartesian_point,
  direction,
  elementary_surface,
  plane
);

REFERENCE FROM topology_schema (* ISO 10303-42 *) (
  face,
  open_shell
);

USE FROM geometric_model_schema (* ISO 10303-42 *) (
  block,
  right_circular_cylinder
);

REFERENCE FROM aic_topologically_bounded_surface (* ISO 10303-511 *) (
  advanced_face
);
```

```
REFERENCE FROM aic_advanced_brep (* ISO 10303-514 *) (
  advanced_brep_shape_representation
);
```

4.2 General types and definitions

4.2.1 Measure units

The types of units supported by ISO 14649 are SI units as well as derived or conversion based units as defined in ISO 10303-41. If no units are given then, the following units are assumed ::

length_measure	millimetres [mm],
time_measure	seconds [s],
plane_angle_measure	degrees [°],
pressure_measure	Pascal [pa],
speed_measure	meters per second [m/sec],
rot_speed_measure	revolutions per second [1/sec].

4.2.1.1 Toleranced length measure

Length measure with tolerance.

```
ENTITY toleranced_length_measure;
  theoretical_size:      positive_length_measure;
  implicit_tolerance:    tolerance_select;
END_ENTITY;
```

theoretical_size: The theoretical length.

implicit_tolerance: The type of tolerance to apply to theoretical_size.

Note that all geometric properties of the workpiece are specified using toleranced_length_measure. If the NC controller has the ability to generate toolpaths or to make decisions about the tool used, it is the controller's responsibility to meet these tolerance requirements. On the other hand, data provided to the NC controller for explicit specification of movements will have no tolerances as the controller cannot do more than try to follow the given theoretical values to the best of its abilities. The same is true for offsets and other data referring to already toleranced dimensions.

4.2.1.1.1 Tolerance select

Select type offering different etities to describe tolerances for scalar values.

```
TYPE tolerance_select = SELECT(plus_minus_value, limits_and_fits);
END_TYPE;
```

4.2.1.1.2 Plus minus value

The plus_minuns_value describes the upper and lower limits valid for a scalar dimension referencing this entity.

```
ENTITY plus_minus_value;
```

```
    upper_limit:           positive_length_measure;
    lower_limit:           positive_length_measure;
    significant_digits:    INTEGER;
  END_ENTITY;
```

upper_limit: The value of the tolerance that shall be added to the exact value to establish the maximum allowed value.

lower_limit: the value of the tolerance that shall be subtracted from the exact value to establish the minimum allowed value.

significant_digits: The number of decimal digits indicating the accuracy of the lower_bound and upper_bound values.

4.2.1.1.3 Limits and fits

A Limits_and_fits contains the necessary information to express a tolerance of the limits-and-fits system standardized by ISO 286.

```
  ENTITY limits_and_fits;
    deviation:             length_measure;
    grade:;                length_measure
    its_fitting_type:      OPTIONAL fitting_type;
  END_ENTITY;
```

deviation: The difference between a measured actual size and the corresponding basic size as defined in ISO 286-1.grade:The grade specifies the quality or the accuracy grade of a toleranc

fitting_type: Specification whether the tolerance declaration applies to a shaft or to a hole.

4.2.1.1.4 Fitting type

The enumeration used to specify the type of fitting.

```
  TYPE fitting_type = ENUMERATION OF (shaft,hole);
  END_TYPE;
```

4.2.1.2 Speed measure

A measure for a linear speed used for cutting speeds and feed rates.

```
  TYPE speed_measure = REAL;
  END_TYPE;
```

4.2.1.3 Rotational speed measure

A measure for a rotational speed. Positive values indicate rotation in the mathematical positive sense, i. e. counter-clockwise motion.

```
  TYPE rot_speed_measure = REAL;
  END_TYPE;
```

4.2.1.4 Pressure measure

A measure for pressure.

```
  TYPE pressure_measure = REAL;
  END_TYPE;
```

4.2.2 Other general types

4.2.2.1 Rotational direction

Enumeration used to identify the direction (sense) of the tool's rotation or of the direction of a circular movement. It should not be used in conjunction with rot_speed_measure which carries its own sense of rotation.

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

Note: cw means clockwise, ccw means counter-clockwise.

4.2.2.2 Shape tolerance

Type for definition of shape tolerance.

```
TYPE shape_tolerance = length_measure;
END_TYPE;
```

4.3 Where to start: Project

Each part programme, i.e. data model, based on ISO14649 must include exactly one top-level entity, called project. The project indicates the workplan to be executed upon interpretation of this model (as several workplans might be included), and it may also provide the workpiece(s) upon which actions are to be performed.

```
ENTITY project;
  its_id:          identifier;
  main_workplan:   workplan;
  its_workpieces:  SET [0:?] OF workpiece;
  its_owner:       OPTIONAL person_and_address;
  its_release:     OPTIONAL date_and_time;
  its_status:      OPTIONAL approval;
  (*
  Informal proposition:
  its_id shall be unique within the part programme.
  *)
END_ENTITY;
```

its_id: The project's identifier. It shall be unique within the part programme.

main_workplan: The top-level workplan in this model.

its_workpieces: The workpieces upon which actions are to be performed.

its_owner: Optional information on the owner of the project.

its_release: Optional date and time reference of the project.

its_status: Optional attribute to indicate the current status of the project.

4.3.1 Person and address

Entity includes data to name and reference a person, who for instance is responsable for creating a project.

```
ENTITY person_and_address;
  its_person:            person;
  its_address:           OPTIONAL address;
END_ENTITY;
```

4.4 What to machine: Workpiece and manufacturing feature

4.4.1 Workpiece

The workpiece entity contains the entire description of the workpiece, if available. This includes material, surface condition and geometric data. Each workpiece has only one surface condition and one material. Dependent on the conformance class the workpiece entity includes the raw part dimension only as an including box or cylinder, or, in a higher class, as a representation which can be a complete geometric model, e.g. the state after previous manufacturing operations.

```
ENTITY workpiece;
  its_id:                identifier;
  its_material:          OPTIONAL material;
  global_tolerance:      OPTIONAL shape_tolerance;
  its_rawpiece:          OPTIONAL workpiece;
  its_geometry:          OPTIONAL advanced_brep_shape_representation;
  its_bounding_geometry: OPTIONAL bounding_geometry_select;
  clamping_positions:    SET [0:?] OF cartesian_point;
END_ENTITY;
```

its_id: The unique identification of a workpiece.

its_material: The material attribute identifies the workpiece material. This data shall be used for determining the technological process parameters for the manufacturing process. It can be done by the machine operator or by an automatic feed rate/cutting condition selection (from a table or a data base on the CNC).

global_tolerance: Tolerance for the workpiece, valid where no other tolerances are specified.

its_rawpiece: The rawpiece geometry of the workpiece may be described here. A recursive description is used, i.e. the rawpiece is of type workpiece itself.

its_geometry: An exact description of the final workpiece geometry according to ISO 10303-514.

its_bounding_geometry: By this attribute the workpiece´s bounding geometry might be defined as a box, a cylinder or a geometry according to the definition of the entity advanced_brep_shape_representation (ISO 10303--514).

clamping_positions: Positions of the clamping device on the workpiece’s surface.

All attribute’s locations, directions and geometrical information are defined relatively to the workpiece’s coordinate system.

4.4.1.1 Material

This entity is for identifying the workpiece material.

```
ENTITY material;
  standard_identifier:   label;
  material_identifier:   label;
  material_property:     SET [0:?] OF property_parameter;
END_ENTITY;
```

standard_identifier: The standard used for identifying the material. This can be a national standard or one used internally in the company.

material_identifier: The name which identifies the material.

material_property: The parameter which describes the properties of material. Since the demand for material properties varies, a generic type „property parameter” is used.

4.4.1.2 Property parameter

Generic property parameter which may be used to characterise any kind of property any kind of parameter might have. Subtypes are descriptive parameters (strings) and numeric parameters.

```
ENTITY property_parameter
  SUPERTYPE OF (ONEOF (descriptive_parameter,numeric_parameter));
  parameter_name:        label;
END_ENTITY;
```

parameter_name: The name of the parameter to be described.

4.4.1.2.1 Descriptive parameter

A parameter description using a string to characterise the parameter.

```
ENTITY descriptive_parameter
  SUBTYPE OF (property_parameter);
  descriptive_string:    text;
END_ENTITY;
```

descriptive_string: String value which describes the parameter.

4.4.1.2.2 Numeric parameter

A parameter description using a numeric value. Both the number and the unit shall be given.

```
ENTITY numeric_parameter
  SUBTYPE OF (Property_parameter);
  its_parameter_value:   parameter_value;
  its_parameter_unit:    label;
END_ENTITY;
```

its_parameter_value: The value which describes the parameter.

its_parameter_unit: The units associated with the value.

4.4.1.3 Bounding geometry select

This type offers three different entities to describe the bounding geometry of a workpiece. Depending on the conformance class the bounding geometry is described as a block or a cylinder using the geometric items of ISO 10303-42 or as a complex shape conformable to ISO 10303-514. All coordinates and directions given in bounding_geometry are defined in the given workpiece's coordinate system.

```
TYPE bounding_geometry_select = SELECT (
                    block, right_circular_cylinder,          (* ISO 10303-42 *)
                    advanced_brep_shape_representation       (* ISO 10303-514*)
                      );
END_TYPE;
```

4.4.2 Manufacturing feature

This entity is the supertype of all manufacturing features. When considering 2½D-manufacturing, features are of type two5D_manufacturing_feature and may be holes, pockets, etc. When considering freeform manufacturing, regions are used in the same sense.

The manufacturing feature describes the feature as such, e.g. by its geometric properties. It does not give any instruction on how the workpiece is manufactured. Such process related information is contained only in the operations and depends on the manufacturing method. The methods are described in the technology specific parts of ISO 14649. For example, plunging into a pocket, roughing the pocket and finishing the pocket bottom may all be individual operations, as well as the finishing of a portion of a freeform surface. For the concept of operations, please refer to section 4.7.1.

```
ENTITY manufacturing_feature
  ABSTRACT SUPERTYPE OF (ONEOF(region, two5D_manufacturing_feature,
    transition_feature));
  its_id:              identifier;
  its_workpiece:       workpiece;
  its_operations:      SET [0:?] OF machining_operation;
END_ENTITY;
```

its_id: Each feature has an unique identifier.

its_workpiece: The workpiece which the feature is part of.

its_operations: A set of all operations associated with the feature required for the manufacturing of the feature. Note that the operations are not necessarily executed immediately after each other. Only the workplan determines the final order of operations of all operations included in a given programme, and manufacturing will typically not occur feature by feature but rather according to technological criteria like minimised tool change. However, it should be guaranteed by the CAM system (or by the controller, if the order of execution is determined by an intelligent CNC) that the order given in this attribute is never violated.

4.5 Catalogue of manufacturing features

4.5.1 Region

The region is the equivalent of a feature in freeform machining. It describes a bounded area of the final workpiece surface to which the associated operations will be applied. Note that for freeform surfaces the regions for different operations (e.g. roughing, finishing) may have different shapes based upon technological decisions.

```
ENTITY region
ABSTRACT SUPERTYPE OF (ONEOF (region_surface_list, region_projection,
                             topological_region))
SUBTYPE OF (manufacturing_feature);
feature_placement:     OPTIONAL axis2_placement_3d;
END_ENTITY;
```

feature_placement: The placement of the feature relative to the workpiece co-ordinate system. The placement is a translation and/or a rotation which transforms the origin of the workpiece co-ordinate system origin into the origin of the feature’s local co-ordinate system. If no feature_placement is given, the region will use workpiece co-ordinates. Regarding coordinate systems see also Section 4.6.4.1.2.

4.5.1.1 Region projection

A type of bounded region generated by projecting a closed curve on a surface. Thus, a region can easily be defined on an existing surface description. The region projection applies to any workpiece surface it hits when moving in space along proj_dir. This is a very simplistic region definition for situations where the use of topology is not desired.

```
ENTITY region_projection
  SUBTYPE OF (region);
  proj_curve :         bounded_curve;
  proj_dir   :         direction;
```

```
    depth:                  toleranced_length_measure;
  END_ENTITY;
```

proj_curve: A curve in space, used for specifying the boundary for a surface through its projection on the surface. It must be a closed curve and will usually be in one plane.

proj_dir: Direction used for projecting the above curve on the surface.

depth: The depth is a positive scalar value. It describes the distance of material removal into negative z-direction of the region's local coordinate system. The depth is measured from the raw piece's surface, onto which the proj_curve has been projected, to the bottom to be machined. The depth is equal for the whole region. In case the bottom's shape shall not be equal to the raw piece's surface, the entity pocket and its subtypes have to be used.

4.5.1.2 Region surface list

A type of region specified by a list of surfaces. This allows for the most general description of regions. The region is bounded by the borders of the surfaces it encloses.

```
  ENTITY region_surface_list
    SUBTYPE OF (region);
    surface_list :          LIST [1:?] OF bounded_surface;
  END_ENTITY;
```

surface_list: List of general surfaces, allowing for various descriptions.

4.5.1.3 Topological region

A type of region specified by a topological representation. Only faces of type advanced_face are allowed as they are certain to carry their own geometric representation.

```
  ENTITY topological_region
    SUBTYPE OF (region, open_shell);
  WHERE
    WR1: SIZEOF(QUERY(it <* SELF.cfs_faces | NOT('MACHINING_SCHEMA.ADVANCED_FACE' IN
      TYPEOF(it)))) = 0;
  END_ENTITY;
```

4.5.2 Two5D manufacturing feature

The entity two5D_manufacturing_feature is the abstract supertype of all 2½D features. This structure is defined in close resemblance to ISO 10303-224.

```
  ENTITY two5D_manufacturing_feature
    ABSTRACT SUPERTYPE OF (ONEOF(machining_feature, replicate_feature,
      compound_feature))
    SUBTYPE OF (manufacturing_feature);
    feature_placement:      axis2_placement_3d;
  END_ENTITY;
```

feature_placement: The placement of the feature relative to the workpiece co-ordinate system. The placement is a translation and/or a rotation which transforms the origin of the workpiece co-ordinate system origin into the origin of the feature's local co-ordinate system. For information on coordinate systems refer to Section 4.6.4.1.2. If the manufacturing feature is part of a compound_feature, then its placement is defined relative to the compound_feature´s origin.

4.5.3 Machining feature

The entity machining_feature is the abstract supertype of all features used for feature based 2½D machining. In this part the following features are foreseen as they are used in different technology specific parts of ISO 14649: planar face, pocket, slot, step, hole, generic feature, and compound feature. The features are defined in close resemblance to ISO 10303-224.

2½D machining is characterised by the fact that most tool movements occur only in the xy plane while the z axis is preset to a certain depth in order to remove a layer of material. For this reason, all machining features have a depth.

All feature geometry, e.g. the contour describing the outline of a pocket, is described in a local xyz co-ordinate system. The definition of the coordinate system's orientation is given in the machining method specific parts. The surrounding surface of the workpiece and the definition of the planar contour of the feature are assumed to lie in the xy plane (z = 0). The material is assumed to be in negative z direction of the xy-plane. In other words, a positive depth within a machining operation requires the tool to advance in negative z direction into the material's direction.

As ISO 14649 allows for a 3D description of the overall workpiece geometry, the inherited attribute feature_placement specifies the actual location of the feature within the workpiece co-ordinate system.

```
ENTITY machining_feature
ABSTRACT SUPERTYPE OF (ONEOF(planar_face, pocket, slot, step, round_hole,
  toolpath_feature, profile_feature, boss, spherical_cap, rounded_end, thread))
  SUBTYPE OF (two5D_manufacturing_feature);
  depth:                    elementary_surface;
END_ENTITY;
```

depth: The depth of the feature is described by a plane which includes the lowest points of the feature (z<0) measured in the feautre's local cartesian-cooridnate system. If the depth is not an orthogonal plane to the z-axis this implies, that the bottom of the feature is inclined. Depending on the explicit features there can be described more complex bottom shapes.

4.5.4 Planar face

The planar face is used to describe machining of the outer face of a workpiece. The geometry of the planar face is given through the boundary and the depth (defined by the supertype). The depth denotes the the bottom of the material that needs to be taken away from the workpiece to reach the final shape of the feature. The elementary surface describing the depth can be inclined or orthogonal to the feature's local z-axis. The planar face may also have one or more bosses where no cutting will be done.

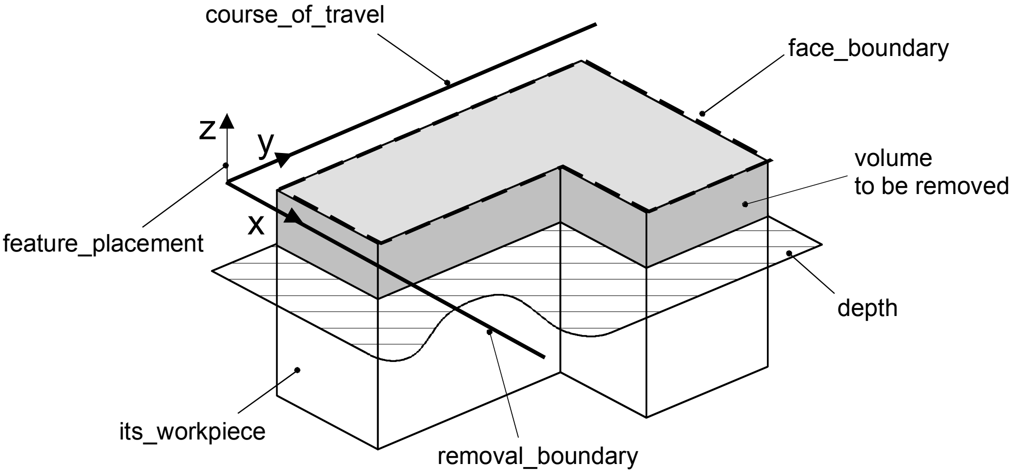


Figure 1.Planar face

```
ENTITY planar_face
  SUBTYPE OF (machining_feature);
  course_of_travel:      linear_path;
  removal_boundary:      linear_profile;
  face_boundary:         OPTIONAL closed_profile;
  its_boss:              SET [0:?] OF boss;
(*
Informal propositions:
- The entire profile lies in the local xy plane.
- The profile is not self-intersecting.
*)
END_ENTITY;
```

removal_direction: The direction of material removal.

course_of_travel: A straight line with magnitude and direction.

removal_boundary: A line with direction and magnitude that when swept along a path defines the area on a workpiece for volume removal.

face_boundary: The complete or partial outside final shape of the workpiece after the planar cut has been applied.

its_boss: An optional list of entities which describe one or more bosses. A boss of a planar_face defines a part of the planar_face which is not cut during manufacturing of the planar_face.

4.5.5 Pocket

This is the abstract supertype for different implicit and explicit pockets. Open and closed pockets are derived from this supertype. The geometry of the pocket is defined by its contour on the outer face of the workpiece and its depth. A pocket may possess one or more bosses.

```
ENTITY pocket
  ABSTRACT SUPERTYPE OF (ONEOF(closed_pocket, open_pocket))
  SUBTYPE OF (machining_feature);
  its_boss:              SET [0:?] OF boss;
  slope:                 OPTIONAL plane_angle_measure;
  bottom_condition:      pocket_bottom_condition;
  planar_radius:          OPTIONAL toleranced_length_measure;
  orthogonal_radius:      OPTIONAL toleranced_length_measure;
END_ENTITY;
```

its_boss: Optional list of bosses. This defines one or more parts of the pocket which are not cut during manufacturing of the pocket. With the cutting of the pocket, the boss is cut simultaneously.

slope: Optional angle of the border of the pocket measured against the local z axis. Default is 0 degrees.

bottom_condition: Possible bottom conditions of the pocket.

planar_radius: The planar radius of a fillet, if existing within a pocket. This may be used to check against the radius of the tool tip.

orthogonal_radius: The orthogonal radius of a fillet, if existing within a pocket. This may be used to check against the radius of the tool.

4.5.5.1 Closed pocket

Derived from the class pocket. A closed pocket is a pocket which is surrounded by material everywhere along its circumference. Its feature boundary is given by a closed profile.

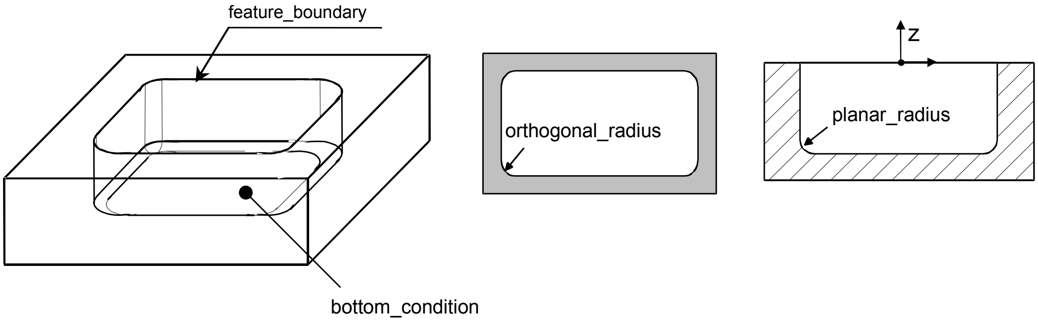


Figure 2.Closed pocket

```
ENTITY closed_pocket
  SUBTYPE OF (pocket);
  feature_boundary:      closed_profile;
(*
Informal propositions:
- The entire profile lies in the feature’s local xy plane.
- The profile is closed and not self-intersecting.
*)
END_ENTITY;
```

feature_boundary: The shape or outline that describes the upper edge of the pocket. It is an enclosed area that has a completely closed profile. The profile specifies the area required by a closed pocket.

4.5.5.2 Open pocket

Derived from the class pocket. An open pocket is a pocket which is not a closed pocket. Its feature boundary is given by a wall contour.

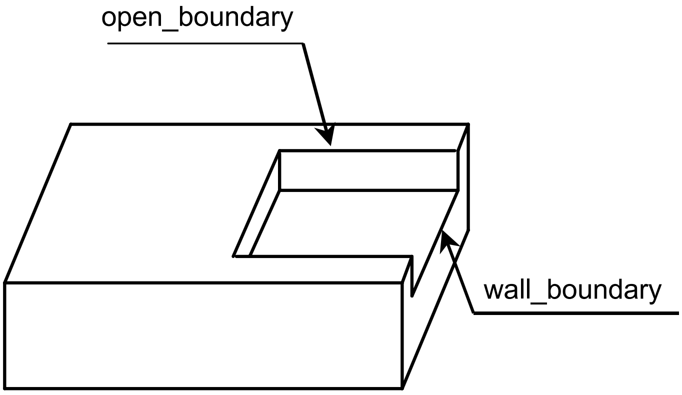


Figure 3.Open pocket

```
ENTITY open_pocket
  SUBTYPE OF (pocket);
  open_boundary:          open_profile;
  wall_boundary:          OPTIONAL open_profile;
(*
Informal propositions:
- The entire open_boundary profile lies in the local xy plane.
- The open_profiles are not self-intersecting.
- Together the two open_profiles form a closed profile.
- wall_boundary is for information only.
*)
END_ENTITY;
```

open_boundary: The outline or shape that forms the upper edge of the open_pocket. The profile specifies the area required by open pocket. When travelling along the profile based as defined by its sense, the material lies on the left side of the profile. The open_boundary lies within the features xy-plane (z=0).

wall_boundary: The outline or shape that forms the side edge of the open_pocket. It forms a closed profile together with open_boundary. Note that this contour will be most likely defined implicitly by the selected tool and the fillet options inherited from machining_feature. If given, it will be informative only.

4.5.5.3 Pocket bottom condition

This entity describes the shape of the bottom of a pocket.

```
ENTITY pocket_bottom_condition
  ABSTRACT SUPERTYPE OF
  (ONEOF (through_pocket_bottom_condition, planar_pocket_bottom_condition,
    radiused_pocket_bottom_condition, general_pocket_bottom_condition));
END_ENTITY;
```

4.5.5.3.1 Through pocket bottom condition

The pocket extends fully through the workpiece.

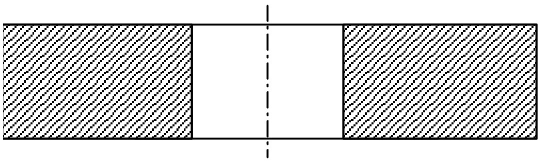


Figure 4.Through pocket bottom condition

```
ENTITY through_pocket_bottom_condition
  SUBTYPE OF (pocket_bottom_condition);
END_ENTITY;
```

4.5.5.3.2 Planar pocket bottom condition

The pocket has a plane bottom.

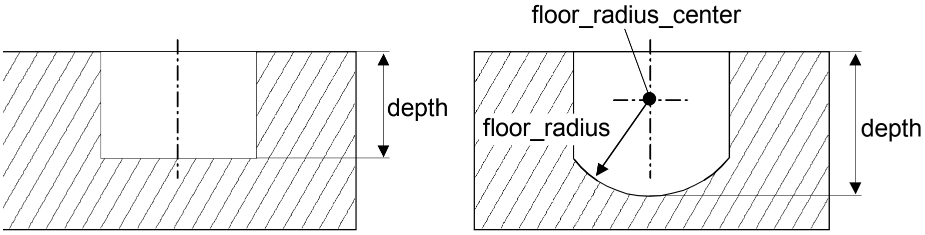


Figure 5.Planar (left) and radiused (right) pocket bottom condition

```
ENTITY planar_pocket_bottom_condition
  SUBTYPE OF (pocket_bottom_condition);
END_ENTITY;
```

4.5.5.3.3 Radiused pocket bottom condition

The pocket has a radiused bottom. As opposed to the use of the attribute planar_radius of the machining feature, which may be used to specify a fillet, the entire bottom surface will be continuously curved.

```
ENTITY radiused_pocket_bottom_condition
  SUBTYPE OF (pocket_bottom_condition);
  floor_radius_center:   cartesian_point;
  floor_radius:          toleranced_length_measure;
END_ENTITY;
```

floor_radius_center: Center of the radius of the pocket floor. It is defined within the features coordinate system.

floor_radius: Radius of the pocket floor. The radius creates a portion of a sphere on the pocket floor. Concave or convex behaviour depends upon the position of floor_radius_center. Note that the diameter of the sphere must span at least the perimeter of the pocket floor, or the result will be undefined.

Note that the depth of the pocket is determined by the combination of floor_radius_center and floor_radius

4.5.5.3.4 General pocket bottom condition

Any bottom condition not covered by planar_bottom_condition or through_bottom_condition. General pocket bottom condition defines a free-form surface at the bottom of a pocket.

Similar to the compound feature, the region referenced by the attribute shape is not allowed to be associated with machining operations of its own. All machining operations to manufacture the pocket must be defined by the attribute its_operations of the pocket itself.

```
ENTITY general_pocket_bottom_condition
  SUBTYPE OF (pocket_bottom_condition);
  shape:                 region;
WHERE
  WR1: SIZEOF(shape\manufacturing_feature.its_operations) = 0;
END_ENTITY;
```

shape: Description of the pocket floor as 3D surface in local xyz co-ordinates. Note that the feature_placement attribute of shape will specify its position relative to the local co-ordinate system of the pocket, not relative to the workpiece co-ordinates.

The pocket's attribute depth should specify the maximum depth of the pocket. Note that the actual depth of the pocket will be determined by the shape of the surface which is defined in 3D space. It will override any conflicting data given in the attribute depth.

4.5.6 Slot

The entity slot is closely related to the entity pocket. Generally speaking, a slot is a special kind of pocket. ISO 10303-224 lists slot as a separate entity. To be compatible, the entity slot is also included in this schema. The attribute course_of_travel describes the location and extension of the slot. Typically, the slot will be manufactured by a single sweep of a tool along the course of travel. In this case, the width of the slot equals the shape of the tool. In case of a milling operation the shape is given by the tool's diameter. If a workingstep which manufactures this slot calls for a smaller tool, more than one cut will have to be made.

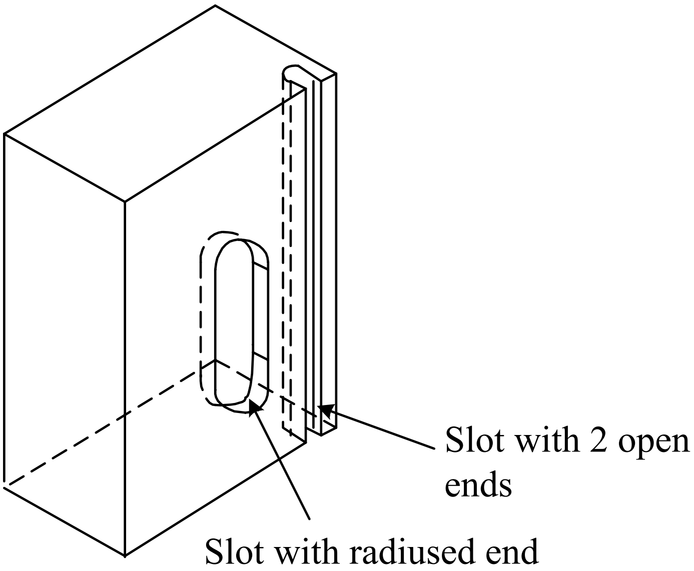


Figure 6.Different examples of slots

```
ENTITY slot
  SUBTYPE OF (machining_feature);
  course_of_travel:      travel_path;
  swept_shape:           open_profile;
  end_conditions:        LIST[0:2] OF slot_end_type;
WHERE
  WR1: (
        ( SIZEOF(QUERY (it <* SELF.end_conditions |
          ('MACHINING_SCHEMA.LOOP_SLOT_END_TYPE' IN TYPEOF(it)) )) = 1)
        AND
        (SIZEOF(end_conditions) = 1)
       )
       OR
       (SIZEOF(end_conditions) <> 1);

(*
Informal propositions:
- The entire travel_path lies in the local xy plane.
- The travel_path is not self-intersecting.
*)
END_ENTITY;
```

course_of_travel: Center line of the slot. The tool is moved along this path to achieve the manufacturing of the slot. As with the entity pocket, the upper edge is given.

swept_shape: The (contoured) cross-section generated by a the tool, required for the selection of the proper tool For simple rectangular slot profiles, a square_u_profile should be specified giving only the width of the slot.

end_conditions: For a slot closed on one side, end conditions may be given here. The sequence is oriented conformable to the direction of the course_of_travel.

4.5.6.1 Slot end type

Supertype of slot end types.

```
ENTITY slot_end_type
ABSTRACT SUPERTYPE OF (ONEOF (woodruff_slot_end_type, radiused_slot_end_type,
  flat_slot_end_type, loop_slot_end_type, open_slot_end_type));
END_ENTITY;
```

4.5.6.1.1 Woodruff slot end type

The end of slot shall be a radius tangent to the slot bottom and curved upward about an axis.

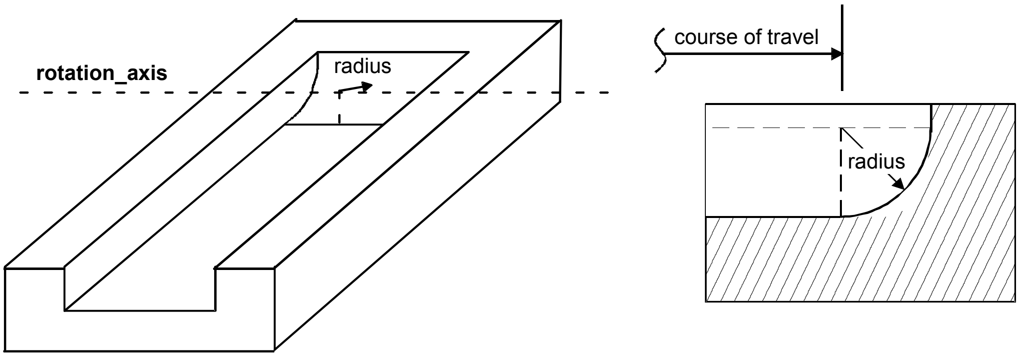


Figure 7.Woodruff type slot end type

```
ENTITY woodruff_slot_end_type
  SUBTYPE OF (slot_end_type);
  radius:              toleranced_length_measure;
END_ENTITY;
```

radius: Radius of the slot end type. The radius continues tangential from the bottom of the slot. It starts at the bottom of the slot where the course_of_travel ends. The radius does not necessarily equal to the depth of the slot.

4.5.6.1.2 Radiused slot end type

The end of the slot consists of an arc. The diameter equals the width of the slot. The center of the arc is identical with the end point of course_of_travel.

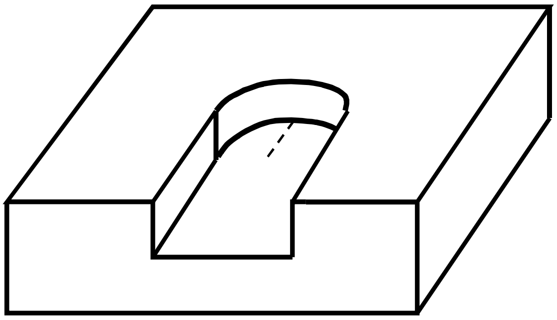


Figure 8.Radiused slot end type

```
ENTITY radiused_slot_end_type
   SUBTYPE OF (slot_end_type);
END_ENTITY;
```

4.5.6.1.3 Flat slot end type

The end of the slot consists of a flat line with two arcs connecting the end to the sides of the slot. The radii of the two arcs are given. When traversing the slot along its center line from start to finish, the radius on the left side of the center line is corner_radius1, the radius on the right side of the center line is corner_radius2. The end point of course_of_travel is in the flat end of the slot.

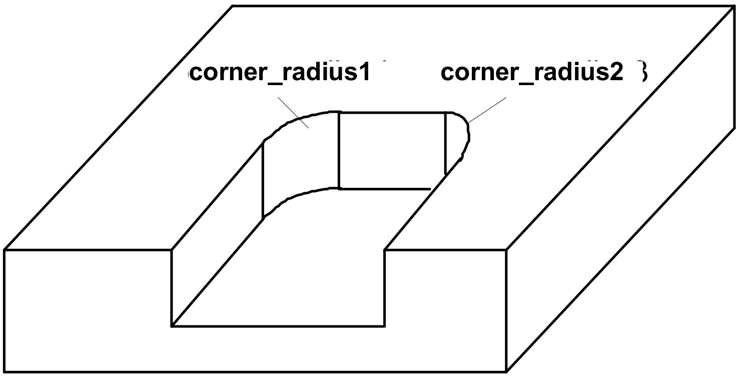


Figure 9.Flat slot end type

```
ENTITY flat_slot_end_type
   SUBTYPE OF (slot_end_type);
   corner_radius1 :       toleranced_length_measure;
   corner_radius2 :       toleranced_length_measure;
END_ENTITY;
```

corner_radius1: Radius of the first arc.

corner_radius2: Radius of the second arc.

4.5.6.1.4 Loop slot end type

The slot forms a loop. The attribute depth is used to determinde the scalar profoundness of the slot. The profoundness is equal to the smallest distance between the course_of_travel and the planar face described by the feature's attribute "depth".

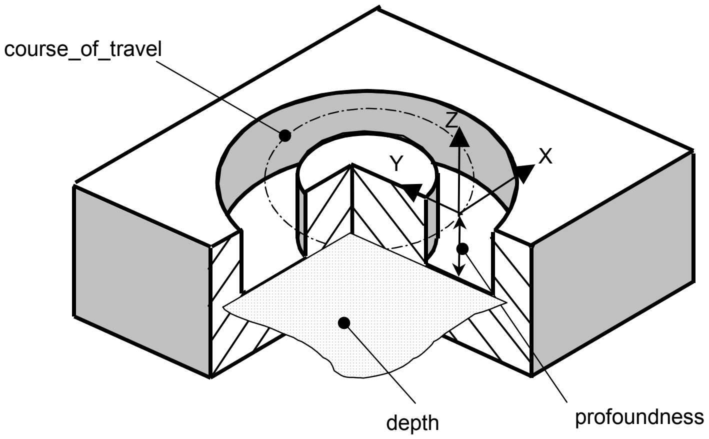


Figure 10.Loop slot end type

```
ENTITY loop_slot_end_type
  SUBTYPE OF (slot_end_type);
END_ENTITY;
```

4.5.6.1.5 Open slot end type

The end of the slot is open.

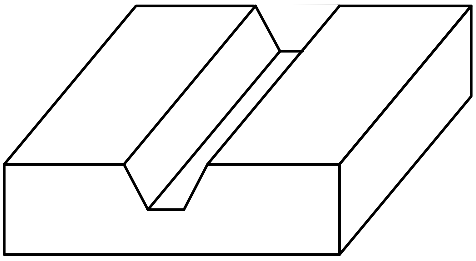


Figure 11.Open slot end condition

```
ENTITY open_slot_end_type
  SUBTYPE OF (slot_end_type);
END_ENTITY;
```

4.5.7 Step

A step (or shoulder) is a volume of material removed from the top and the sides of the workpiece. Like an open pocket, its contour is open to its sides. The part of the V-profile describing the step's bottom lies in the elementary surface defined by the feature's attribute depth. The step may have one or more bosses as given by the optional attribute.

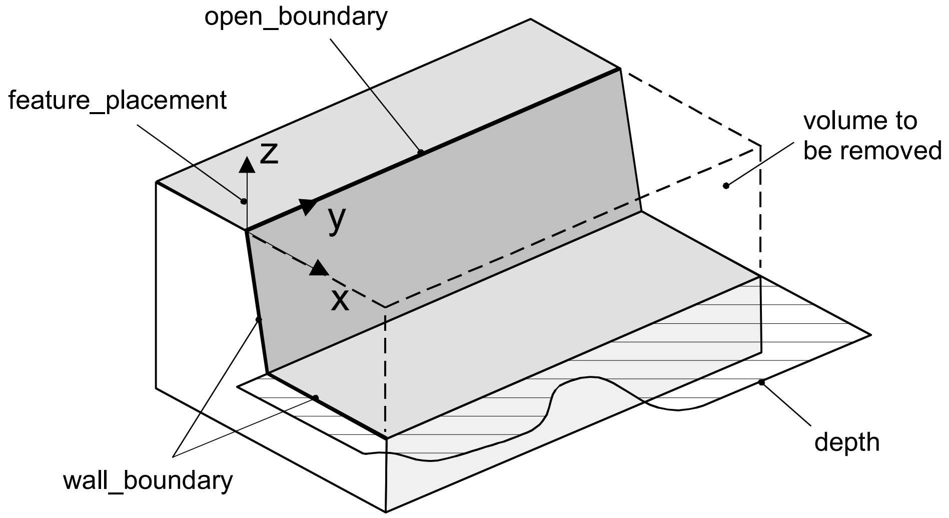


Figure 12.Step

```
ENTITY step
   SUBTYPE OF (machining_feature);
   open_boundary:         linear_path;
   wall_boundary:         OPTIONAL vee_profile;
   its_boss:              SET[0:?] OF boss;
  (*
Informal propositions:
- The entire linear_path lies in the same plane.
*)
END_ENTITY;
```

open_boundary: The outline or shape that forms the upper edge of the step. When travelling along the curve based as defined by its sense, the material is to the left of the curve.

wall_boundary: The outline or shape that forms the side edge of the step, i. e. the “profile” of the cut.

its_boss: A step may have one or more bosses..

4.5.8 Profile feature

A profile feature is a volume of material removed from the boundary shape of a workpiece. This is an abstract supertype of general outside profile and shape profile.

```
ENTITY profile_feature
ABSTRACT SUPERTYPE OF(ONEOF(general_outside_profile, shape_profile))
   SUBTYPE OF (machining_feature);
      profile_swept_shape : linear_path;
END_ENTITY;
```

profile_swept_shape: An 2D line, when combined with a profile, which creates the shape of the profile feature. The placement of the linear path shall be the same as the profile feature. The direction of linear shall be with the Z-axis toward the direction of travel of the profile boundary.

4.5.8.1 General outside profile

A general outside profile is the removal volume of arbitrary shape from the outside shape of a workpiece. It may remove material from the entire outside shape or some portion of the shape. The contour of the shape is given by the attribute feature_boundary.

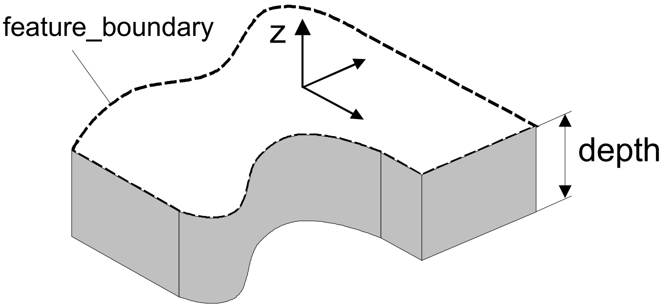


Figure 13.General outside profile

```
ENTITY general_outside_profile
  SUBTYPE OF (profile_feature);
  feature_boundary:      profile;
(*
Informal propositions:
- The entire profile lies in the local xy plane.
- The profile is not self-intersecting.
*)
END_ENTITY;
```

feature_boundary: The contour of the profile to be followed by the tool. Note that this attribute describes the profile itself, not the tool path. When travelling along the profile based as defined by its sense, the material lies to the left of the profile.

4.5.8.2 Shape profile

A shape profile is the removal volume of shaped profile from the boundary shape of a workpiece. The bottom of the boundary shape is limited by a floor condition. This is an abstract supertype of general shape profile, partial circular shape profile, circular closed shape profile, rectangular open shape profile, and rectangular closed shape profile.

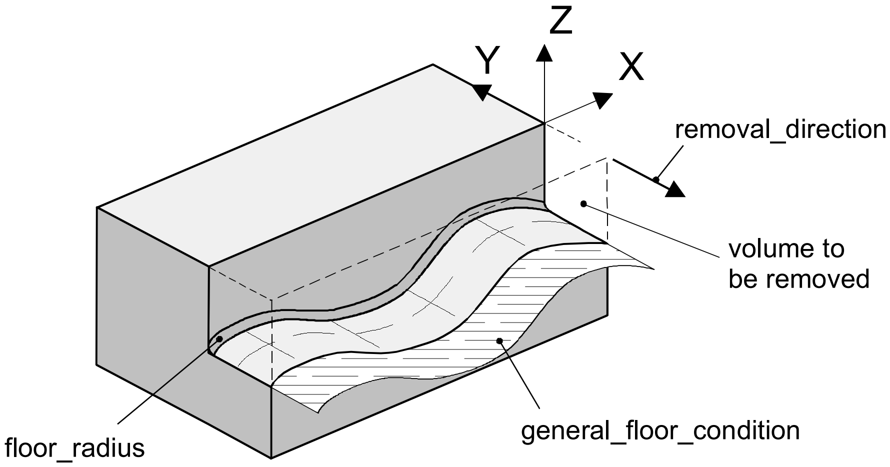


Figure 14.Shape Profile

```
ENTITY shape_profile
  ABSTRACT SUPERTYPE
    OF(ONEOF(general_shape_profile,partial_circular_shape_profile,
    circular_closed_shape_profile,rectangular_open_shape_profile,
    rectangular_closed_shape_profile))
  SUBTYPE OF (profile_feature);
     floor_condition:          profile_select;
     removal_direction:        direction;
END_ENTITY;
```

floor_condition: Specification of the shape of the bottom.

removal_direction: A vector that points in the general direction away from the material.

4.5.8.2.1 Profile select

The floor condition of shape profile is either a through profile floor or a through profile floor.

```
TYPE profile_select =
     SELECT (through_profile_floor, profile_floor);
END_TYPE;
```

4.5.8.2.2 Through profile floor

A through profile floor describes the bottom condition of a shape profile which is open.

```
ENTITY through_profile_floor;
END_ENTITY;
```

4.5.8.2.3 Profile floor

A profile floor describes the bottom condition of a shape profile which may be flat or any arbitrary shape.

```
ENTITY profile_floor
ABSTRACT SUPERTYPE OF(ONEOF(general_profile_floor, planar_profile_floor));
  floor_radius :             OPTIONAL numeric_parameter;
  start_or_end :             BOOLEAN;
END_ENTITY;
```

floor_radius: The radius of curvature for an arc between the bottom and the sides.

start_or_end: If true, the profile_floor is positioned at the end of a shape_profile. If false, its is at the start of the shape_profile.

4.5.8.2.4 General profile floor

A general_profile_floor is a type of profile_floor that specifies an enclosed area bounded by an arbitrary shape.

```
ENTITY general_profile_floor
SUBTYPE OF (profile_floor);
  floor :               face;
END_ENTITY;
```

floor: Specification of the face at the bottom of a shape_profile, adjacent to all the shape_profile walls.

4.5.8.2.5 Planar profile floor

A planar_profile_floor is a type of a profile_floor that characterises the bottom of a shape_profile which is flat.

```
ENTITY planar_profile_floor
SUBTYPE OF (profile_floor);
  floor :              plane;
END_ENTITY;
```

floor: Specification of a planar face at the bottom of a shape_profile.

4.5.8.2.6 General shape profile

A general_shape_profile is a type of shape_profile that is a volume of arbitrary shape which defines a portion of the workpiece.

```
ENTITY general_shape_profile
SUBTYPE OF (shape_profile);
  profile_boundary :   profile;
END_ENTITY;
```

profile_boundary: Specification of the outline of the shape_profile feature. It defines an area that may or may not be entirely enclosed.

4.5.8.2.7 Partial circular shape profile

A partial_circular_shape_profile is a type of shape_profile that defines a volume that is not enclosed on all sides.

```
ENTITY partial_circular_shape_profile
SUBTYPE OF (shape_profile);
  open_boundary :      partial_circular_profile;
END_ENTITY;
```

open_boundary: Specification of the outline of the shape_profile feature. It defines an area that shall be circular and shall not be enclosed.

4.5.8.2.8 Circular closed shape profile

A circular_closed_shape_profile is a type of shape_profile that defines a completely enclosed volume. This may have a thread.

```
ENTITY circular_closed_shape_profile
SUBTYPE OF (shape_profile);
  closed_boundary :      circular_closed_profile;
END_ENTITY;
```

closed_boundary: Specification of the outline of the shape_profile feature. It defines an area that shall be enclosed and circular.

4.5.8.2.9 Rectangular open shape profile

A rectangular_open_shape_profile is a type of shape_profile that is an open profile with opposite sides that are of equal length and with one side that does not make contact with the workpiece.

```
ENTITY rectangular_open_shape_profile
SUBTYPE OF (shape_profile);
  open_boundary :        square_u_profile;
END_ENTITY;
```

open_boundary: Specification of the outline or shape that is an enclosed area that is open on one side.

4.5.8.2.10 Rectangular closed shape profile

A rectangular_closed_shape_profile is a type of shape_profile that is an enclosed volume with opposite sides that are equal in length.

```
ENTITY rectangular_closed_shape_profile
SUBTYPE OF (shape_profile);
  closed_boundary :      rectangular_closed_profile;
END_ENTITY;
```

closed_boundary: Specification of the outline or shape that is an enclosed area that has a completely closed profile.

4.5.9 Round hole

This entity defines both holes and threaded holes. The feature_placement of a hole is the center point at the surface, i. e. the hole is located at x = y = 0 in the local co-ordinate system. The specified (positive) depth causes the tool to advance into the hole in negative z direction.

Note that the bottom of the hole is not considered in the hole's depth.

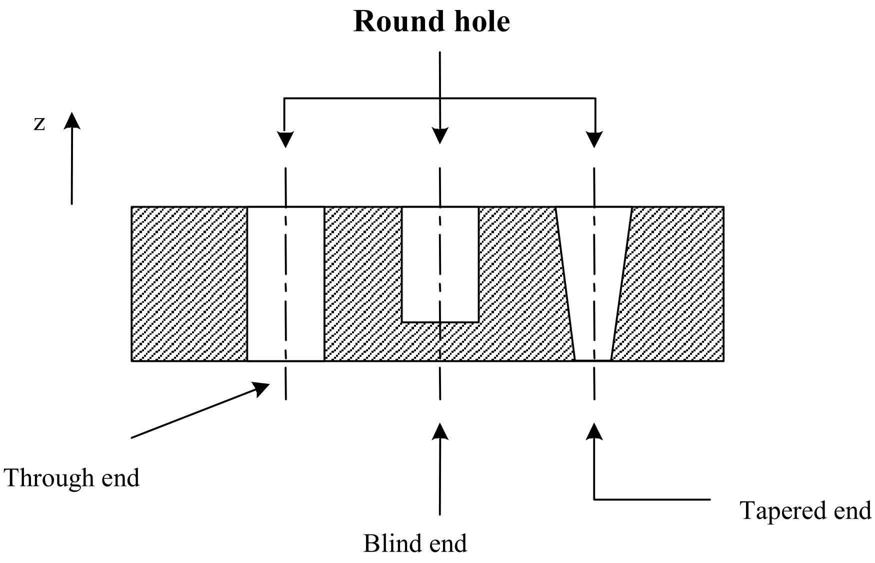


Figure 15.Hole types

```
ENTITY round_hole
  SUBTYPE OF (machining_feature);
  diameter:                toleranced_length_measure;
  change_in_diameter:      OPTIONAL taper_select;
  bottom_condition:        hole_bottom_condition;

END_ENTITY;
```

diameter: The diameter of the hole. The diameter is measured in the xy-plane where z = 0.

change_in_diameter: An optional parameter used to specify holes with a taper.

bottom_condition: Specification of the bottom of a hole.

4.5.9.1 Taper select

The taper select indicates the manner by which a taper is described. A taper may be described either by diameter or by angle.

```
TYPE taper_select = SELECT (diameter_taper, angle_taper);
END_TYPE;
```

4.5.9.1.1 Diameter taper

Entity to describe a taper by its final diameter.

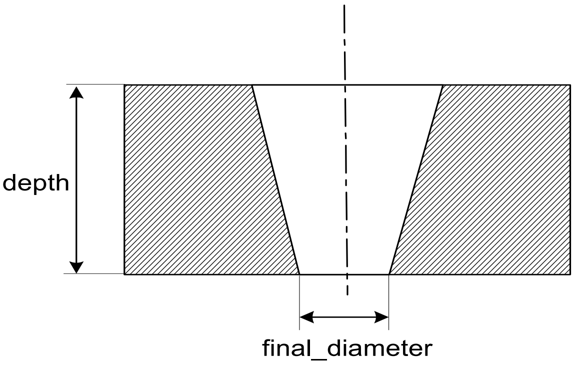


Figure 16.Diameter taper

```
ENTITY diameter_taper;
  final_diameter:        toleranced_length_measure;
END_ENTITY;
```

final_diameter: The final diameter of the tapered hole at the indicated depth.

4.5.9.1.2 Angle taper

Entity to describe a taper by its half angle.

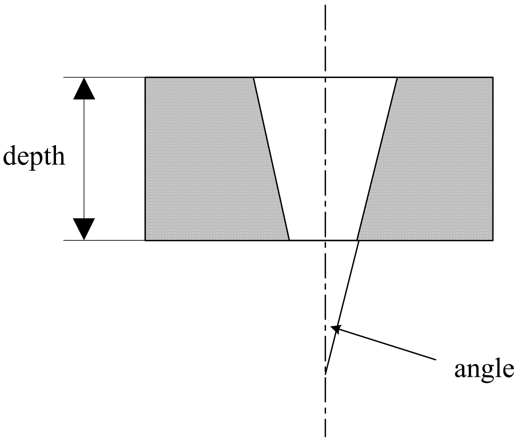


Figure 17.Angle taper

```
ENTITY angle_taper;
  angle:                 plane_angle_measure;
END_ENTITY;
```

angle: The angle of the tapered hole.

4.5.9.2 Hole bottom condition

Abstract supertype for the description of a bottom of a hole.

```
ENTITY hole_bottom_condition
  ABSTRACT SUPERTYPE OF (ONEOF (blind_bottom_condition,
    through_bottom_condition));
END_ENTITY;
```

4.5.9.2.1 Through bottom condition

Entity which describes a hole bottom which is open.

```
ENTITY through_bottom_condition
  SUBTYPE OF (hole_bottom_condition);
END_ENTITY;
```

4.5.9.2.2 Blind bottom condition

Supertype to describe different types of blind bottom conditions. The bottom may break through the bottom of the workpiece, but is not entirely open. The depth of the hole is the length of the cylindrical section of the hole.

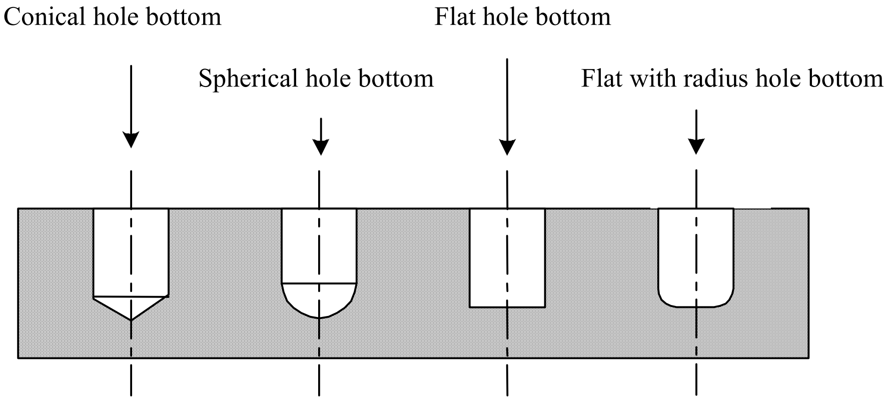


Figure 18.Bottom conditions of holes

```
ENTITY blind_bottom_condition
ABSTRACT SUPERTYPE OF (ONEOF(flat_hole_bottom, flat_with_radius_hole_bottom,
  spherical_hole_bottom, conical_hole_bottom))
  SUBTYPE OF (hole_bottom_condition);
END_ENTITY;
```

4.5.9.2.2.1 Flat hole bottom

A hole with a flat bottom.

```
ENTITY flat_hole_bottom
  SUBTYPE OF (blind_bottom_condition);
END_ENTITY;
```

4.5.9.2.2.2 Flat with radius hole bottom

A hole with a flat bottom which has a radius.

```
ENTITY flat_with_radius_hole_bottom
  SUBTYPE OF (blind_bottom_condition);
  corner_radius:         toleranced_length_measure;
END_ENTITY;
```

corner_radius: The radius of the corner in the bottom.

4.5.9.2.2.3 Spherical hole bottom

A hole with a spherical bottom.

```
ENTITY spherical_hole_bottom
  SUBTYPE OF (blind_bottom_condition);
  radius:                 toleranced_length_measure;
END_ENTITY;
```

radius: The radius of the spherical hole bottom.

4.5.9.2.2.4 Conical hole bottom

A hole with a conical bottom, as manufactured with a standard drill. A conical bottom is a constant decrease in diameter until the diameter is ‘zero’, or until it becomes tangent to a tip radius.

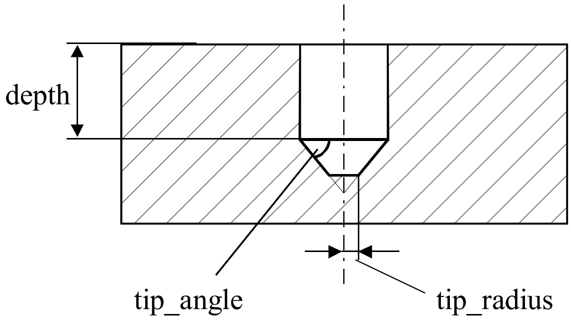


Figure 19.Conical hole bottom

```
ENTITY conical_hole_bottom
  SUBTYPE OF (blind_bottom_condition);
  tip_angle:              plane_angle_measure;
  tip_radius:             OPTIONAL toleranced_length_measure;
END_ENTITY;
```

tip_angle: The angle of the tip.

tip_radius: Optional attribute for specification of a tip radius. Default is zero.

4.5.10 Toolpath feature

To enable the definition of tool movements not covered by the previously listed features, the toolpath_ feature has been introduced. Explicit toolpaths shall be assigned to the operations associated with this feature.

```
ENTITY toolpath_feature
  SUBTYPE OF (machining_feature);
END_ENTITY;
```

This feature shall only be used if instead of a geometry like an explicit feature, a region, etc. there is only to be described a toolpath. Regular manufacturing features should be used whenever possible. Note that **all** manufacturing features and **all** operations offer the possibility to assign manually programmed toolpaths to them. Such toolpaths will always override any automatic generation of tool movements by the machine controller and offer the CAM system complete control over the machine.

So even if some special tool movement is needed it is preferred to use a regular manufacturing feature with the associated workingsteps and attach explicit tool paths to the workingsteps.

4.5.11 Boss

A boss is a feature which has to be related to an other feature. No separate workingstep can be assigned to manufacture a boss (i.e. it is not possible to first cut the feature and then cut the boss). Instead, the boss is that material which remains unworked after the machining of a feature with a boss. For this reason, boss is not an independent machining feature but exists only as an attribute.

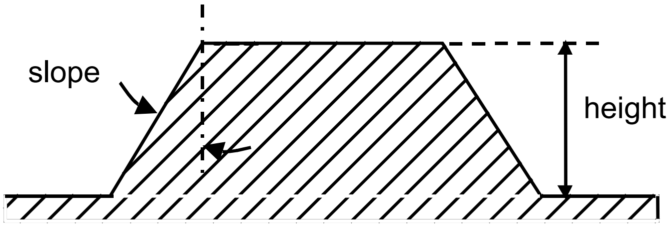


Figure 20.Boss

```
ENTITY boss
SUBTYPE OF(machining_feature);
  its_boundary:          closed_profile;
  slope:                 OPTIONAL plane_angle_measure;
(*
Informal propositions:
- The entire profile lies in the same plane.
- The profile is not self-intersecting.
*)
END_ENTITY;
```

its_boundary: The contour of the boss at the top of the boss. The contour at the bottom may be calculated by traversing the slope along the contour. Also, for the boss to have a well-defined height, its_boundary should be parallel to the bottom of the parent feature. When travelling along the profile based as defined by its sense, the material is to the left of the profile.

slope: The slope of the contour of the boss relative to the local z axis of the parent feature. The default is 0 degrees.

4.5.12 Spherical cap

This feature is circular about an axis of rotation. It consists of all points at a given distance from a point constituting its center. The center is defined by it's placement, that is not located in the feature's highest point, but in the sphere's center.

```
ENTITY spherical_cap
SUBTYPE OF (machining_feature);
  internal_angle:        numeric_parameter;
  radius:                numeric_parameter;
END_ENTITY;
```

internal_angle: The size of an angle from an axis for defining a portion of a sphere.

radius: The constant distance from a point for defining a sphere.

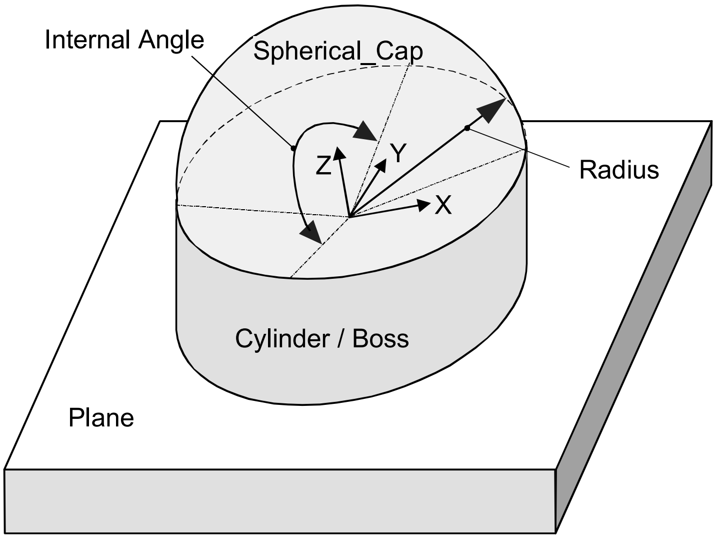


Figure 21. Spherical cap

4.5.13 Rounded end

A rounded end is a partially circular shape passed along a linear path.

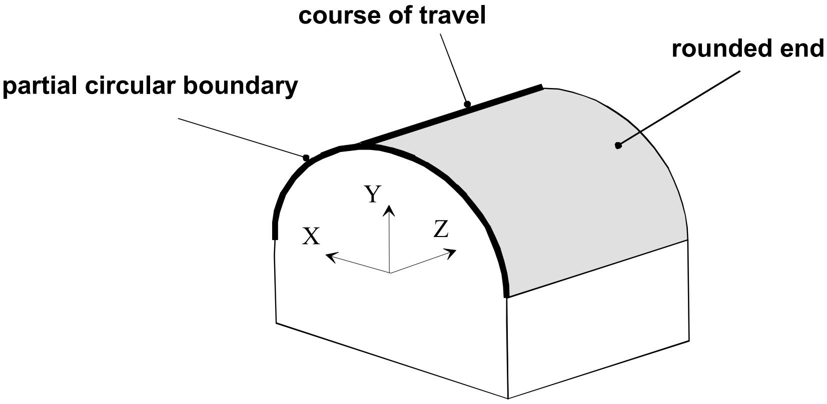


Figure 22. Rounded end

```
ENTITY rounded_end
SUBTYPE OF (machining_feature);
  course_of_travel:          linear_path;
  partial_circular_boundary: partial_circular_profile;
END_ENTITY;
```

course_of_travel: A straight line with magnitude and direction.

partial_circular_boundary: The arc that when swept along a path defines the area on a workpiece for volume removel.

4.5.14 Compound feature

A compound feature is a feature composed of two or more features. As opposed to the replicate feature, there is no regular spacing between the elements of the compound feature. In general, they will also not be of the same type.

The volume of removed material of the compound feature is the union of all volumes of the elements of the compound feature. It is illegal to create a compound feature where the volumes of the elements are not connected in such a way that their union can be machined.

Note that the individual elements of the compound feature are associated with machining operation. In addition the compound feature includes by the attribute its_operations all operations of its minor elements.

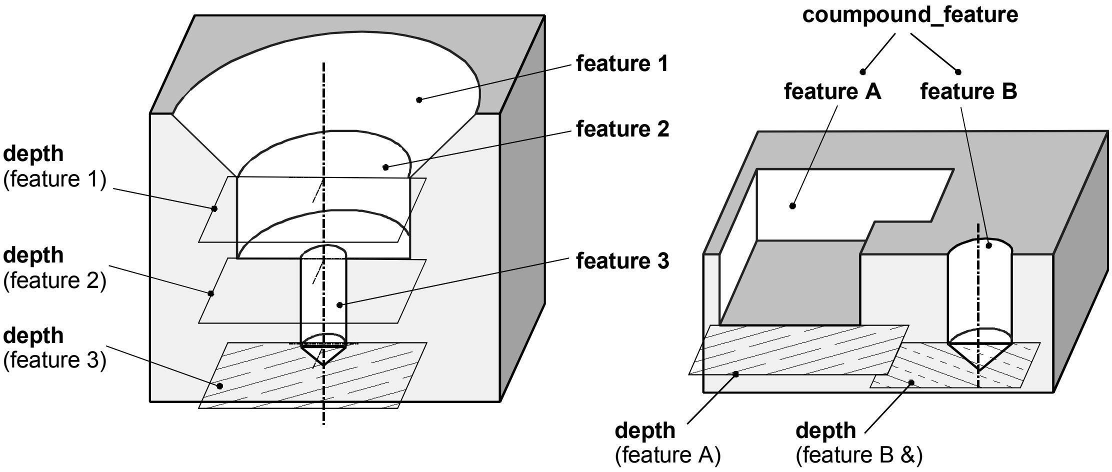


Figure 23.Compound feature

```
ENTITY compound_feature
  SUPERTYPE OF (ONEOF(counterbore_hole, countersunk_hole))
  SUBTYPE OF (two5D_manufacturing_feature);
  elements: SET [2:?] OF compound_feature_select;
WHERE
  WR1: SIZEOF(QUERY(e <* elements | SIZEOF(e\manufacturing_feature.its_operations)
    <> 0)) = 0;
END_ENTITY;
```

elements: Set of features composing the compound feature. Note that the feature_placement attribute of the elements will specify their position relative to the local co-ordinate system of the compound feature, not relative to the workpiece co-ordinates.

4.5.14.1 Compound feature select

Selection of the elements of compound_feature.

```
TYPE compound_feature_select = SELECT(
  machining_feature, transition_feature
  );
END_TYPE;
```

4.5.14.2 Counterbore hole

The counterbore hole is a type of compound feature that consists of two round_holes. The round hole closer to the counterbore's placement has to have a larger diameter than the one more inside of the material. The bottom of the former round_hole shall mate with the top of the latter round_hole. The orientation of counterbore_hole shall be the same as the orientation of the latter round_hole. Both round_holes shall be co-axial.

```
ENTITY counterbore_hole
   SUBTYPE OF (compound_feature);
WHERE
   WR1: SIZEOF(elements) =2;
   WR2: (SIZEOF(QUERY ( it <* SELF.elements |
        (('MACHINING_SCHEMA.ROUND_HOLE' IN TYPEOF(it))) )) = 2);
   WR3: SELF.elements[1].diameter.theoretical_size <>
        SELF.elements[2].diameter.theoretical_size;
END_ENTITY;
```

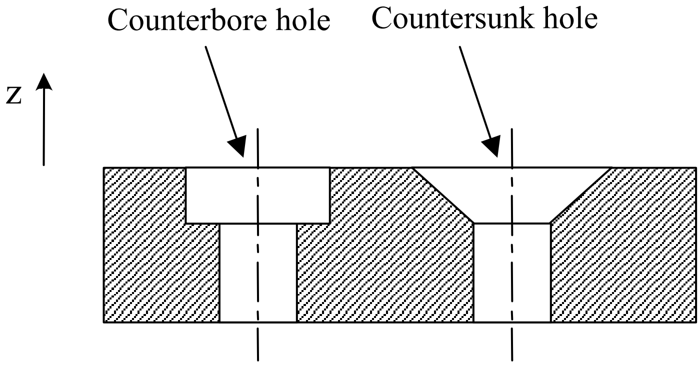


Figure 24. Counterbore and countersunk hole

4.5.14.2.1 Countersunk hole

The countersunk hole is a type of compound feature that consists of two round_holes. The bottom of the round_hole closer to the countersunk's placement shall mate with the top of the round_hole more inside of material. The taper of the former round_hole shall be larger than the diameter of the latter round_hole, decreasing to the same diameter at the point where the two holes join. The orientation of countersunk_hole shall be the same as the orientation of the latter round_hole. Both round_holes shall be co-axial.

```
ENTITY countersunk_hole
  SUBTYPE OF (compound_feature);
WHERE
   WR1: SIZEOF(elements) =2;
   WR2: (SIZEOF(QUERY ( it <* SELF.elements |
        (('MACHINING_SCHEMA.ROUND_HOLE' IN TYPEOF(it))) )) = 2);
   WR3: SELF.elements[1].diameter.theoretical_size <>
        SELF.elements[2].diameter.theoretical_size;
   WR4: NOT EXISTS(SELF.elements[1].change_in_diameter) AND
        EXISTS(SELF.elements[2].change_in_diameter);
END_ENTITY;
```

4.5.15 Replicate feature

A replicate is an assembly of a number of similar features, e.g. a circle of holes or a mesh of holes. The feature is described only once and the number and spacing of the feature is described. Note that the attribute feature_placement of replicate_base_feature will describe the location of the feature relative to its position in the replication pattern as specified by the subtypes of replicate_feature.

```
ENTITY replicate_feature
  ABSTRACT SUPERTYPE OF (ONEOF(rectangular_pattern, circular_pattern,
    general_pattern))
  SUBTYPE OF (two5D_manufacturing_feature);
  replicate_base_feature: two5D_manufacturing_feature;
END_ENTITY;
```

replicate_base_feature: The feature which forms the basis of the replicate feature.

4.5.15.1 Circular pattern

A circular pattern of features. A complete circle of features is a special case of circular pattern.

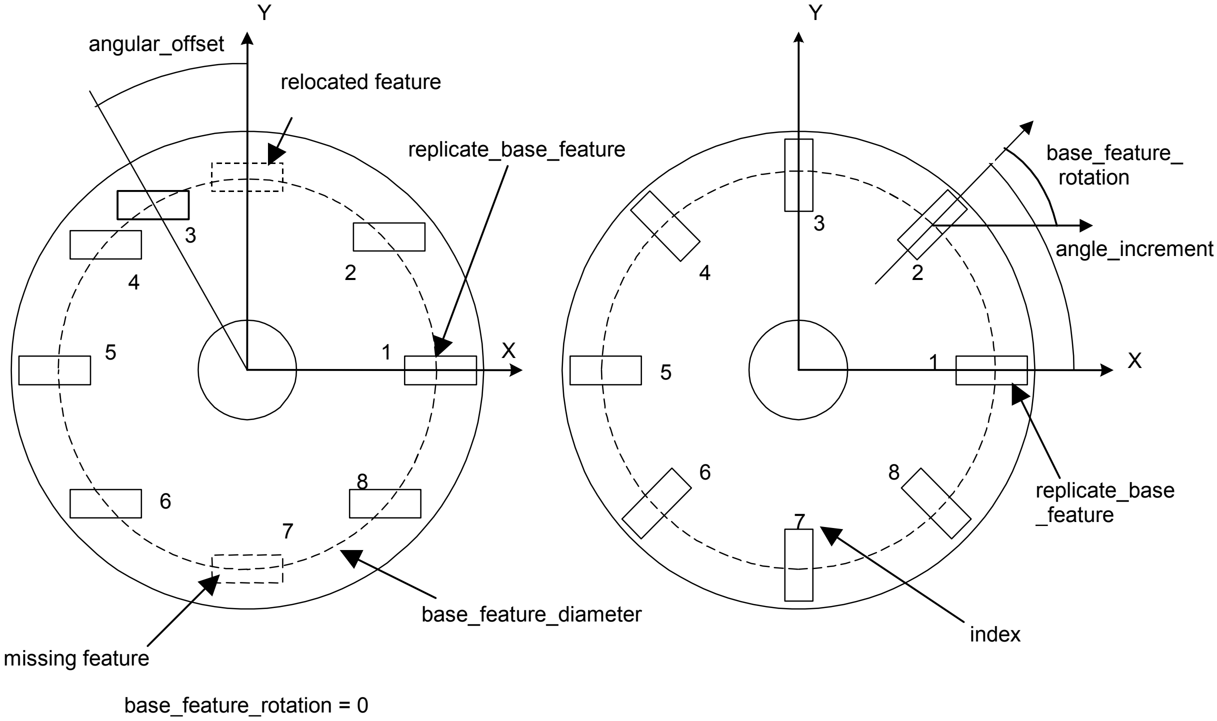


Figure 25.Circular Pattern

```
ENTITY circular_pattern
  SUBTYPE OF(replicate_feature);
  angle_increment:        plane_angle_measure;
  number_of_feature:      INTEGER;
  relocated_base_feature: SET[0:?] OF circular_offset;
  missing_base_feature:   SET[0:?] OF circular_omit;
  base_feature_diameter:  OPTIONAL toleranced_length_measure;
  base_feature_rotation:  plane_angle_measure;
END_ENTITY;
```

angle_increment: Angle between two elements of the pattern. A positive angle means a counter-clockwise increment when looking towards the plane in negative z direction.

number_of_feature: The total number of features in the replicate_feature. The maximum index of the circular pattern is calculated from number_of_feature plus the number of elements in

	the missing_base_feature set. The maximum index times angle_increment should not exceed 360° or the behaviour will be undefined.
relocated_base_feature:	Optional description of relocated features.
missing_base_feature:	Optional description of omitted features.
base_feature_diameter:	The diameter of the circular pattern. It has to be specified if rotate_feature is "false". If it is "true", the diameter can be calculated from the feature position and the location of the center.
base_feature_rotation	Specification of the angle to rotate one element in regard to the orientation of the previous element. The previous element is located in mathematical negative direction (counter-clockwise) to the current element.

4.5.15.1.1 Circular offset

Definition of elements offset from the circle of elements.

```
ENTITY circular_offset;
  angular_offset:       plane_angle_measure;
  index         :       INTEGER;
END_ENTITY;
```

angular_offset:	Offset of angle of the relocated element.
index:	Number of the relocated element.

4.5.15.1.2 Circular omit

Definition of elements omitted from the circle of elements.

```
ENTITY circular_omit;
  index:                INTEGER;
END_ENTITY;
```

index:	Number of the element to be omitted.

4.5.15.2 Rectangular pattern

The description of elements arranged in a rectangular pattern. This may be either a grid of elements with n rows and m columns and a total number of n x m elements or a single line of m elements (n=1).

```
ENTITY rectangular_pattern
  SUBTYPE OF(replicate_feature);
  spacing:                 toleranced_length_measure;
  its_direction:           direction;
  number_of_rows:          INTEGER;
  number_of_columns:       INTEGER;
  row_spacing:             OPTIONAL toleranced_length_measure;
  row_layout_direction:    OPTIONAL direction;
  relocated_base_feature: SET[0:?] OF rectangular_offset;
  missing_base_feature:    SET[0:?]OF rectangular_omit;
  WHERE
   WR1: ( (SELF.number_of_rows > 1 )
         AND EXISTS(SELF.row_spacing)
         AND EXISTS(SELF.row_layout_direction)
         );
END_ENTITY;
```

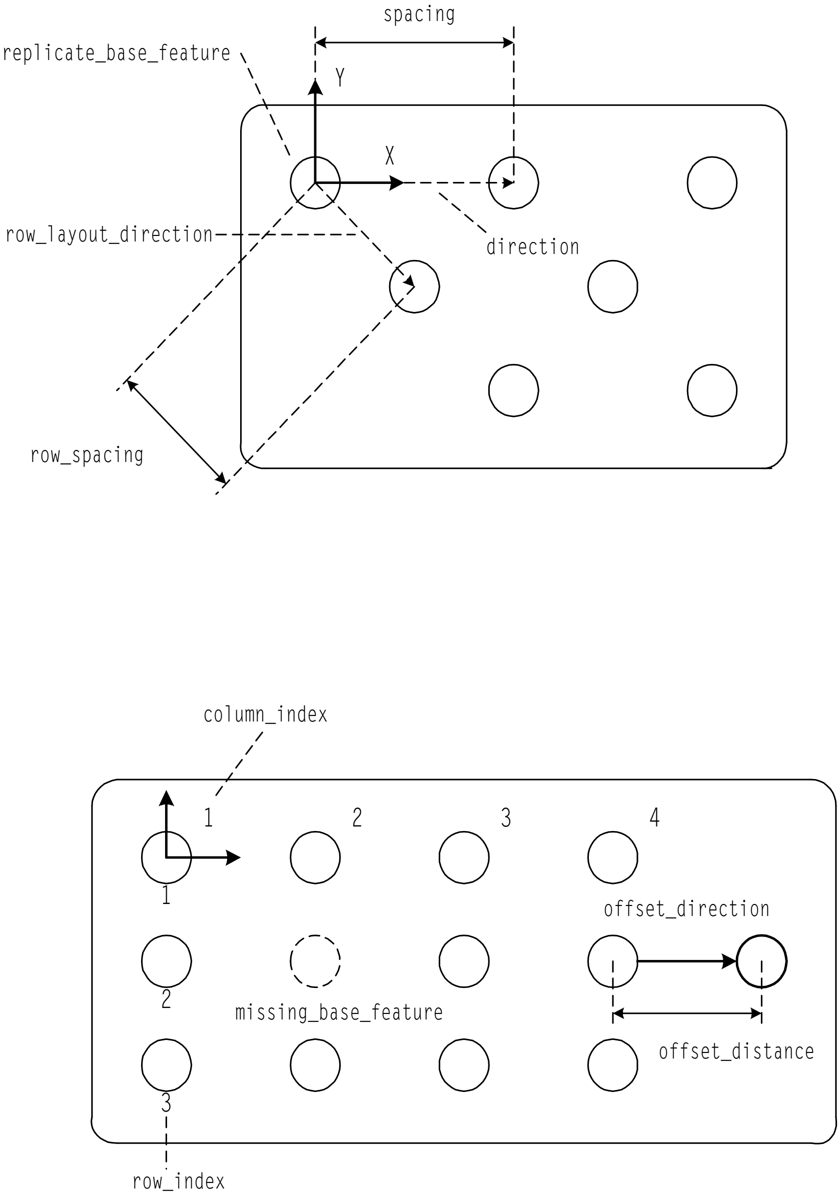


Figure 26.Rectangular pattern

spacing: The spacing of the columns of the pattern. If there is only one row, this is also the spacing between the elements.

its_direction: Direction of the first row measured from the direction given in feature_placement. It points into the direction of the row, describing the order to manufacture the replicated feature.

number_of_rows: The number of rows. Default is one, i. e. the rectangular pattern is a line of features. If number_of_rows is larger than one, the attributes row_spacing and row_layout_direction need to be specified.

number_of_columns: The number of columns.

row_spacing: The spacing of the rows (optional, needed if number_of_rows is larger than one).

row_layout_direction: Optional description of the direction of the first column measured from the direction given in feature_placement. It points into the direction of the next row.

relocated_base_feature: Optional description of relocated features.

missing_base_feature: Optional description of missing features.

Note that the number of features can be calculated from the number_of_rows, number_of_columns, missing_base_feature, and relocated_base_feature.

4.5.15.2.1 Rectangular_offset

Entity to describe the position of a single relocated element whitin a rectangular pattern of elements.

```
ENTITY rectangular_offset;
  offset_direction:      direction;
  offset_distance:       length_measure;
  row_index:             INTEGER;
  column_index:          INTEGER;
END_ENTITY; -- rectangular_offset
```

offset_direction: Direction of the offset of the element.

offset_distance: Amount of offset of the element.

row_index: Row of the offset element.

column_index: Column of the offset element.

4.5.15.2.2 Rectangular omit

Entity to omit one or more elements of a rectangular pattern of elements.

```
ENTITY rectangular_omit;
  row_index:             INTEGER;
  column_index:          INTEGER;
END_ENTITY; -- rectangular_omit
```

row_index: Row of the omitted element.

column_index: Column of the omitted element.

4.5.15.3 General pattern

Definition of a general list of identical elements (used i.e. for identical holes positioned arbitrarily).

```
ENTITY general_pattern
  SUBTYPE OF(replicate_feature);
  replicate_locations:  LIST [2:?] OF axis2_placement_3d;
END_ENTITY;
```

replicate_locations: List of the placement of the features relative to the local co-ordinate system of general_pattern. The order of the features to be machined is given by the sequence of the features' placements within the list.

4.5.16 Transition feature

A transition feature may be added at the border of two features. Example: An edge round or a chamfer between two planar faces or between a planar face and a pocket. Unlike in ISO 10303-224, only those transition features which are generated using additional tool movements are considered here. Example: An edge round between a pocket and a planar face needs an additional manufacturing operation (workingstep). On the contrary, a fillet between the sides and the bottom of a pocket results from the geometry of the tool and thus needs no additional tool movement. It is therefore not considered in this context.

```
ENTITY transition_feature
  ABSTRACT SUPERTYPE OF (ONEOF(chamfer, edge_round))
  SUBTYPE OF (manufacturing_feature);
  first_feature:          machining_feature;
  second_feature:         machining_feature;
END_ENTITY;
```

first_feature: The first of the two features connected by the transition feature.

second_feature: The second of the two features connected by the transition feature.

4.5.16.1 Chamfer

A chamfer is one of the two defined transition features for 2½D-machining This chamfer is always an *outer* chamfer as only this can be generated in an separate manufacturing operation.

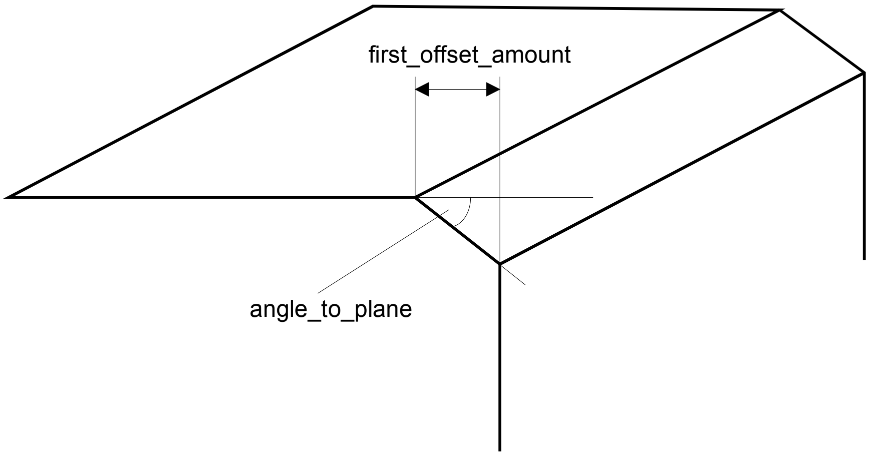


Figure 27.Chamfer

```
ENTITY chamfer
  SUBTYPE OF (transition_feature);
  angle_to_plane:        plane_angle_measure;
  first_offset_amount:   toleranced_length_measure;
END_ENTITY;
```

angle_to_plane: Angle between the first feature and the chamfer face measured from the first feature.

first_offset_amount: The offset of the chamfer measure from the edge of the first feature.

4.5.16.2 Edge round (fillet)

An edge round is the other of the two defined transition features. In 2½D-machining, it is generated using a contoured tool which has to be selected according to the given fillet radius. The edge_round is always an **outer** fillet, since only this type of fillet can be manufactured in an extra manufacturing operation.

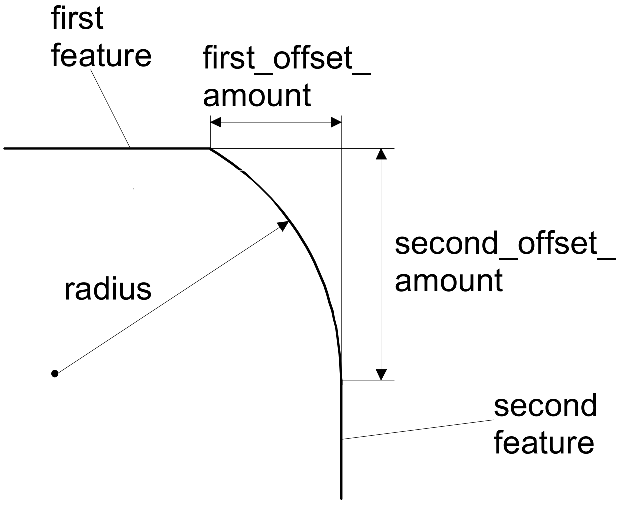


Figure 28.Edge round

```
ENTITY edge_round
SUBTYPE OF (transition_feature);
  radius:                 toleranced_length_measure;
  first_offset_amount:    OPTIONAL toleranced_length_measure;
  second_offset_amount:   OPTIONAL toleranced_length_measure;
END_ENTITY;
```

radius:	Radius of the edge round.
first_offset_amount:	The edge_round may or may not be tangent to the two features, this attribute specifies the location of the edge round (see figure).
second_offset_amount:	A non-tangent edge_round may or may not be symmetrical to the two features. If it is not symmetrical, this feature specifies the location of the edge round (see figure).

4.5.17 Thread

A thread is a ridge of a uniform section on the form of a helix on the external or internal surface of a cylinder. Each thread is either a catalogue_thread or a defined_thread.

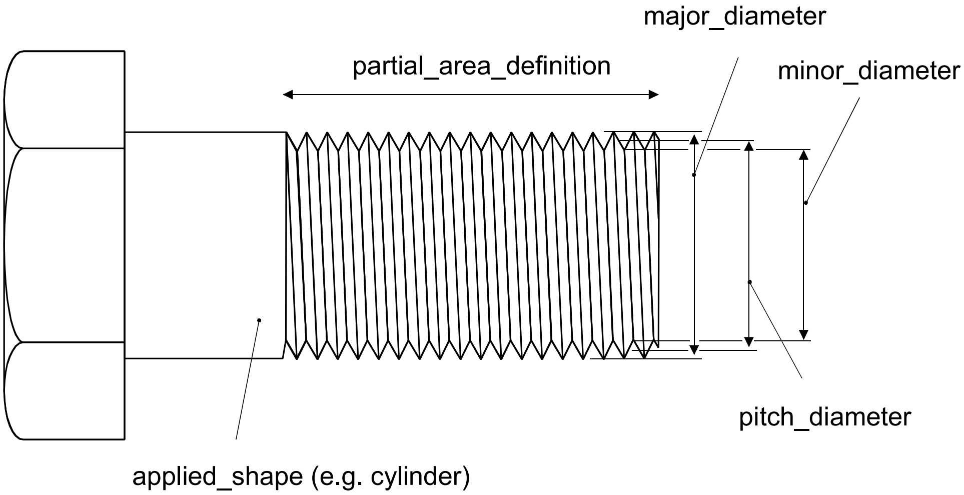


Figure 29.Thread

```
ENTITY thread
ABSTRACT SUPERTYPE OF(ONEOF(catalogue_thread, defined_thread))
SUBTYPE OF(machining_feature);
  partial_profile:       partial_area_definition;
  applied_shape:         SET[1:?] OF machining_feature;
  inner_or_outer_thread: BOOLEAN;
  qualifier:             OPTIONAL descriptive_parameter;
  fit_class:             descriptive_parameter;
  form:                  descriptive_parameter;
  major_diameter:        length_measure;
  number_of_threads:     numeric_parameter;
  thread_hand:           descriptive_parameter;
 WHERE
  WR1: ('MACHINING_SCHEMA.ROUND_HOLE' IN TYPEOF(applied_shape)) OR
       ('MACHINING_SCHEMA.CIRCULAR_CLOSED_SHAPE_PROFILE' IN TYPEOF(applied_shape))
      OR ('MACHINING_SCHEMA.BOSS' IN TYPEOF(applied_shape));
END_ENTITY;
```

partial_profile: Specification of limitations of a surface to be applied on the thread.

applied_shape: Physical shape of the workpiece that will define where the thread will be applied.

inner_or_outer_thread: A flag specifies whether or not a thread is applied as an internal thread or an external thread.

qualifier: Additional text information that describes a thread.

fit_class: The value for the type of fit specification. These types are distinguished from each other by the amount of the tolerance and allowance.

form: The definition of the shape of the thread. Various forms of threads are used to hold parts together, to adjust parts with reference to each other, or to transmit power.

major_diameter: The dimension of the largest diameter of the thread and is applied to both an internal and external thread.

number_of_threads: The thread pitch when used with metric unit of measure and the density of threads per inch when used with English unit of measure.

thread_hand: A description of whether the thread is right or left handed. When viewed toward an end, a right hand winds in a clockwise direction and a left hand winds in a counterclockwise direction.

4.5.17.1.1 Partial area definition

A partial_area_definition includes the limitations of a surface for applying a thread. It places a limitation on how much and where to apply the thread on the cylindrical shape.

```
ENTITY partial_area_definition;
  effective_length:       length_measure;
  placement:              axis2_placement_3D;
  maximum_length:         OPTIONAL length_measure;
END_ENTITY;
```

effective_length — The length of the thread which is usable.

placement: — The location and direction of the partial_area_definition.

maximum_length: — The dimension along a surface to apply a thread. It limits The dimensional distance limits the length along the surface axis for defining a thread.

4.5.17.2 Catalogue thread

A catalogue_thread is a type of thread that is defined by a document containing the information to create threads on a workpiece.

```
ENTITY catalogue_thread
SUBTYPE OF (thread);
  documentation:          specification;
END_ENTITY;
```

documentation: — Specification of the document that defines information pertaining to a thread.

4.5.17.3 Specification

A specification is a document that defines information pertaining to properties or processes for a workpiece or an aspect of a workpiece.

```
ENTITY specification;
  constraint:                 SET [0:?] OF specification_usage_constraint;
  specification_id:           text;
  specification_description: OPTIONAL text;
  specification_class:        OPTIONAL text;
END_ENTITY;
```

constraint: — A set of the restrictions on the specification.

specification_id: — A unigue identifier of the document.

specification_description: — A description of the content of the specification and any notes in human interpretable prose.

specification_class: — A section within a specification that is divided into classes.

4.5.17.3.1 Specification usage constraint

A specification_usage_constraint is a restriction on the application of information defined within a specification.

```
ENTITY specification_usage_constraint;
  element:                text;
  class_id:               text;
END_ENTITY;
```

element: The particular piece or area of information that is being restricted within the specification.

class_id: The data or range of data with respect to the element that defines the restriction imposed on the usage of the specification.

4.5.17.4 Defined thread

A defined_thread is a type of thread that is specified explicitly.

```
ENTITY defined_thread
SUBTYPE OF (thread);
  pitch_diameter:        length_measure;
  minor_diameter:        OPTIONAL length_measure;
  crest:                 OPTIONAL length_measure;
END_ENTITY;
```

pitch_diameter: The dimension of an imaginary cylinder passing through the thread so as to make equal the widths of the threads and the widths of the spaces cut by the cylinder.

minor_diamter: The dimension of the smallest diameter of the defined_thread and is applied to both an internal and external thread.

crest: The distance between the opposing points of the thread. It is formed by the intersection of the sides of the thread if extended, if necessary, beyond the top of the thread.

4.5.18 Profile

A profile is a planar outline used in the definition of a feature. A profile may be either open or closed. A profile shall be in the X-Y plane and may have an orientation that will position it in reference to the local coordinate system of a manufacturing_feature, which may require a profile as a part of its definition.

```
ENTITY profile
  ABSTRACT SUPERTYPE OF(ONEOF(closed_profile, open_profile));
  placement:                  OPTIONAL axis2_placement_3d;
END_ENTITY;
```

placement: Specification where to locate the profile in reference to the orientation of the manufacturing_feature. If not given, the orientation is at the zero point of the manufacturing_feature.

4.5.18.1 Open profile

An open_profile is a type of profile that is an outline or shape with no enclosing or confining bounds. The open ends of the profile may extend infinitely.

```
ENTITY open_profile
  ABSTRACT SUPERTYPE OF
  (ONEOF (linear_profile, square_u_profile, rounded_u_profile, tee_profile,
    vee_profile, partial_circular_profile, general_profile))
  SUBTYPE OF(profile);
END_ENTITY;
```

4.5.18.1.1 Linear profile

A linear_profile is a type of open_profile that is a straight line of a specified length. The linear_profile shall have orientation parallel to the X-axis.

```
ENTITY linear_profile
SUBTYPE OF (open_profile);
  profile_length:        numeric_parameter;
END_ENTITY;
```

profile_length: The length of the profile.

4.5.18.1.2 Square U profile

A square U profile is a type of open profile that is bounded by three lines. One is the base line and has a defined length. The other two lines begin at the ends of the base line, and extend infinitely at any obtuse or acute angle that is equal to or larger than a right angle. The corners of the square U profile need not be blended by a radius.

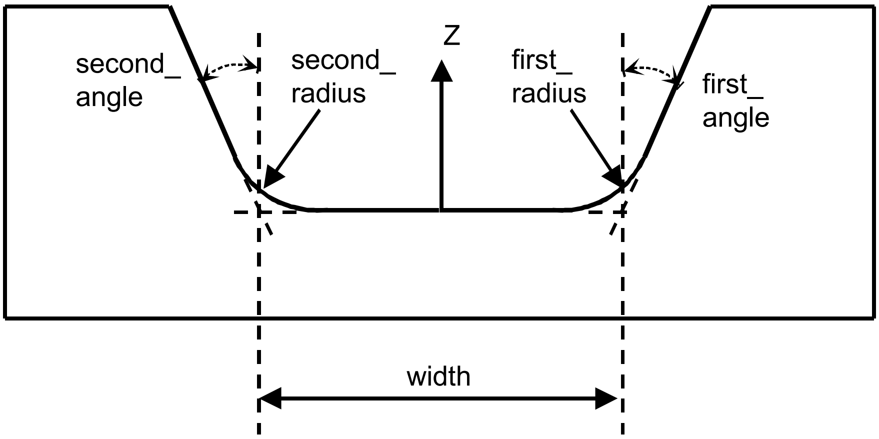


Figure 30.Square U profile

For a rectangular slot, all values except for the width shall be zero.

```
ENTITY square_u_profile
  SUBTYPE OF (open_profile);
  width:                 toleranced_length_measure;
  first_radius:          toleranced_length_measure;
  first_angle:           plane_angle_measure;
  second_radius:         toleranced_length_measure;
  second_angle:          plane_angle_measure;
END_ENTITY;
```

width: The size of the base line for a square U profile.

first_radius: The radius shape blend between one side of the profile and the base.

first_angle: The angle of the one side of the profile measured against the local z axis.

second_radius: The radius shape blend between the second side of the profile and the base.

second_angle: The angle of the another side of the profile measured against the local z axis.

4.5.18.1.3 Rounded U profile

A rounded U profile is a type of open profile that is a shape bounded by two parallel lines and a semicircle.

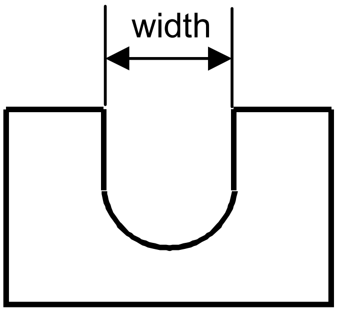


Figure 31.Rounded U profile

```
ENTITY rounded_u_profile
  SUBTYPE OF (open_profile);
  width:                 toleranced_length_measure;
END_ENTITY;
```

width: Distance across the rounded u profile.

4.5.18.1.4 T profile

A T profile is a type of open profile the cross-section of which has the shape of the letter "T". The attributes are explained in the figure.

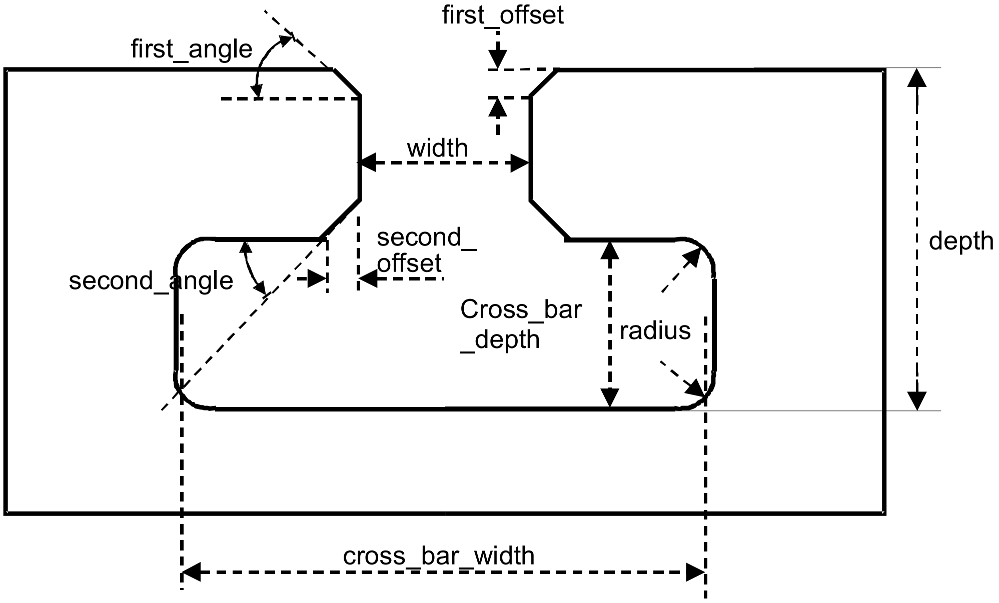


Figure 32.T profile

```
ENTITY tee_profile
  SUBTYPE OF (open_profile);
  first_angle:           plane_angle_measure;
  second_angle:          plane_angle_measure;
  cross_bar_width:       toleranced_length_measure;
  cross_bar_depth:       toleranced_length_measure;
  radius:                toleranced_length_measure;
  width:                 toleranced_length_measure;
```

```
    first_offset:          toleranced_length_measure;
    second_offset:         toleranced_length_measure;
  END_ENTITY;
```

first_angle: Angular measurement for creating a chamfer on the open end.

second_angle: Angular measurement for creating a chamfer between the stem and the cross bar parts of a T profile.

cross_bar_width: Width of the T cross bar.

cross_bar_depth: Depth of the T cross bar.

radius: Arc size for blending the sides of a tee cross bar.

width: Width of the T stem.

first_offset: Distance from the edge of the T stem to create a chamfer on the open end.

second_offset: Distance from the edge of the T stem to create a chamfer a distance from the edge of a surface to the finish of a chamfer.

4.5.18.1.5 V profile

A V profile is a type of open profile that is a shape bounded by two lines that connect at a point and extend infinitely. The enclosed angle is less than 180 degrees.

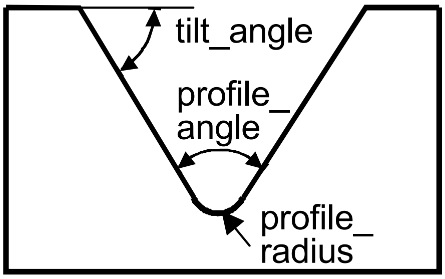


Figure 33.V profile

```
  ENTITY vee_profile
    SUBTYPE OF (open_profile);
    profile_radius:        toleranced_length_measure;
    profile_angle:         plane_angle_measure;
    tilt_angle:            plane_angle_measure;
  END_ENTITY;
```

profile_radius: Size of the blend radius at the point of the V or where the two sides come together.

profile_angle: The size of the angle between the two sides of the V profile.

tilt_angle: The size of the angle between one side of the V profile and the surrounding workpiece surface.

4.5.18.1.6 Partial circular profile

A partial circular profile is a type of open_profile that is specified by an arc. The arc shall be a constant radius swept about a point. The orientation of the profile shall be positioned at the origin of the are, with one end point of the arc on the X-axis.

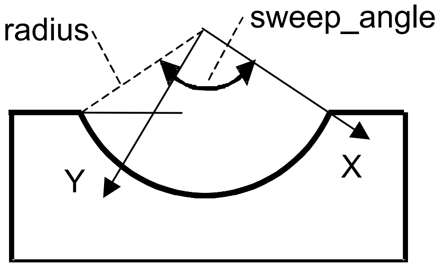


Figure 34.Partial circular profile

```
ENTITY partial_circular_profile
  SUBTYPE OF (open_profile);
  radius:             toleranced_length_measure;
  sweep_angle:        plane_angle_measure;
END_ENTITY;
```

radius: The size of the arc to define a partial circular profile.

sweep_angle: The size of the angle to define a circular shaped profile.

4.5.18.1.7 General profile

Any profile as defined by an arbitrary contour.

```
ENTITY general_profile
  SUBTYPE OF (open_profile);
  its_profile:          bounded_curve;
(*
Informal propositions:
- The entire profile lies in the local yz plane.
- The profile is not self-intersecting.
*)
END_ENTITY;
```

its_profile: A contour describing the general profile.

4.5.18.2 Closed profile

A closed_profile is a type of profile that is an outline or shape that bounds an enclosed area with no opening.

```
ENTITY closed_profile
  ABSTRACT SUPERTYPE OF
  (ONEOF (rectangular_closed_profile, circular_closed_profile, ngon_profile,
    general_closed_profile))
  SUBTYPE OF(profile);
END_ENTITY;
```

4.5.18.2.1 Rectangular closed profile

A rectangular_closed_profile is a type of closed_profile that is an enclosed area bounded by four sides with opposite sides equal in length and corners at 90 degrees. The orientation is at the center of the rectangle, the X-axis is parallel to the length of the rectangle and the Y-axis is parallel to the width.

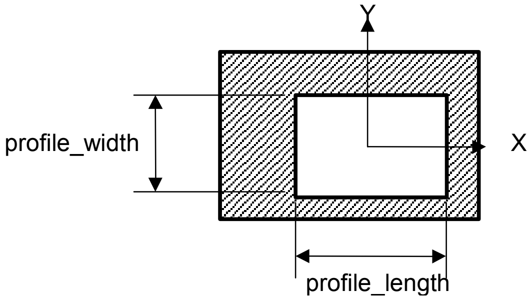


Figure 35.Rectangular closed profile

```
ENTITY rectangular_closed_profile
  SUBTYPE OF(closed_profile);
  profile_width:        toleranced_length_measure;
  profile_length:       toleranced_length_measure;
END_ENTITY;
```

profile_width: The size of the shortest side of the rectangular_profile.

profile_length: The size of the longest side of the rectangular_profile.

4.5.18.2.2 Circular closed profile

A circular_closed_profile is a type of closed_profile that is an enclosed area bounded by a circle. The orientation is at the center of the circle.

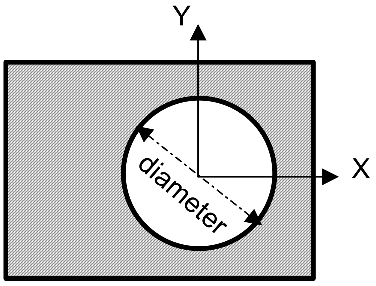


Figure 36.Circular closed profile

```
ENTITY circular_closed_profile
  SUBTYPE OF(closed_profile);
  diameter:             toleranced_length_measure;
END_ENTITY;
```

diameter: The distance across the circular_closed_profile.

4.5.18.2.3 Ngon profile

An ngon_ profile is a type of closed_profile that is an enclosed area bounded by three or more connected straight line sides. The orientation is at the center of the profile with one side of the ngon parallel to the X-axis crossing the Y-axis at a negative value.

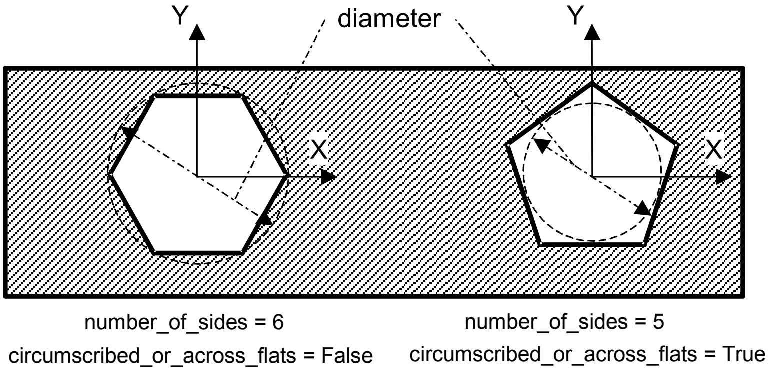

Y
diameter
Y
X
X
number_of_sides = 6
circumscribed_or_across_flats = False
number_of_sides = 5
circumscribed_or_across_flats = True

Figure 37.Ngon profile

```
ENTITY ngon_profile
  SUBTYPE OF(closed_profile);
  diameter:             toleranced_length_measure;
  number_of_sides:      INTEGER;
  circumscribed_or_across_flats: BOOLEAN;
END_ENTITY;
```

diameter: The size of the circle which surrounds the ngon or is surrounded by the ngon. It depends on the flag of circumscribed_or_across_flats.

number_of_sides: Number of sides needed for the ngon.

circumscribed_or_across_flats:If false, the ngon is surrounded by a circle with the specified diameter. If true, the ngon surrounds a circle with the specified diameter.

4.5.18.2.4 General closed profile

A general_closed_profile is a type of closed_profile that is an enclosed area bounded by a arbitrary shape. The orientation is defined by the explicit geometry of the shape.

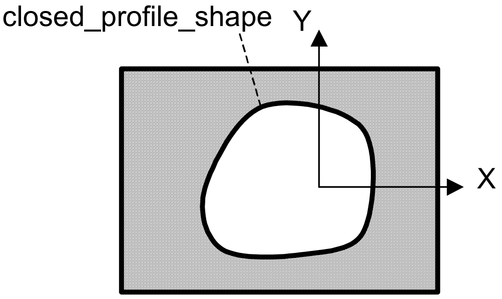


Figure 38.General closed profile

```
ENTITY general_closed_profile
  SUBTYPE OF(closed_profile);
  closed_profile_shape:  bounded_curve;
  END_ENTITY;
```

closed_profile_shape: A bounded curve that defines the arbitrary shape of the profile.

4.5.19 Travel path

A travel_path is a continuous set of curves that define a direction of travel. These curves do not intersect or duplicate themselves. A travel_path may have its own orientation in reference to the local coordinate system of the machining_feature which requires it as a part of a feature definition.

```
  ENTITY travel_path
  ABSTRACT SUPERTYPE OF(ONEOF(general_path, linear_path, circular_path));
  placement:                   OPTIONAL axis2_placement_3d;
  END_ENTITY;
```

placement: Specification where to locate the travel_path in reference to the orientation of the manufacturing_feature. If not given, the orientation is at the zero point of the manufacturing_feature.

4.5.19.1 General path

A General_path is a type of path that is a direction of travel along an arbitrary curve.

```
  ENTITY general_path
  SUBTYPE OF(travel_path);
    swept_path:             bounded_curve;
  END_ENTITY;
```

swept_path: A continuous set of curves that define an arbitrary direction of travel.

4.5.19.2 Linear path

A linear_path is a type of path that is a direction of travel along a line.

```
  ENTITY linear_path
  SUBTYPE OF(travel_path);
    distance:               toleranced_length_measure;
    its_direction:          direction;
  END_ENTITY;
```

distance: The length of the path.

its_direction: A vector which indicates the direction of the path starting from the path placement.

4.5.19.3 Circular path

The cicular_path is a type of path that is a direction of travel along an arc of constant radius around the Z-axis of the feature.

```
  ENTITY circular_path

  ABSTRACT SUPERTYPE OF(ONEOF(complete_circular_path, partial_circular_path))
  SUBTYPE OF(travel_path);
    radius:                 toleranced_length_measure;
  END_ENTITY;
```

radius: The constant distance from an axis for the circular_path

4.5.19.3.1 Complete circular path

A complete_circular_path is a type of circular_path that is a direction of travel that begins and ends the same point on the arc.

```
ENTITY complete_circular_path
SUBTYPE OF(circular_path);
END_ENTITY;
```

4.5.19.3.2 Partial circular path

A partical_circular_path is a type of circular_path that is a direction of travel along an arc of constant radius around an axis. The path shall begin and end at different points on the arc.

```
ENTITY partial_circular_path
SUBTYPE OF(circular_path);
  sweep_angle:           plane_angle_measure;
END_ENTITY;
```

sweep_angle: The size of the angle to define an arc shaped path.

4.5.20 Surface texture parameter

A Surface_texture_parameter is a combination of a parameter name and possibly indices describing one particular parameter of a surface texture such as roughness or waviness

```
ENTITY surface_texture_parameter;
  its_value:       parameter_value;
  parameter_name:        label;
  measuring_method:      identifier;
  parameter_index:       OPTIONAL identifier;
  applied_surfaces:      SET [1:?] OF machined_surface;
END_ENTITY;
```

its_value: The value of the surface_texture_parameter.parameter_name:The name of the surface_texture

measuring_method: The measuring_method specifies the method or standard that describes the method used to characterize the Surface_texture. Where applicable the following values shall be used: 'ISO 4287': The used surface texture parameters are defined in ISO 4287; 'ISO 12085': The used surface texture parameters are defined in ISO 12085; 'ISO 13565': The used surface texture parameters are defined in ISO 13565.

parameter_index: An index that specifies the name of the measuring_method further.

applied_surfaces: Physical surface of machining_feature that will define where the surface_texture_parameter will be applied.

4.5.20.1 Machined surface

The physical surface of machining_feature where the surface_texture_parameter will be applied.

```
ENTITY machined_surface;
  its_machining_feature: machining_feature;
  surface_element:       bottom_or_side;
END_ENTITY;
```

its_machining_feature: The physical shape of the workpiece where the machined_surface belongs.

surface_element: The selection of the location of the machined_surface in reference to its_machining_feature

4.5.20.1.1 Bottom or side

Selection of the location of a surface in reference to the machining_feature.

```
TYPE bottom_or_side =
ENUMERATION OF (bottom, side, bottom_and_side);
END_TYPE;
```

4.6 To make things happen: Executables

4.6.1 Executable

Executable is the base entity of all executable objects. They initiate actions on a machine and shall be arranged in a defined order. There are three types of executable objects: workplans, nc_function, and workingstep.

```
ENTITY executable
  ABSTRACT SUPERTYPE OF (ONEOF(workingstep, nc_function, program_structure));
  its_id:               identifier;
(*
Informal proposition:
its_id shall be unique within the part programme.
*)
END_ENTITY;
```

its_id:The executable's identifier. It shall be unique within the part programme.

Workingsteps (Section 4.6.2) describe manufacturing or handling operations which involve interpolating axes, i.e. whose execution normally spans a certain period of time and which can only be executed in conjunction with a workpiece. They have attributes which describe the state of the machine during their execution.

NC functions on the other hand describe switching operations or other non-interpolating machine functionality, i.e. typically singular events. The current NC functions are listed in section 4.6.3.

Program structures (Section 4.6.4) are used to build logical blocks for structured programming of the manufacturing operation The program structures, not the list of manufacturing features, have the authority over the actual manufacturing sequence. The most important structure is the workplan which provides linear sequence of executables. To arrange the manufacturing sequence in an arbitrary manner, structures for parallel processing, loops and conditional execution can be used.

4.6.2 Workingstep

The workingsteps represent the essential building blocks of an ISO 14649 NC programme. They can either be technology-independent actions, like rapid movements or probing operations, or machining workingsteps which relate to the different technologies like milling, drilling, turning etc.

The actual content of the workingsteps is specified in the entity operation and its subtypes. The reason for this design is the possibility to re-use the information specified for an operation for several features of the workpiece. An operation can be associated with multiple features and can be used in different locations. A workingstep, on the other hand, is unique. Duplicating a workingstep in a workplan will replicate the exact same physical machine action. For more information on operations see Section 4.7.1.

Workingsteps describe manufacturing or handling operations which involve interpolating axes. They do not describe a sequence of events in time but rather the conditions under which the operation has to be performed. For example, the workingstep calls for a specific tool which is required for an operation, and may specify the use of coolant. It does not, however, specify at exactly which moment the tool needs to be changed and whether the tool change or the activation of the coolant comes first. These decisions are left to the programmer or in real time to the NC controller. If in this example the requested tool is already in the spindle during the previous workingstep, the NC controller will decide not to perform any tool change.

At the lowest level of information provided by ISO 14649, the operations can also contain an explicit and exact description of the toolpath if this is required by the CAM system or the NC controller. For the explicit definition of tool movements, see Section 4.8.1.

```
ENTITY workingstep
ABSTRACT SUPERTYPE OF (ONEOF (machining_workingstep, rapid_movement,
  touch_probing))
SUBTYPE OF (executable);
  its_secplane :         elementary_surface;
END_ENTITY;
```

its_secplane: The security plane for the workingstep. On or above this plane, i. e. for z values greater than those of the elementary_surface, a safe movement of the tool without danger of collision is possible. The dimensions given are relative coordinates as measured from the workpiece co-ordinate system or, if this workingstep is associated with a manufacturing_feature, in the local co-ordinate system of the feature.

4.6.2.1 Machining workingsteps

Machining workingsteps represent the machining process for a specified area of the workpiece. As opposed to the other workingsteps, machining workingsteps cannot exist independent of a feature. Rather, they specify the association between a distinct feature and an operation to be performed on the feature. This removes the ambiguity of the n:1 relation between features and operations, thus creating an unambiguous specification which can be executed by the machine.

As the underlying operation, the machining workingstep is characterised by the use of a single tool and a set of technological parameters which normally remain constant during the reign of this machining workingstep. Upon the execution of a machining workingstep the first operation will be, if necessary, a tool change. During the machining workingstep, no tool change is possible.

The machine is instructed to reach the operating conditions specified by its_operation (attributes its_tool and its_technology) before the operation of the workingstep commences. If the machine is unable to reach these conditions during the preceding workingstep (e.g. during a preceding rapid movement) a halt will occur before the execution of the workingstep until all parameters are stable.

See the technology specific parts of ISO 14649 for details on operations, especially on how the association between the feature and the operation will change the interpretation of the operation.

```
ENTITY machining_workingstep
  SUBTYPE OF (workingstep);
  its_feature:           manufacturing_feature;
  its_operation:         machining_operation;
  its_effect:            OPTIONAL in_process_geometry;

END_ENTITY;
```

its_feature: The manufacturing_feature upon which the workingstep operates. its_operation:The operatior

its_effect: The change to the geometry of the workpiece effected by the operation. A CAM system can use this attribute to describe the predicted effect of this operation on the geometry of the workpiece. If given the controller can compare the geometry change described by this attribute with the geometry change predicted by its internal algorithm. ISO 14649 does not describe how closely the two geometries must match in order for the controller to be considered to be in conformance.

Note: Based on the interpretation of the underlying operation a machining workingstep typically has a well-defined start and end point of the tool motion. Thus it will not generally be possible that two subsequent elements in a workplan are machining workingsteps. A gap between the end point of the predecessor and the start point of the successor shall cause the machine to stop.

There are three exceptions to this rule:

(a) the respective end and start points of two subsequent workingsteps happen to coincide,

(b) a tool change is required between the two workingsteps in which case the controller has to calculate the respective toolpaths to and from the tool change position,

(c) the start point of the workingstep is not well-defined and left to the NC controllers discretion, e.g. a pocket finishing operation following a roughing operation where the finishing may start at either boundary of the pocket. However, if an operation contains an explicit toolpath, start and end point are always well-defined.

So often a rapid movement will be used between two machining workingsteps in order to bridge the gap, see Section 4.6.2.1.1.

4.6.2.1.1 In_process_geometry

Entity to represent in-process geometry for additional checking routines. A CAM system can use this information to describe the predicted effect of an operation on one feature. The controller can compare the geometry given in this entity with the geometry change predicted by its internal algorithm. ISO 14649 does not describe how closely the two geometries must match in order for the controller to be considered to be in conformance.

```
ENTITY in_process_geometry;
  as_is:                 OPTIONAL advanced_brep_shape_representation;
  to_be:                 OPTIONAL advanced_brep_shape_representation;
  removal:               OPTIONAL advanced_brep_shape_representation;
WHERE
  WR1: EXISTS (as_is) OR EXISTS (to_be) OR EXISTS (removal);
END_ENTITY;
```

as_is: Attribute to describe the geometry before an operation is executed.

to_be: Attribute to describe the desired effect of an operation.

removal: Volume removed by an operation.

4.6.2.2 Rapid movement

For rapid movements between two workingsteps the entity rapid_movement is used. If no toolpath is given, i. e. the inherited attribute its_toolpath is not set, the NC controller will move the tool in rapid motion from the current position to the beginning of the next workingstep which needs to have a defined starting point. The connection will be done via the security plane.

If a toolpath is specified, this toolpath will be executed in rapid motion. Note that a toolpath of type parameterised path can be used so as to avoid an explicit toolpath definition in cartesian space. This will provide greater flexibility in case of changing start and/or end points of the neighbouring workingsteps.

Unlike a machining workingstep, the rapid movement by itself has no well-defined start and end point unless an explicit toolpath is specified in cartesian space.

If rapid_movement is used during a five-axis machining operation, it will interpolate the tool direction between the previous and the next workingstep. In this case, the tool will retract in the tool direction from the last point of the previous workingstep. On the security plane the tool will rotate to the new tool direction, and then it will move down in the tool direction for the first point of the next workingstep. No change of the tool direction takes place during the lift and approach to and from the security plane.

However, if a toolpath is specified and the toolpath contains an explicit definition of the toolaxis then this definition will prevail.

```
ENTITY rapid_movement
  SUPERTYPE OF (return_home)
  SUBTYPE OF (workingstep, operation);
END_ENTITY;
```

Note that the first element in a workplan needs to be a rapid movement without explicit toolpath in order to move the tool from its unknown start position to the start point of the first machining operation.

4.6.2.2.1 Return home

This workingstep positions all machine axes to the machine-defined home position in the absolute machine coordinate system with a pre-defined sequence. This should normally be the last operation in a workplan.

```
ENTITY return_home
  SUBTYPE OF (rapid_movement);
END_ENTITY;
```

4.6.2.3 Touch probing

This is the supertype of touch probe workingsteps. Unlike other workingsteps, touch_probing returns a value for further consideration by the NC programme.

In future releases of this part of ISO 14649, touch_probing may be moved to a separate part specifying operations for measurement equipment. In this case it will become a subtype of machining_operation and needs to be associated with an appropriate measurement feature.

```
ENTITY touch_probing
  ABSTRACT SUPERTYPE OF (ONEOF (workpiece_probing, workpiece_complete_probing,
    tool_probing))
  SUBTYPE OF (workingstep, operation);
  measured_offset: nc_variable;
END_ENTITY;
```

measured_offset: The measured value of probing.

4.6.2.3.1 Workpiece probing

Probing of a dimension with one axis movement. Probing tool movement starts at start_position in the direction "direction" towards the workpiece. When the probe tool touches the workpiece, the machine stops and the difference of expected_value and tool position is stored in the inherited attribute measured_offset.

Like machining_workingstep, workpiece_probing has a well defined start point and cannot be the first element in a workplan.

```
ENTITY workpiece_probing
  SUBTYPE OF (touch_probing);
  start_position:         axis2_placement_3d;
  its_workpiece:          workpiece;
  its_direction:          direction;
  expected_value:         toleranced_length_measure;
  its_probe:              touch_probe;
END_ENTITY;
```

start_position: The starting position for the probing operation.

its_workpiece: The workpiece to be probed.

its_direction: The direction of the probing movement.

expected_value: The approximate value for probing.

its_probe: The identification of the probe which has to be used.

4.6.2.3.1.1 Touch probe

This gives an identification of touch probe which has to be used for workpiece probing.

```
ENTITY touch_probe;
  its_id: identifier;
END_ENTITY;
```

its_id: The identification of touch probe used.

4.6.2.3.2 Workpiece_complete_probing

This entity is similar to workpiece probing only that a complete measurement cycle at six locations of the given workpiece is automatically performed and the translation and rotational offset of the detected workpiece position compared to the given workpiece is computed. The locations of probing are automatically determined by the NC controller based on the geometry of the workpiece. The offset vector is returned in computed_offset while the inherited attributed measured_offset will hold the average of all (six) measured offsets.

Based on the automatic determination of the probing locations, workpiece_complete_probing does *not* have a well defined start point.

```
ENTITY workpiece_complete_probing
  SUBTYPE OF (touch_probing);
  its_workpiece:          workpiece;
  probing_distance:       toleranced_length_measure;
  its_probe:              touch_probe;
  computed_offset:        offset_vector;
END_ENTITY;
```

its_workpiece: The workpiece to be probed.

probing_distance: The distance between the probe's tip and the workpiece normal to its surface at the beginning of each probing movement. This value should be well above the maximum expected deviation between the actual position and the workpiece position specified by its_workpiece.

its_probe: The identification of the probe which has to be used.

computed_offset:The attributes 'translation' and 'rotate' describe the actual workpiece position compared to the position specified by the respective setup. It is computed as a result of the (six) measurements performed by this workingstep.

4.6.2.3.2.1 Offset vector

This entity is used to store the offset resulting from a workpiece probing operation. For entity nc_variable, see section 4.6.4.9. All variables must have initial values, most likely zeros.

```
ENTITY offset_vector;
  translate:  LIST [3:3] OF nc_variable;
  rotate:     OPTIONAL LIST [3:3] OF nc_variable;
WHERE
  WR1: (SIZEOF(QUERY(i <* translate | NOT EXISTS(i.initial_value))) = 0)
    AND (NOT EXISTS(rotate) OR (SIZEOF(QUERY(i <* rotate |
    NOT EXISTS(i.initial_value))) = 0));
END_ENTITY;
```

translate: Variables to store the translation required in each co-ordinate direction (x-, y-, z-axis) in order to move from the theoretical position to the actual position, measured in millimetres.

rotate: Variables to store the rotation around the x-, y-, and z-axis, respectively, in order to move from the theoretical position to the actual position, measured in degrees. If omitted, only the translational offset will be considered.

4.6.2.3.3 Tool probing

Probing of the length and width/diameter of a tool. The selected tool starts its movement at a machine dependent start position. From that position the tool position is shifted to a fixed sensor position by the offset. Then the tool is moved in direction to the sensor until contact. This is done in longitudinal and perpendicular direction of the tool-axis.

The result of probing is stored in the inherited attribute measured_offset and the NC controllers tool database is automatically updated. NC controllers providing tool correction algorithms are expected to use the updated information of its_tool from the moment the tool_probing workingstep has been completed.

```
ENTITY tool_probing
  ABSTRACT SUPERTYPE OF (ONEOF (tool_length_probing, tool_radius_probing))
  SUBTYPE OF (touch_probing);
  offset:                 cartesian_point;
  max_wear:               positive_length_measure;
  its_tool:               machining_tool;
END_ENTITY;
```

offset: location of the sensor.

max_wear: maximum permissible wear in length direction.

its_tool: The tool which has to be probed.

4.6.2.3.4 Machining tool

This is the supertype for all tools which are needed for machining. They are to be specified in the technology-specific parts of ISO 14649 or future standards might be referenced.

```
ENTITY machining_tool
  ABSTRACT SUPERTYPE OF (ONEOF(cutting_tool));
  its_id:                     label;
END_ENTITY;
```

its_id : An unique label to exactly identify the tool.

4.6.2.3.5 Cutting tool

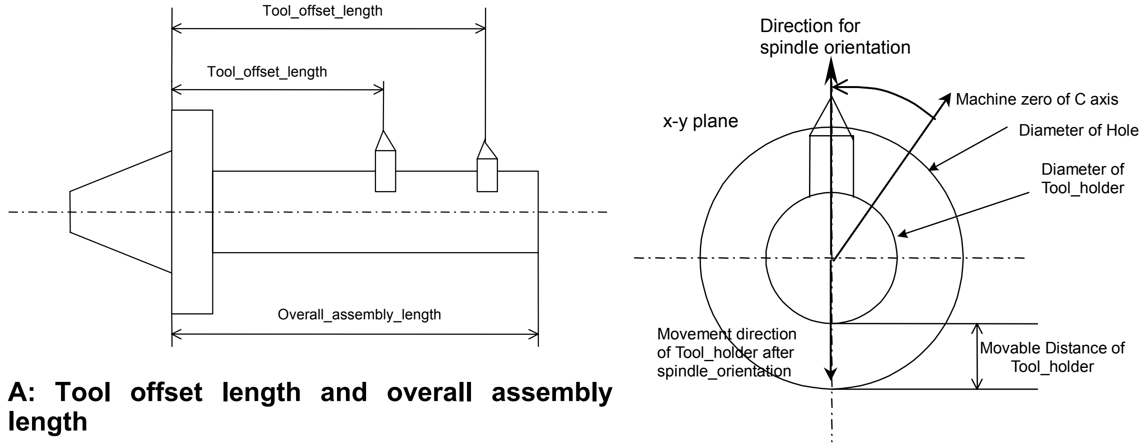
Entity to describe the basic information needed for cutting tool description, which includes all type tools for chipping technologies (e.g. lathe tool, milling cutter, reamer, drill, tap, boring tool). As Part 10 cannot be implemented without at least one specific technology dependent part this Entity only includes general data and therefore is an abstract supertype.

```
ENTITY cutting_tool
 ABSTRACT SUPERTYPE
 SUBTYPE OF (machining_tool);
 its_tool_body:           tool_body;
 its_cutting_edge:        LIST [1:?] OF cutting_component;
 overall_assembly_length: OPTIONAL length_measure;
END_ENTITY;
```

its_tool_body: The information describing the kind of cutting tool and it’s corresponding characteristics.

its_cutting_edge: The information describing the cutting edge(s) of the cutting tool.

overall_assembly_length: The entire length of the assembled tooling measured along the tooling axis for tool collision check. (Figure 39)



A: Tool offset length and overall assembly length

B: Direction for spindle orientation and tool holder diameter for spindle orientation

Figure 39.Cutting tool

4.6.2.3.6 Tool body

This is the abstract supertype for all types of tool bodies. These tools may be of any kind used for chipping technologies. The subtypes will be defined in the technology specific parts of ISO 14649.

```
ENTITY tool_body
  ABSTRACT SUPERTYPE;
END_ENTITY;
```

4.6.2.3.7 Cutting component

Entity to describe the name of the material composing the cutting edge of a solid tool or the insert and it's technological data.

```
ENTITY cutting_component;
 tool_offset_length:    length_measure;
 its_material:          OPTIONAL material;
 technological_data:    OPTIONAL cutting_edge_technological_data;
 expected_tool_life:    OPTIONAL time_measure;
 its_technology:        OPTIONAL technology;
END_ENTITY;
```

tool_offset_length: The measured distance from the cutting tip to a designated standard reference point on the tool holding assembly for tool presetting. (see figure 1)

its_material: The identification of the material composing the cutting edge of a solid tool or the insert. (Examples: The attribute material_identifier of the entity material can be High speed steel(HSS), Carbide, or Polycrystalline diamond(PCD) [ISO/CD 13399-3, Table 8]

technological_data: Technological information about the cutting edge.

expected_tool_life: The expected number of hours that a given cutting tool can be used before tool wear significantly impacts the performance parameters. This value will also be dependent on a number of external factors, including workpiece material, desired part tolerances, and selected cutting parameters.

its_technology: The technology defines the technological parameters to be used for machining (e.g. the spindle speed and the feed of the tool which guarantee expected_tool_life).

4.6.2.3.8 Cutting edge technological data

Entity to describe the technological data associated with the cutting edge.

```
ENTITY cutting_edge_technological_data;
 cutting_angle:         OPTIONAL plane_angle_measure;
 free_angle:            OPTIONAL plane_angle_measure;
 aux_angle:             OPTIONAL plane_angle_measure;
END_ENTITY;
```

cutting_angle: The angle of the edges of the cutting tool.

free_angle: The free angle of the cutting edges.

aux_angle: The auxiliary angle of the cutting edges.

4.6.2.3.9 Tool length probing

The entity can be used for probing the tool length.

```
ENTITY tool_length_probing
  SUBTYPE OF (tool_probing);
END_ENTITY;
```

4.6.2.3.10 Tool radius probing

The entity can be used for probing the tool radius.

```
ENTITY tool_radius_probing
  SUBTYPE OF (tool_probing);
END_ENTITY;
```

4.6.3 NC function

A NC function is an executable object. It describes manufacturing or handling operations which do not involve interpolation of axes. It shall be switching operations or other singular-event machine functionality.

```
ENTITY nc_function
  ABSTRACT SUPERTYPE
  SUBTYPE OF (executable);
END_ENTITY;
```

4.6.3.1 Display message

This function is used to display a message on the operator's screen.

```
ENTITY display_message
  SUBTYPE OF (nc_function);
  its_text:             text;
END_ENTITY;
```

its_text: The message to be displayed on the operator's screen, until the next message is sent. To clear the screen, send an empty message.

4.6.3.2 Optional stop

This function is used to stop executing the NC program until the operator presses the start button. The optional stop permits the user to guide the tool. The controller has to consider the changed axis positions etc. after the optional stop is released. However, it is only effective under the condition that the operating mode “optional stop” is enabled on the NC controllers operating panel.

```
ENTITY optional_stop
  SUBTYPE OF (nc_function);
END_ENTITY;
```

4.6.3.3 Program stop

This function is used to stop executing the NC program until the operator presses the start button.

```
ENTITY program_stop
  SUBTYPE OF (nc_function);
END_ENTITY;
```

4.6.3.4 Set mark

This function is used to synchronize the multiple channel operation. When it is invoked in the channel specified by the attribute its_channel of the entity wait_for_mark, the operation in that channel is started.,

```
ENTITY set_mark
  SUBTYPE OF (nc_function);
END_ENTITY;
```

4.6.3.5 Wait for mark

This function is used to synchronize the multiple channel operation. When it is invoked, the specified channel start to wait until the mark is read from another channel's program or from the main program.

```
ENTITY wait_for_mark
  SUBTYPE OF (nc_function);
  its_channel:          channel;
END_ENTITY;
```

its_channel: The identification of the channel engaged in a synchronized operation.

4.6.4 Program structure

A program structure is an executable object which includes other executables. The included executables can be executed depending on conditions or in manners determined by the program structure object.

```
ENTITY program_structure
```

```
    ABSTRACT SUPERTYPE OF (ONEOF(workplan, parallel, non_sequential, selective,
      if_statement, while_statement, assignment))
    SUBTYPE OF (executable);
  END_ENTITY;
```

4.6.4.1 Workplan

The entity workplan allows to combine several workingsteps and NC functions in a linear order. It also serves as an attribute of the top level entity project which has to be provided in each ISO 14649 data model exactly once. The recursive definition of workplan as subtype of executable allows to group manufacturing operations into larger units, e.g. to summarise all turning operations and all grinding operations if a workpiece is to undergo several different processes.

```
  ENTITY workplan
    SUBTYPE OF (program_structure);
    its_elements:          LIST[1:?] OF executable;
    its_channel:           OPTIONAL channel;
    its_setup:             OPTIONAL setup;
    its_effect:            OPTIONAL in_process_geometry;
  WHERE
    WR1: SIZEOF(QUERY(it <* its_elements | it = SELF)) = 0;
  END_ENTITY;
```

its_elements: An ordered sequence of executable objects. The workplan cannot contain itself as element. It can only contain itself indirectly within a control structure which allows to skip this workplan in order to prevent an infinite loop.

its_channel: The identifier of the channel used for the execution of workplan. This is only for machine control systems which support multiple channel operation.

its_setup: The setup includes the workplan's global security plane and a zero offset where all feature placements are referred to.

its_effect: The change to the geometry of the workpiece effected by the operation. A CAM system can use this attribute to describe the predicted effect of this operation on the geometry of the workpiece. If given the controller can compare the geometry change described by this attribute with the geometry change predicted by its internal algorithm. ISO 14649 does not describe how closely the two geometries must match in order for the controller to be considered to be in conformance.

4.6.4.1.1 Channel

This entity is used to identify the channel when multiple channel operation is engaged. It is currently a primitive implementation and cannot be used for machine independent programs. Further attributes, like special kinematical descriptions, can be added to this entity in the future. For simple synchronisation, the NC functions set_mark and wait_for_mark can be used.

```
  ENTITY channel;
    its_id:                identifier;
  END_ENTITY;
```

its_id: The identification of the channel.

4.6.4.1.2 Setup

This entity includes information concerning the location of the workpieces' coordinate systems. As one setup might include several workpieces of which each one might have its own coordinate system there is one more hierarchical level of frame of reference:

By the setup's attribute its_origin a cartesian coordinate system is defined relative to the machine coordinate system. It is the frame of reference for the location of each workpiece related to the setup.

The setup's origin is only valid within this particular setup and is not referenced by any other setup.

```
ENTITY setup;
  its_id:                identifier;
  its_origin:            OPTIONAL axis2_placement_3d;
  its_secplane:          elementary_surface;
  its_workpiece_setup: LIST [0:?] OF workpiece_setup;
(*
Informal proposition:
If its_origin is not set, the default for the origin
of the setup is identical with the machine origin.
*)
END_ENTITY;
```

its_id: The identification of the setup.

its_origin: Position and orientation of the setup's cartesian coordinate system relative to the machine coordinate system.

its_secplane: The security plane for the whole setup. On or above this plane, i. e. for z values greater than those of the elementary_surface, a safe movement of the tool without danger of collision is possible. The dimensions given are relative coordinates as measured from the origin of the setup.

its_workpiece_setup: Each workpiece which is included within the setup and which will be machined within the respective workplan is listed with its placement and additional informations for its setup.

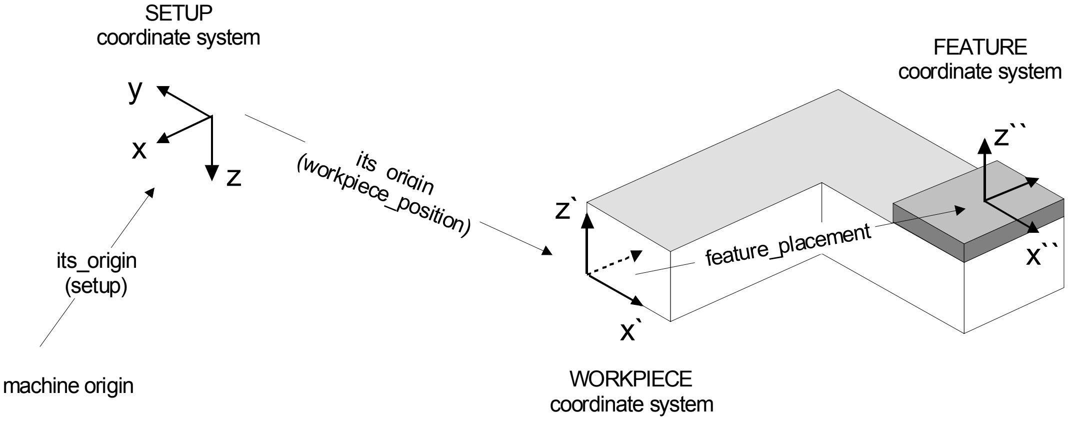


Figure 40.Coordinate systems

4.6.4.1.3 Workpiece_setup

This entity allows to specify for a workpiece its placement relative to the setup's origin.

```
ENTITY workpiece_setup;
  its_workpiece:        workpiece;
  its_origin:           axis2_placement_3d;
  its_offset:           OPTIONAL offset_vector;
```

```
    its_restricted_area: OPTIONAL restricted_area_select;
    its_instructions:    LIST [0:?] OF setup_instruction;
  END_ENTITY;
```

its_workpiece: affected workpiece

its_origin: the workpiece's cartesian system relative to the setup's cartesian system.

its_offset: The translational and rotational offset of the actual workpiece compared to the position specified by its_origin. It is used to adjust the positions due to a measurement operation. The offset can be changed by a measurement operation (e.g.4.6.2.3).

its_restricted_area: Area or volume within which tool-movements are forbidden due to possible collisions. Generally used to avoid collisions with fixtures or machine parts.

its_instructions: Contains optional setup instructions such as description or external documents reference etc.

4.6.4.1.4 Restricted_area_select

This select type offers different optional geometric elements to describe areas or volumes in which due to fixtures, machine elements, etc. there might occur collisions. (Tool-) Movements within this area are forbidden.

If the restricted area is of type bounded_surface movements below the surface risc collisions. By the bounding geometry a volume can be described. Within this volume might occur collisions.

```
    TYPE restricted_area_select = SELECT (
                         bounded_surface,
                         bounding_geometry_select
                           );
    END_TYPE;
```

Note: The directions mentioned are considered to be positive in positive z-direction of the its_origin. “Below” therefore means a decreasing z-value within the local coordinate system defined by its_origin of the workpiece position.

4.6.4.2 Setup_Instruction

The Setup_instruction includes operator instructions for the setup of its workpiece.

```
  ENTITY setup_instruction;
    description:         OPTIONAL text;
    external_document:   OPTIONAL identifier;
    WHERE
    WR1: EXISTS (description) OR EXISTS (external_document);
  END_ENTITY;
```

internal: Description of the setup instruction

external: Identifier used to identify external document reference such as tables, guidelines etc.

4.6.4.3 Parallel

The parallel entity allows to run several executables in parallel. All branches are started at the same time; the execution of the parallel entity ends when the last branch has finished. If the NC controller determines that the branches cannot be executed in parallel the machine will stop with an error condition. Note that normally parallel execution will require the use of different channels in each branch. NC functions can be used to synchronise events across different channels.

```
  ENTITY parallel
```

```
    SUBTYPE OF (program_structure);
    branches:              SET [2:?] OF executable;
  END_ENTITY;
```

branch: Executables to be executed in parallel.

Note: The parallel entity allows to define the general processing of execuables in parallel. To define the delay of single executalbes etc. we concede that there have to be defined more detailed subtypes of parallel.

4.6.4.4 Non_sequential

The non_sequential entity allows to define a set of executables which all shall be executed but which's order is not prescribed. In contrast to the workplan entity the non_sequential does not define a sequence.

```
  ENTITY non_sequential
  SUBTYPE OF (program_structure);
    its_elements:          SET[2:?] OF executable;
  END_ENTITY;
```

its_elements: An set of executable objects. The non_sequential cannot contain itself as element. It can only contain itself indirectly within a control structure which allows to skip itself in order to prevent an infinite loop.

Note: This is the minimum requirement for controller determined sequencing. Additional attributes providing the controller extra information for sequencing will be defined by future work.

4.6.4.5 Selective

The selective entity contains a set of executables from which only "one" will be executed. The other executables are to be omitted.

```
  ENTITY selective
  SUBTYPE OF (program_structure);
    its_elements:          SET[2:?] OF executable;
  END_ENTITY;
```

its_elements: A set of executable objects of which exactly one has to be selected and executed by the controller of the machine tool.

Note: This is the minimum requirement for controller determined selection. Additional attributes providing the controller extra information for selection will be defined by future work.

4.6.4.6 If statement

The if statement runs an executable if a given condition is fulfilled; it may include an alternative option.

```
  ENTITY if_statement
  SUBTYPE OF (program_structure);
    condition:             boolean_expression;
    true_branch:           executable;
    false_branch:          OPTIONAL executable;
  END_ENTITY;
```

condition: Expression to be tested based on the rules given below.

true_branch: Executable to be executed if the condition evaluates to be true.

false_branch: Executable to be executed if the condition evaluates to be false.

4.6.4.7 While statement

The while statement runs and repeats an executable as long a given condition is fulfilled. Note that all executables are defined statically so the operation in the body of the while statement cannot be modified during the execution of the loop.

```
ENTITY while_statement
  SUBTYPE OF (program_structure);
  condition:             boolean_expression;
  body:                  executable;
END_ENTITY;
```

condition: Expression to be tested based on the rules given below. It is tested before attempting the first execution of the body and then each time after the execution of body has been completed.

body: Executable to be executed as long as the condition is true. If the condition is not true upon the initial encounter of the while statement, no action occurs at all.

4.6.4.8 Assignment

Assigns a value to a nc_variable.

```
ENTITY assignment
  SUBTYPE OF (program_structure);
  its_lvalue:            nc_variable;
  its_rvalue:            rvalue;
END_ENTITY;
```

its_lvalue: The nc_variable which will be assigned a value.

its_rvalue: The value to be assigned to its_lvalue. This is either a numeric constant or a reference to another nc_variable.

4.6.4.9 NC variable

The variable concept introduced here is primitive. Only numeric variables are allowed. The NC controller will map the variable name given by this entity to an internal storage. The attribute its_name must therefore be unique throughout the entire model supplied by an ISO 14649 physical file. Not that several nc_variable entities using the same name will actually refer to the same storage location.

Accessing nc_variables in a program_structure entity may take place only after the previous program_structure has been completely executed by the machine. Note that due to the pipeline concept of most NC controllers there can be a significant time gap between the interpretation of the program and the actual execution. It is there the responsibility of the NC controller to halt interpretation until the value of the nc_variable(s) used are available. This is especially true for the lvalue in an assignment operation because the premature execution of the assignment would allow an ongoing NC process to overwrite the assigned nc_variable before it is used, leading to unexpected and possibly fatal results.

Note also that there is no guarantee at which time a certain value from any parallel branch would be available. If timing matters, a synchronising NC function has to be inserted before the use of a nc_variable.

Variables can be used to access controller-internal information, like sensor readings. This, however, would be controller-dependent and implemented by the use of reserved names as specified by the control vendor.

```
ENTITY nc_variable;
  its_name:               LABEL;
  initial_value:          OPTIONAL NUMBER;
END_ENTITY;
```

its_name: The unique reference to an internal storage provided by the NC controller. All characters in LABEL are significant. The number of different nc_variable names allowed is controller dependent.

initial_value: The value of the nc_variable before it is accessed the first time. If omitted, the initial value is undefined.

4.6.4.10 NC_constant

The nc_constant is used to assign a constant value to an attribute. Its value is determined by the NC-Program and cannot be modified within the process.

```
ENTITY nc_constant;
  its_name:                  LABEL;
  its_value:                 OPTIONAL NUMBER;
END_ENTITY;
```

its_name: Name of the constant. It shall be unique within the program.

its_value: The value of the nc_constant.

4.6.4.11 Rvalue

In assignment and comparison operations, the rvalue can be used to specify either a numeric constant or a reference to a nc_variable.

```
TYPE rvalue = SELECT(nc_constant, nc_variable);
END_TYPE;
```

4.6.4.12 Boolean expression

The Boolean expression entity is used to determine whether a given condition is met. At this time, it can either be a numeric comparison or a composition of several other Boolean expressions.

```
ENTITY boolean_expression
  ABSTRACT SUPERTYPE OF(ONEOF(unary_boolean_expression, binary_boolean_expression,
    multiple_arity_boolean_expression, comparison_expression));
END_ENTITY;
```

4.6.4.12.1 Unary_boolean_expression

Unary operators perform Boolean algebra on one arguments and produce a false or true value.

```
ENTITY unary_boolean_expression
  ABSTRACT SUPERTYPE OF(not_expression)
  SUBTYPE OF (boolean_expression);
   operand: boolean_expression;
END_ENTITY;
```

4.6.4.12.2 Not expression

Logical negation 'NOT': NOT 0 = 1 and NOT 1 = 0.

```
ENTITY not_expression
  SUBTYPE OF (unary_boolean_expression);
END_ENTITY;
```

4.6.4.12.3 Binary_boolean_expression

Binary operators perform Boolean algebra on two arguments and produce a false or true value.

```
ENTITY binary_boolean_expression
  ABSTRACT SUPERTYPE OF(xor_expression)
  SUBTYPE OF (boolean_expression);
  operand1: boolean_expression;
  operand2: boolean_expression;
END_ENTITY;
```

4.6.4.12.4 Xor expression

Logical "not equal to" function: 0 XOR 0 = 0, 0 XOR 1 = 1, 1 XOR 0 = 1, and 1 XOR 1 = 0.

```
ENTITY xor_expression
  SUBTYPE OF (binary_boolean_expression);
END_ENTITY;
```

4.6.4.12.5 Multiple_arity_boolean_expression

Binary operators perform Boolean algebra on multiple arguments and produce a false or true value: AND(a,b,c, ...) is true if a,b,c,etc are all true and OR(a,b,c, ...) true if one of a,b,c,etc is true.

```
ENTITY multiple_arity_boolean_expression
ABSTRACT SUPERTYPE OF(ONEOF(and_expression, or_expression))
SUBTYPE OF (boolean_expression);
  operands: LIST [2:?] OF boolean_expression;
END_ENTITY;
```

4.6.4.12.6 And expression

Logical conjunction: 0 AND 0 = 0, 0 AND 1 = 0, 1 AND 0 = 0, 1 AND 1 = 1.

```
ENTITY and_expression
SUBTYPE OF (multiple_arity_boolean_expression);
END_ENTITY;
```

4.6.4.12.7 Or expression

Logical disjunction: 0 OR 0 = 0, 0 OR 1 = 1, 1 OR 0 = 1, 1 OR 1 = 1.

```
ENTITY or_expression
SUBTYPE OF (multiple_arity_boolean_expression);END_ENTITY;
```

4.6.4.12.8 Comparison_expression

The comparison establishes a relationship between the values of two operands.. Carefully observe the note on the time-critical behaviour of variables.

```
ENTITY comparison_expression
  ABSTRACT SUPERTYPE OF(ONEOF(comparison_equal, comparison_not_equal,
    comparison_greater, comparison_greater_equal, comparison_less,
    comparison_less_equal))
  SUBTYPE OF (boolean_expression);
```

```
    operand1:               nc_variable;
    operand2:               rvalue;
  END_ENTITY;
```

operand1: Designation of a nc_variable which is the first of two values to compare.

operand2: Designation of an rvalue which may either be a nc_variable of a numeric constant.

The result of a comparison is a Boolean depending whether the relationship is true or false. The comparison operators are: ‘equal’ (==), ‘not_equal’ (!=), ‘greater_equal’ (>=), ‘greater’ (>), ‘less_equal’ (<=), ‘less’ (<):

```
  ENTITY comparison_equal
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_not_equal
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_greater
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_greater_equal
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_less
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_less_equal
    SUBTYPE OF (comparison_expression);
  END_ENTITY;
```

4.7 How to machine: Operations

Operations are used to specify the content of a workingstep. Unlike a workingstep, an operation cannot be executed by itself – unless it is also a workingstep by means of multiple inheritance, like rapid_movement or touch_probing. The latter do not require the presence of a feature for unambiguous interpretation.

However, the most typical operation, machining operation, needs an association with a manufacturing feature in order to be interpreted. It relies on the geometric information provided by the feature, namely its placement in the workpiece co-ordinate system. Such association can either be established through a reference from a manufacturing_feature or through a reference from a machining_workingstep which references in turn a manufacturing_feature.

The important difference between a workingstep and an operation is in the number of associated features. The workingstep is associated with zero or one features. So at the time of execution the NC controller will know unambiguously which feature to use for geometric placement (or to use workpiece co-ordinates if no feature is associated). A workingstep is thus unique. Duplicating a workingstep in a workplan will replicate the exact same physical machine action.

The operation on the other hand can be associated with zero or multiple features. The same operation can thus be used in different locations. The NC controller would be unable to decide on which of the associated features to execute the operation, so an execution of a “naked” operation is not possible.

The reason for this design is the possible re-use of operation data for several manufacturing features. Consider ten identical holes in different positions around the workpiece. They can all share the operation data, but they can not share the workingsteps which represent an operation at a certain instance in time and space. As the “heavyweight” operation entities can be shared, this may reduce the volume of data to be exchanged.

4.7.1 Operation

The supertype operation specifies the generic data required by all operations.

All operations have the option of specifying an explicit toolpath. For 2½D machining operations, many numerical controls will specify cycles for generating their own toolpaths. However, for older controls which are not able to generate toolpaths or if technological reasons require the communication of exactly prescribed toolpaths, the attribute its_toolpath may be used. Also, the 3- to 5-axis milling of freeform surfaces will typically require the explicit specification of toolpaths by a CAM system.

```
ENTITY operation
  ABSTRACT SUPERTYPE OF (ONEOF (machining_operation, rapid_movement,
    touch_probing));
  its_toolpath:          OPTIONAL toolpath_list;
  its_tool_direction:    OPTIONAL tool_direction;
END_ENTITY;
```

its_toolpath: Its_toolpath defines a list of all toolpaths in this operation. Note: If its_toolpath exists, the attribute its_machining_strategy in the subtype machining_operation is for information only. Also, all semantics or parameters provided by the operation will be overridden by its_toolpath. The toolpaths are given in coordinates as measured from the workpiece co-ordinate system or, if this operation is associated with a manufacturing_feature, in the local co-ordinate system of the feature.

its_tool_direction: Specification of the type of tool orientation used. If none is given, a technology specific default is assumed. Therefore refer to the technology specific parts of ISO 14649.

Note that the operation is geometrically linked with the associated manufacturing_feature, if such an association exists. If the feature has a feature_placement attribute, this transformation will also apply to all geometry specified here, i. e. its_toolpath and to all other geometric information like directions or orientations which may be used in subtypes of operation. In other words, all geometry and parameters of the machining_operation are defined in the local co-ordinate system of the feature. As the operation may be associated with multiple manufacturing_features it can only be interpreted in the context of such feature.

For features without a feature_placement attribute or for operations without an associated feature, all data is to be interpreted in workpiece co-ordinates as specified by the attribute its_origin of the workplan which uses this operation.

4.7.1.1 Toolpath list

The entity toolpath_list contains a list of toolpaths (trajectories) which will be executed by the CNC one after the other.

```
ENTITY toolpath_list;
  its_list:              LIST [1:?] OF toolpath;
END_ENTITY;
```

its_list: A list of toolpaths. These toolpaths shall be executed in the defined order. See Section 4.8.1 for details about the toolpath definition.

4.7.1.2 Tool direction

The following entities define the tool direction for freeform machining.

```
ENTITY tool_direction
```

```
    ABSTRACT SUPERTYPE OF (ONEOF (two_axes, three_axes));
  END_ENTITY;
```

4.7.1.2.1 Two axes

Only two axes can be simultaneously active.

```
  ENTITY two_axes
    SUBTYPE OF (tool_direction);
  END_ENTITY;
```

4.7.1.2.2 Three axes

Simultaneous tool movements in three axes are used for machining. The tool orientation is always parallel to the third axis (generally, the z axis) in the machine co-ordinate system. Therefore the tool direction will not be affected by any workpiece placements or workingstep transformation.

```
  ENTITY three_axes
    SUBTYPE OF (tool_direction);
  END_ENTITY;
```

4.7.2 Machining operation

Machining operations define the machining process for a limited area of the workpiece, i. e. the contents of a machining workingstep. For the feature to which they refer they specify, at a minimum, the tool to be used, and a set of technological parameters. This data forms an integral part of the operation and cannot be normally changed during its scope.

However, for special operations, and if an explicit toolpath is provided in the inherited attribute its_toolpath, it is also possible to specify certain deviating parameters along portions of the toolpath. In this case, the data given in machining_operation serves as default data in case no specific technological information for a toolpath is provided. Note that any parameters or semantics specified by this operation or the associated feature will be overridden by the toolpath definition, even if the actions defined by the toolpath contradict the intuitive notion of this operation of the associated feature. In other words, you can even use a pocket operation to bore a hole.

The first tool movement within a machining_operation will typically be an approach movement to the defined start point, allowing the machine to reach its operating speed. This can be either be defined explicitly in a toolpath, or by means of a parameterised path, or it can be defined within the strategy for the operation where the exact definition of the path is left to the NC controller. The last tool movement will typically be a retract or lift movement.

The tool movement within the machining_operation, unless defined by an explicit toolpath, will be determined by the technology-dependent strategy and additional parameters as defined by subtypes of machining_operation.

Machining_operation is the supertype for all technologies included in ISO 14649. For each of these technologies, specific machining strategies are defined in separate, technology depending parts. They may be used to instruct the NC controller on how to generate toolpaths if no toolpath is explicitly specified for this operation.

```
  ENTITY machining_operation
    ABSTRACT SUPERTYPE
    SUBTYPE OF (operation);
    its_id:                 identifier;
    retract_plane:          OPTIONAL length_measure;
    start_point :        OPTIONAL cartesian_point;
    its_tool:               machining_tool;
    its_technology:         technology;
    its_machine_functions:  machine_functions;
  (*
Informal proposition:
If attribute SELF\operation.its_toolpath exists, then attributes
its_machining_strategy, retract_plane and start_point, if present, are for
information only.
*)
END_ENTITY;
```

its_id: A unique identifier of the operation.

retract_plane: The height of a retract plane associated with this operation. This is not the security plane. The start and the end point of the operation – as discussed in Section 4.6.2.1 – shall be within z-direction at the length given by the attribute retract_plane. It is guaranteed that the approach movement from the retract plane to the first cut and the lift movement from the last cut to the retract plane are executed in cutting feed as specified for the operation. A change to rapid feed, e.g. to reach the security plane, can only occur above this plane, and only in the context of a new operation which should be of type rapid_movement.

If not given, and if the inherited attribute its_toolpath is not given, the NC controller will determine an appropriate retract plane which may be the security plane. If its_toolpath is given, or if approach or retract strategies are given which do not make use of a retract plane, this attribute will be ignored.

Depending on the type of operation, the attribute will be interpreted as follows:

- For plane milling or drilling operations, retract_plane is the z coordinate of the retract plane from which the tool starts (for approach) and to which it is retracted (for retract movements).
- For side milling, retract_plane indicates a distance perpendicular to the manufacutred surface.
- For freeform milling, retract_plane is the distance between the workpiece surface and the retract plane which is oriented perpendicular to the surface normal in the first cutting point (for approach) or the last cutting point (for lift).

start_point: Optional starting point of the cutting process specified as tool centre point in the local xy plane. The z co-ordinate of start_point is determined depending on the type of operation:

- For milling, the z co-ordinate is the depth of the first cut, excluding any plunge or approach movements. If an approach or plunge movement is used, the actual start point of the operation will be the start point of that approach or plunge movement which can be calculated based on this attribute. The start_point will be reached at the end of such approach or plunge. Thus the x and y co-ordinates of start_point not necessarily coincide with those of the start point of the operation, depending on the type of approach or plunge.
- For drilling, the start_point is identical with the start point of the operation as no approach or plunge is foreseen here. Therefore, the z co-ordinate is given by the attribute retract_plane.

If this attribute is given, the operation has a defined start point as discussed in Section 4.6.2.1, otherwise not. In the latter case, the NC controller will determine a reasonable default.

If its_toolpath is given, or if start_point specifies a point violating the feature's boundary or otherwise incompatible with the machining strategy, this attribute will be ignored.

its_tool: The tool which has to be used for this workingstep. For the definition of machining_tool, please refer to technology-specific parts of ISO 14649. It is important to understand that the tool data given describes the *ideal required tool*. If this attribute calls for a tool with diameter 10.0 mm, the NC controller may select a tool with diameter 9.983 mm provided that it has the ability for on-line tool correction and has the actual value of the tool stored in its tool memory data, and provided that the result of the operation remains within the given tolerances of the feature. (In other words, the NC controller is not allowed to select a 9.983 mm drill if the hole has the toleranced dimension $10.0_{-0.01}$ mm.) The possibility to find a suitable tool in the machine's tool magazine is reduced the more details for the tool are specified. So only the technologically necessary parameters should be given. (In other words, do not specify a tool length if only the diameter is of importance.) If a tool id is specified, only an exact match can be selected.

its_technology: The technological parameters of the machining operation, like spindle speed and feed of the tool.

its_machine_functions: Indicates the state of various machine functions, like coolant, chip removal, etc. to be applied during the time span of this operation.

Please observe carefully the note about the geometric relation with the associated feature in Section 4.7.1. Keep in mind that in the local co-ordinate system the planar contour of the feature lies in the xy plane (z = 0), and material is assumed to be in negative z direction, as defined in 4.5.2.

4.7.2.1 Technology

This entity is the supertype for the technologies defined in the following parts of ISO 14649.

```
ENTITY technology
  ABSTRACT SUPERTYPE;
  feedrate:                      OPTIONAL speed_measure;
  feedrate_reference:            tool_reference_point;
  END_ENTITY;
```

feedrate : Feed of the tool expressed as a linear speed. The feed rate specified applies to the motion of the tool center point.

feedrate_reference: Specifies whether the feed rate given above is to be calculated with regard to the tool center point or the cutter contact point.

4.7.2.1.1 Tool reference point

This type can be used to select a tool reference point.

```
TYPE tool_reference_point = ENUMERATION OF (tcp, ccp);
END_TYPE;
```

tcp: Tool center point, i. e. the point where the rotational axis of the tool leaves the tip.

ccp: Cutter contact point, i. e. the point where the tool tip, modelled as an ideal cylinder, would touch the ideal surface of the finished workpiece.

4.7.2.2 Machine functions

Each technology has its specific machine functions. This entity is their abstract supertype.

```
ENTITY machine_functions
  ABSTRACT SUPERTYPE;
END_ENTITY;
```

4.8 To be in full control: Explicit toolpath definition

The toolpath entities generally describe the movement of the tool or of the axis of a machine. They represent the lowest level of information within ISO 14649 which actually allows to re-create all the information given in today's ISO 6983 G codes. However, even the low-level definition of the toolpaths in ISO 14649 still has an advantage over the legacy programmes. By connecting this information with the high-level operation and feature data, the toolpaths can always be interpreted within their semantic context. They are provided in a structure which allows to identify the individual toolpath rather than to search through thousands of lines of unstructured code for axis movements.

And, of course, the mathematical representation of the toolpaths in ISO 14649 fulfils modern CNC standards. The use of splines for toolpaths is supported as well as the definition of cutter-contact paths for controllers which allow on-line tool correction and kinematic transformation. This is required, among others, for today's new parallel kinematics.

Because the definition of the conformance classes has been done according to the interpolating requirements of a CNC, several subtypes of toolpaths have been defined. Two general classes of toolpaths are available, trajectories and parameterised paths. A trajectory is a precalculated movement and can reference as underlying element either a polyline or a spline. A parameterised path describes approach- ,lift - and connector-movements by the definition of type and parameters of the movement (e.g. tangential approach). The exact movement itself has to be calculated by the CNC.

4.8.1 Toolpath

This is the supertype for all explicit toolpaths. If needed, the toolpaths can have technological parameters and the machine functions which override the default value specified at the operation level.

If the attributes its_technology and/or its_machine_functions are given, the machine is instructed to reach the specified operating conditions before entering the toolpath. Unlike the similar case for workingsteps, no halt is allowed between toolpaths in an operation's toolpath list. So the previous toolpath may be adversely affected if a significant change of parameters occurs and the controller tries to provide run up to these parameters prior to the current toolpath. Also, if the time span of the previous toolpath is insufficient to reach stable parameters, there is no guarantee that the new toolpath will be using these parameters from the start.

Caution is therefore advised when using technological parameters at the toolpath level. If required, e.g. for special operations requiring a reversing of the spindle rotation, a feedstop toolpath should be inserted in order to grant the machine sufficient time for changing the parameters.

```
ENTITY toolpath
  ABSTRACT SUPERTYPE OF (ONEOF(feedstop, trajectory,  parameterised_path));
  its_priority :         BOOLEAN;
  its_type :             toolpath_type;
  its_speed :            OPTIONAL toolpath_speedprofile;
  its_technology:        OPTIONAL technology;
  its_machine_functions: OPTIONAL machine_functions;
END_ENTITY;
```

its_priority: As a toolpath may contain all necessary movements to manufacture a feature the priority of execution between the toolpath and the feature has to be determined. In case there is a toolpath of higher priority (true) than the geometry of the path generation based on the manufacturing feature is subordinated to this toolpath.

its_type: Type of the toolpath.

its_speed: A means to influence the feed rate. This factor will be multiplied into the feed rate defined in the operation. It can be either a fixed value or a curve specifying a velocity profile along the toolpath.

its_technology: The technology defines the spindle and the feed of the tool.

its_machine_functions: Various machine functions, like coolant, chip removal, etc.

4.8.1.1 Toolpath type

Enumerator describing the type of the trajectory. This can be used by the controller e.g. to identify whether rapid motion may be allowed or not.

```
TYPE toolpath_type = ENUMERATION OF (
  approach,
  lift,
  connect,
  non_contact,
  contact,
  trajectory_path
);
END_TYPE;
```

The following values are used:

approach:	The movement towards the workpiece.
lift:	The movement away from the workpiece.
connect:	The connecting movement (usually in rapid speed) between the lift and the approach movement.
non_contact:	A movement without contact with the workpiece.
contact:	A movement with contact of the workpiece.
trajectory_path:	A movement described by a trajectory.

Note that any toolpath can be tagged with any of these types. For example, it is not necessary to use specialised toolpath elements like approach_lift_path in order to build an approach movement. This can just as well done through a pre-calculated cutter location path using the approach toolpath_type.

4.8.1.2 Toolpath speedprofile

Select type of the speed profile to influence the feed rate given by the attribute technology of either the operation or the toolpath itself.

```
TYPE toolpath_speedprofile = SELECT (
  toolpath_speed,
  positive_ratio_measure,
  speed_name
);
END_TYPE;
```

4.8.1.2.1 Toolpath speed

This is a one-dimensional curve specifying the speed profile as defined by type positive_ratio_measure. If the attribute basiccurve of the toolpath exists, the parameterisation of speed must be the same as basiccurve. Otherwise, speed must be parameterised by the path length of the underlying toolpath, i. e. the first parameter is 0, the last parameter equals the length of the underlying toolpath as defined by type length_measure.

```
ENTITY toolpath_speed;
  speed:                b_spline_curve;
WHERE
  WR1: speed\geometric_representation_item.dim = 1;
(*
Informal proposition:
- speed shall have only values greater than zero.
*)
END_ENTITY;
```

4.8.1.2.2 Speed name

Enumerator to specify rapid speed. It can be extended in the future to include other named speeds.

```
TYPE speed_name = ENUMERATION OF(RAPID);
END_TYPE;
```

4.8.2 Feedstop

Feedstop is a toolpath element with no feed motion. The spindle remains in the selected speed while the execution of the subsequent toolpath is delayed for the specified time.

```
ENTITY feedstop
  SUBTYPE OF (toolpath);
  dwell:                time_measure;
END_ENTITY;
```

dwell: The time to stop the feed axis.

4.8.3 Trajectory

Supertype for all explicit trajectories.

```
ENTITY trajectory
  ABSTRACT SUPERTYPE OF (ONEOF(cutter_location_trajectory,
    cutter_contact_trajectory, axis_trajectory))
  SUBTYPE OF (toolpath);
  its_direction: OPTIONAL BOOLEAN;
END_ENTITY;
```

its_direction: If true or omitted, the tool has to be moved from the beginning to the end of the referenced geometric curve. Otherwise, it will be moved from the end point to the start point of the referenced geometric curve.

4.8.4 Cutter location trajectory

Cutter location paths define a tool movement with respect to the tool center point. This is the point where the rotational axis of the tool leaves the tool tip.

```
ENTITY cutter_location_trajectory
  SUBTYPE OF (trajectory);
  basiccurve     :      bounded_curve;
  its_toolaxis   :      OPTIONAL bounded_curve;
  surface_normal :      OPTIONAL bounded_curve;
(*
Informal proposition:
its_toolaxis must have the same parameter range as basiccurve.
*)
END_ENTITY;
```

basiccurve: A 3D curve defining the cartesian co-ordinates of the cutter location point in the workpiece co-ordinate system or the local co-ordinate system of the feature, respectively.

its_toolaxis: A 3D curve defining the toolaxis of the tool. The three co-ordinates of the curve represent the three components of the vector in the workpiece co-ordinate system or the local co-ordinate system of the feature, respectively. If given, the parameterisation of the toolaxis curve must correspond to that of the basiccurve.

Note that this attribute will override any information about the tool orientation given by the its_tool_direction attribute of the workingstep.

surface_normal: A curve in space defining the surface normal. It has to be of the same type and has to have the same parameterisation than the basiccurve.

4.8.5 Cutter contact trajectory

Cutter contact paths define a tool movement with respect to the cutter contact point.

```
ENTITY cutter_contact_trajectory
  SUBTYPE OF (trajectory);
  basiccurve     :       curve_with_surface_normal;
  its_toolaxis   :       OPTIONAL bounded_curve;
  its_contact_type:      OPTIONAL contact_type;
(*
Informal proposition:
its_toolaxis must have the same parameter range as basiccurve.
*)
END_ENTITY;
```

basiccurve: A 3D curve defining the cartesian co-ordinates of the cutter contact point in the workpiece co-or

its_toolaxis: A 3D curve defining the toolaxis of the tool. The first co-ordinate represents the yaw angle, the second co-ordinate the tilt angle in degrees. For the definition of tilt and yaw angles, see Section 5.2.2.3 of ISO 14649-11. If given, the parameterisation of the toolaxis curve must correspond to that of the basiccurve. Note that this attribute will override any information about the tool orientation given by the its_tool_direction attribute of the workingstep.

4.8.5.1 Curve with surface normal

Select type for definition of a curve with surface normal. The bounded_pcurve is taken from AP42 and the curve_with_normal_vector is defined in this part of ISO 14649.

```
TYPE curve_with_surface_normal = SELECT (
  bounded_pcurve,  curve_with_normal_vector
);
END_TYPE;
```

4.8.5.2 Curve with normal vector

A bounded curve which has a second curve defining the surface normal.

```
ENTITY curve_with_normal_vector;
  basiccurve    :       bounded_curve;
  surface_normal:       bounded_curve;
(*
Informal propositions:
basiccurve and surface_normal must have the same parameter range
*)
END_ENTITY;
```

basiccurve: The reference to the geometric curve in the geometry description.

surface_normal: A 3D curve in space defining the surface normal. The three co-ordinates of the curve represent the three components of the surface normal vector in the co-ordinate system of the basiccurve. It has to be of the same type and has to have the same parameterisation than the basiccurve.

4.8.5.3 Contact type

Contact type specifies how the cutter contact point is determined. Depending on the mode of milling – plane milling, side milling or freeform milling – the cutter contact is desired at different locations of the tool, relative to its motion.

```
TYPE contact_type = ENUMERATION OF (side, front);
END_TYPE;
```

According to Figure 41 the interpretation of contact type is as follows:

side: Side milling. The cutter contact point is located at the circumference of the tool, perpendicular to the cutting direction. Whether the material can be found on the left-hand or right-hand side of the cutter depends on the direction of the surface normal specified by the toolpath's attribute basiccurve.

front: End milling as used for plane or freeform milling. The cutter contact point is a point at the bottom of the cutter. Note that such point is only properly defined when the cutter is inclined against the normal of the surface which it passes. If this is not the case, as for most two5D milling operations, the tool's forward-most contact point in the direction of the cut shall be regarded as cutter-contact point.

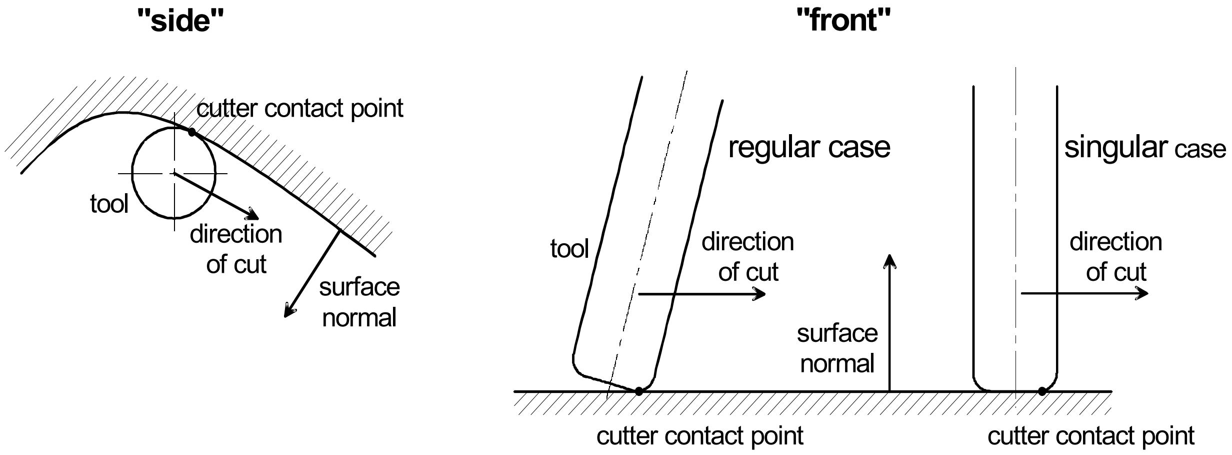


Figure 41.Contact type

4.8.6 Axis trajectory

Axis trajectories do not directly define a tool movement, but rather the movement of the machine axis. They are comparable to the G00/G01 commands of ISO 6983.

They should not be used because the resulting NC programme will no longer be machine independent.

```
ENTITY axis_trajectory
  SUBTYPE OF (trajectory);
  axis_list :           LIST [1:?] OF identifier;
  commands  :           LIST [1:?] OF bounded_curve;
WHERE
  WR1: SIZEOF(QUERY(cmd <* commands |
    cmd\geometric_representation_item.dim <> 1)) = 0;
END_ENTITY;
```

axis_list: A list with axis names defining these axes which will be moved by the trajectory.

commands: A list of one-dimensional curves. The array contains one curve per axis whose name is given by the respective index in the axis_list. The values represent axis positions in units as defined for length_measure (for linear axes) or planar_angle_measure (for rotary axes).

4.8.7 Parameterised path

This subtype of the toolpath entity is used to define help movements like connectors, approach and lift movements without giving the actual tool path. The movement is rather described by a movement type and its parameters. The CNC itself then has to calculate the resulting toolpath for the actual tool.

This concept is especially useful when working with cutter contact trajectories for the milling operations. As cutter contact trajectories can only be used for motions on the workpiece surface itself there is a need for defining the connecting movements between the individual toolpaths in an indirect way. Also, the parameterised paths given here can be used to connect cutter contact and cutter location trajectories.

Note that even though the parameterised paths are named for their most frequent use they can be used for any portion of a toolpath description just as any other form of toolpath. Their actual use is not indicated by the entity names but rather by the inherited its_type attribute.

```
ENTITY parameterised_path
  ABSTRACT SUPERTYPE OF (ONEOF (approach_lift_path, connector))
  SUBTYPE OF (toolpath);
END_ENTITY;
```

4.8.8 Connector

The connector entity defines a connection path between a lift and a following approach movement. This path type if typically needed to implement a rapid_movement operation but can also be used within a machining operation for connecting several cutting paths.

```
ENTITY connector
  ABSTRACT SUPERTYPE OF (ONEOF(connect_secplane, connect_direct))
  SUBTYPE OF (parameterised_path);
END_ENTITY;
```

4.8.8.1 Connect two points via the security plane

The security plane connector moves the tool up to the security plane until the cutter location point reaches the security plane, then linear to the projected starting point of the following movement and again from the security plane down to this starting point. This type of connection may be needed for unidirectional machining of freeform surface. For bidirectional machining, a connect_direct between the preceding lift and the following approach movement will be more efficient.

```
ENTITY connect_secplane
  SUBTYPE OF (connector);
  up_dir  : OPTIONAL direction;
  down_dir: OPTIONAL direction;
END_ENTITY;
```

up_dir: specification of the direction of the upward movement. If none is given, the motion will be in tool direction.

down_dir: specification of the direction of the downward movement. If none is given, the motion will be in tool direction.

Note: If this connector is used during a five-axis milling operation, it will interpolate the tool direction between the previous and the next cut. In this case, the tool will retract in the tool direction from the last point of the previous cut (unless overridden by up_dir). On the security the tool will rotate to the new tool direction, and then it will move down in the tool direction (unless overridden by down_dir) for the first point of the next cut. No change of the tool direction takes place during the lift and approach to and from the security plane.

4.8.8.2 Connect two points directly

The direct connector moves the tool in a straight line from the end point of the previous toolpath to the starting point of the next toolpath.

```
ENTITY connect_direct
  SUBTYPE OF (connector);
END_ENTITY;
```

Note: If this connector is used during a five-axis milling operation, the connector may be used to adjust the tool direction between the previous and the next cut. The tool will rotate to the new tool direction during the linear motion.

4.8.9 Approach and lift path

This subtype of parameterised path describes the lift and approach movement in terms of angles or tangential movements. See also Section 4.8.1. The strategy will initiate the generation of toolpath based on the surrounding feature. This entity, on the other hand, specifies a distinct path in cartesian space. It can be the result of a path generation based on an air strategy in which case the fix_point would lie in the retract_plane of the associated operation.

```
ENTITY approach_lift_path
  ABSTRACT SUPERTYPE OF (ONEOF (ap_lift_path_angle, ap_lift_path_tangent))
  SUBTYPE OF (parameterised_path);
  fix_point    : cartesian_point;
  fix_point_dir: OPTIONAL direction;
END_ENTITY;
```

fix_point: The starting point for the approach movement or the destination for the lift movement in the workpiece co-ordinate system, or the feature's local co-ordinate system, respectively. The reference point on the tool is the tool center point (tcp), which is the point where the rotational axis of the tool leaves the tip.

fix_point_dir: The tool direction in fix_point as 3D vector in the workpiece co-ordinate system, or the feature's local co-ordinate system, respectively. If given, the machine will attempt to interpolate between the tool direction in the fix point and the tool direction in the start (approach) or end (lift) point of the connecting toolpath during the approach/lift movement. If not given, the tool direction will not be changed during the approach/lift movement.

4.8.9.1 Approach lift angle

Approach or lift at an angle to the path using a linear approach (see figure).

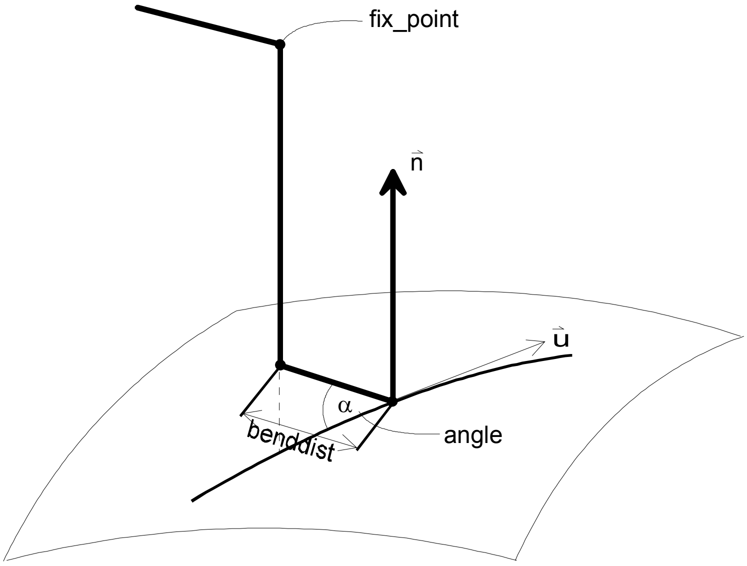


Figure 42.Linear approach with angle

```
ENTITY ap_lift_path_angle
   SUBTYPE OF (approach_lift_path);
   angle       : plane_angle_measure;
   benddist    : positive_length_measure;
END_ENTITY;
```

angle: Approach or lift angle.

benddist: The length of the angular approach angle.

4.8.9.2 Approach lift tangent

Approach or lift to the path using a tangential approach (see figure).

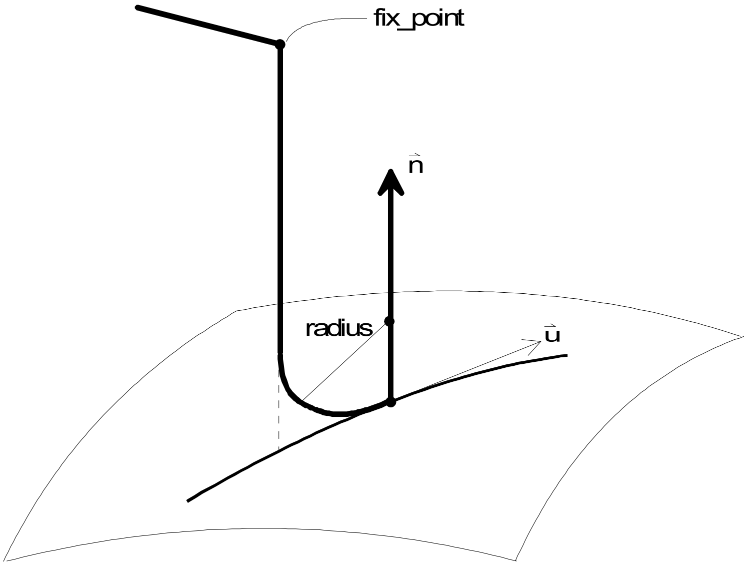


Figure 43.Tangential approach

```
ENTITY ap_lift_path_tangent
  SUBTYPE OF (approach_lift_path);
  radius: positive_length_measure;
END_ENTITY;
```

radius: The radius of the approach movement.

4.9 Rules

The following rules are needed to ensure the consistency of the geometric, topological and technological model.

4.9.1 dependent_instantiable_representation_item

The dependent_instantiable_representation_item rule specifies that all instances of representation_item are dependent on the usage to define another entity.

```
RULE dependent_instantiable_representation_item FOR (representation_item);
WHERE
  WR1: SIZEOF (QUERY (ri <* representation_item |
        NOT (SIZEOF (USEDIN (ri, '')) >= 1))) = 0;
END_RULE;
```

4.9.2 dependent_instantiable_shape_representation

The dependent_instantiable_shape_representation rule specifies that all instances of shape_representation are dependent on the usage to define another entity.

```
RULE dependent_instantiable_shape_representation FOR (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT (SIZEOF(USEDIN(sr, '')) >= 1))) = 0;
END_RULE;
```

4.9.3 geometric_representation_item_3d

The geometric_representation_item_3d rule specifies that every geometric_representation_item must be founded within a geometric_representation_context that has a dimensionality of 3 except for use in defining the pcurve entity. This rule constrains all geometry to be three dimensional. The pcurve entity is an exception because it must be founded in a 2 dimensional context which is the parameter space of a surface.

```
RULE geometric_representation_item_3d FOR (geometric_representation_item);
WHERE
  WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
       NOT ((dimension_of (gri) = 3) OR
       (SIZEOF (bag_to_set (USEDIN (gri, '')) - bag_to_set (USEDIN (gri,
       'MACHINING_SCHEMA.DEFINITIONAL_REPRESENTATION.ITEMS'))) = 0)
       ))) = 0;
END_RULE;
```

5 Conformance requirements

Conformance to this part of ISO 14649 includes satisfying the requirements stated in this part, the requirements of the implementation methods supported, and the relevant requirements of the normative references.

For requirements with respect to implementation methods see Annex C.

This part of ISO 14649 provides a number of options that may be supported by an implementation. These options shall all be supported by six classes of conformance.

These conformance classes are characterized as follows:

- conformance class 1 – `c0m0`:Minimum set of curve geometry and minimum set of manufacturing data;
- conformance class 2 – `c0m0m1`:Class 1 plus full set of manufacturing data, especially manufacturing features;
- conformance class 3 – `s0c0m0m1`:Class 2 plus minimum set of surface geometry;
- conformance class 4 – `s0s1c0c1m0m1`:Class 3 plus full curve and surface geometry;
- conformance class 5 – `t0t1s0c0m0m1`:Class 3 plus topological information;
- conformance class 6 – `t0t1s0s1c0c1m0m1`:Class 4 plus topological information.

The identifiers for the respective data sets (`m0/m1`, `c0/c1`, `s0/s1`, `t0/t1`) can be found in the expanded EXPRESS listing in Appendix A for each member of the data set in order to facilitate implementation.

5.1 Conformance class 1 entities

An implementation of conformance class 1 of this part of ISO 14649 shall support the following entities and related constructs:

```
address
approval
approval_status
axis_trajectory
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
bounded_curve
calendar_date
cartesian_point
channel
composite_curve
composite_curve_segment
coordinated_universal_time_offset
curve
cutter_location_trajectory
cutting_component
cutting_edge_technological_data
cutting_tool
date
date_and_time
descriptive_parameter
direction
display_message
elementary_surface
executable
feedstop
geometric_representation_context
geometric_representation_item
local_time
machine_functions
machining_operation
machining_tool
machining_workingstep
nc_function
numeric_parameter
operation
optional_stop
ordinal_date
parameterised_path
person
person_and_address
placement
plane
point
polyline
program_stop
program_structure
project
property_parameter
rapid_movement
rational_b_spline_curve
representation_item
return_home
set_mark
setup
surface
technology
three_axes
tool_body
tool_direction
toolpath
toolpath_list
trajectory
trimmed_curve
two_axes
wait_for_mark
week_of_year_and_day_date
workingstep
workpiece_setup
```

5.2 Conformance class 2 entities

An implementation of conformance class 2 of this part of ISO 14649 shall support the following entities and related constructs:

```
address
and_expression
angle_taper
ap_lift_path_angle
ap_lift_path_tangent
approach_lift_path
approval
approval_status
assignment
axis_trajectory
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
binary_boolean_expression
blind_bottom_condition
block
boolean_expression
boss
bounded_curve
calendar_date
cartesian_point
catalogue_thread
chamfer
channel
circular_closed_profile
circular_closed_shape_profile
```

```
circular_offset
circular_omit
circular_path
circular_pattern
closed_pocket
closed_profile
comparison_equal
comparison_expression
comparison_greater
comparison_greater_equal
comparison_less
comparison_less_equal
comparison_not_equal
complete_circular_path
composite_curve
composite_curve_segment
compound_feature
conical_hole_bottom
connect_direct
connect_secplane
connector
coordinated_universal_time_offset
counterbore_hole
countersunk_hole
curve
curve_with_normal_vector
cutter_contact_trajectory
cutter_location_trajectory
cutting_component
cutting_edge_technological_data
cutting_tool
date
date_and_time
defined_thread
descriptive_parameter
diameter_taper
direction
display_message
edge_round
elementary_surface
executable
feedstop
flat_hole_bottom
flat_slot_end_type
flat_with_radius_hole_bottom
general_closed_profile
general_outside_profile
general_path
general_pattern
general_pocket_bottom_condition
general_profile
general_profile_floor
general_shape_profile
geometric_representation_context
geometric_representation_item
hole_bottom_condition
if_statement
in_process_geometry
limits_and_fits
linear_path
linear_profile
local_time
loop_slot_end_type
machine_functions
machined_surface
machining_feature
machining_operation
machining_tool
machining_workingstep
manufacturing_feature
material
multiple_arity_boolean_expression
nc_constant
nc_function
nc_variable
ngon_profile
non_sequential
not_expression
numeric_parameter
offset_vector
open_pocket
open_profile
open_slot_end_type
operation
optional_stop
or_expression
ordinal_date
parallel
parameterised_path
partial_area_definition
partial_circular_path
partial_circular_profile
partial_circular_shape_profile
person
person_and_address
placement
planar_face
planar_pocket_bottom_condition
planar_profile_floor
plane
plus_minus_value
pocket
pocket_bottom_condition
point
polyline
profile
profile_feature
profile_floor
program_stop
program_structure
project
property_parameter
radiused_pocket_bottom_condition
radiused_slot_end_type
rapid_movement
rational_b_spline_curve
rectangular_closed_profile
rectangular_closed_shape_profile
rectangular_offset
rectangular_omit
rectangular_open_shape_profile
rectangular_pattern
```

```
region
region_projection
region_surface_list
replicate_feature
representation_item
return_home
right_circular_cylinder
round_hole
rounded_end
rounded_u_profile
selective
set_mark
setup
setup_instruction
shape_profile
slot
slot_end_type
specification
specification_usage_constraint
spherical_cap
spherical_hole_bottom
square_u_profile
step
surface
surface_texture_parameter
technology
tee_profile
thread
three_axes
through_bottom_condition
through_pocket_bottom_condition
through_profile_floor
toleranced_length_measure
tool_body
tool_direction
tool_length_probing
tool_probing
tool_radius_probing
toolpath
toolpath_feature
toolpath_list
toolpath_speed
touch_probe
touch_probing
trajectory
transition_feature
travel_path
trimmed_curve
two_axes
two5D_manufacturing_feature
unary_boolean_expression
vee_profile
wait_for_mark
week_of_year_and_day_date
while_statement
woodruff_slot_end_type
workingstep
workpiece
workpiece_complete_probing
workpiece_probing
workpiece_setup
workplan
xor_expression
```

5.3 Conformance class 3 entities

An implementation of conformance class 3 of this part of ISO 14649 shall support the following entities and related constructs:

```
address
and_expression
angle_taper
ap_lift_path_angle
ap_lift_path_tangent
approach_lift_path
approval
approval_status
assignment
axis_trajectory
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
b_spline_surface
b_spline_surface_with_knots
binary_boolean_expression
blind_bottom_condition
block
boolean_expression
boss
bounded_curve
bounded_pcurve
bounded_surface
calendar_date
cartesian_point
catalogue_thread
chamfer
channel
circular_closed_profile
circular_closed_shape_profile
circular_offset
circular_omit
circular_path
circular_pattern
closed_pocket
closed_profile
comparison_equal
comparison_expression
comparison_greater
comparison_greater_equal
comparison_less
comparison_less_equal
```

```
comparison_not_equal
complete_circular_path
composite_curve
composite_curve_segment
compound_feature
conical_hole_bottom
connect_direct
connect_secplane
connector
coordinated_universal_time_offset
counterbore_hole
countersunk_hole
curve
curve_with_normal_vector
cutter_contact_trajectory
cutter_location_trajectory
cutting_component
cutting_edge_technological_data
cutting_tool
date
date_and_time
defined_thread
definitional_representation
descriptive_parameter
diameter_taper
direction
display_message
edge_round
elementary_surface
executable
feedstop
flat_hole_bottom
flat_slot_end_type
flat_with_radius_hole_bottom
general_closed_profile
general_outside_profile
general_path
general_pattern
general_pocket_bottom_condition
general_profile
general_profile_floor
general_shape_profile
geometric_representation_context
geometric_representation_item
hole_bottom_condition
if_statement
in_process_geometry
limits_and_fits
linear_path
linear_profile
local_time
loop_slot_end_type
machine_functions
machined_surface
machining_feature
machining_operation
machining_tool
machining_workingstep
manufacturing_feature
material
multiple_arity_boolean_expression
nc_constant
nc_function
nc_variable
ngon_profile
non_sequential
not_expression
numeric_parameter
offset_vector
open_pocket
open_profile
open_slot_end_type
operation
optional_stop
or_expression
ordinal_date
parallel
parameterised_path
partial_area_definition
partial_circular_path
partial_circular_profile
partial_circular_shape_profile
pcurve
person
person_and_address
placement
planar_face
planar_pocket_bottom_condition
planar_profile_floor
plane
plus_minus_value
pocket
pocket_bottom_condition
point
polyline
profile
profile_feature
profile_floor
program_stop
program_structure
project
property_parameter
radiused_pocket_bottom_condition
radiused_slot_end_type
rapid_movement
rational_b_spline_curve
rational_b_spline_surface
rectangular_closed_profile
rectangular_closed_shape_profile
rectangular_offset
rectangular_omit
rectangular_open_shape_profile
rectangular_pattern
region
region_projection
region_surface_list
replicate_feature
representation_item
return_home
right_circular_cylinder
round_hole
rounded_end
```

rounded_u_profile
selective
set_mark
setup
setup_instruction
shape_profile
slot
slot_end_type
specification
specification_usage_constraint
spherical_cap
spherical_hole_bottom
square_u_profile
step
surface
surface_texture_parameter
technology
tee_profile
thread
three_axes
through_bottom_condition
through_pocket_bottom_condition
through_profile_floor
toleranced_length_measure
tool_body
tool_direction
tool_length_probing
tool_probing
tool_radius_probing
toolpath
toolpath_feature
toolpath_list
toolpath_speed
touch_probe
touch_probing
trajectory
transition_feature
travel_path
trimmed_curve
two_axes
two5D_manufacturing_feature
unary_boolean_expression
vee_profile
wait_for_mark
week_of_year_and_day_date
while_statement
woodruff_slot_end_type
workingstep
workpiece
workpiece_complete_probing
workpiece_probing
workpiece_setup
workplan
xor_expression

5.4 Conformance class 4 entities

An implementation of conformance class 4 of this part of ISO 14649 shall support the following entities and related constructs:

address
and_expression
angle_taper
ap_lift_path_angle
ap_lift_path_tangent
approach_lift_path
approval
approval_status
assignment
axis_trajectory
axis1_placement
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
b_spline_surface
b_spline_surface_with_knots
bezier_curve
bezier_surface
binary_boolean_expression
blind_bottom_condition
block
boolean_expression
boss
bounded_curve
bounded_pcurve
bounded_surface
calendar_date
cartesian_point
catalogue_thread
chamfer
channel
circle
circular_closed_profile
circular_closed_shape_profile
circular_offset
circular_omit
circular_path
circular_pattern
closed_pocket
closed_profile
comparison_equal
comparison_expression
comparison_greater
comparison_greater_equal
comparison_less
comparison_less_equal
comparison_not_equal
complete_circular_path
composite_curve
composite_curve_segment
compound_feature
conic
conical_hole_bottom
connect_direct

```
connect_secplane
connector
coordinated_universal_time_offset
counterbore_hole
countersunk_hole
curve
curve_with_normal_vector
cutter_contact_trajectory
cutter_location_trajectory
cutting_component
cutting_edge_technological_data
cutting_tool
date
date_and_time
defined_thread
definitional_representation
descriptive_parameter
diameter_taper
direction
display_message
edge_round
elementary_surface
ellipse
executable
feedstop
flat_hole_bottom
flat_slot_end_type
flat_with_radius_hole_bottom
general_closed_profile
general_outside_profile
general_path
general_pattern
general_pocket_bottom_condition
general_profile
general_profile_floor
general_shape_profile
geometric_representation_context
geometric_representation_item
hole_bottom_condition
hyperbola
if_statement
in_process_geometry
limits_and_fits
line
linear_path
linear_profile
local_time
loop_slot_end_type
machine_functions
machined_surface
machining_feature
machining_operation
machining_tool
machining_workingstep
manufacturing_feature
material
multiple_arity_boolean_expression
nc_constant
nc_function
nc_variable
ngon_profile
non_sequential
not_expression
numeric_parameter
offset_vector
open_pocket
open_profile
open_slot_end_type
operation
optional_stop
or_expression
ordinal_date
oriented_surface
parabola
parallel
parameterised_path
partial_area_definition
partial_circular_path
partial_circular_profile
partial_circular_shape_profile
pcurve
person
person_and_address
placement
planar_face
planar_pocket_bottom_condition
planar_profile_floor
plane
plus_minus_value
pocket
pocket_bottom_condition
point
polyline
profile
profile_feature
profile_floor
program_stop
program_structure
project
property_parameter
quasi_uniform_curve
quasi_uniform_surface
radiused_pocket_bottom_condition
radiused_slot_end_type
rapid_movement
rational_b_spline_curve
rational_b_spline_surface
rectangular_closed_profile
rectangular_closed_shape_profile
rectangular_offset
rectangular_omit
rectangular_open_shape_profile
rectangular_pattern
region
region_projection
region_surface_list
replicate_feature
representation_item
return_home
right_circular_cylinder
round_hole
rounded_end
```

```
rounded_u_profile
selective
set_mark
setup
setup_instruction
shape_profile
slot
slot_end_type
specification
specification_usage_constraint
spherical_cap
spherical_hole_bottom
spherical_surface
square_u_profile
step
surface
surface_of_linear_extrusion
surface_of_revolution
surface_texture_parameter
swept_surface
technology
tee_profile
thread
three_axes
through_bottom_condition
through_pocket_bottom_condition
through_profile_floor
toleranced_length_measure
tool_body
tool_direction
tool_length_probing
tool_probing
tool_radius_probing
toolpath
toolpath_feature
toolpath_list
toolpath_speed
touch_probe
touch_probing
trajectory
transition_feature
travel_path
trimmed_curve
two_axes
two5D_manufacturing_feature
unary_boolean_expression
uniform_curve
uniform_surface
vector
vee_profile
wait_for_mark
week_of_year_and_day_date
while_statement
woodruff_slot_end_type
workingstep
workpiece
workpiece_complete_probing
workpiece_probing
workpiece_setup
workplan
xor_expression
```

5.5 Conformance class 5 entities

An implementation of conformance class 5 of this part of ISO 14649 shall support the following entities and related constructs:

```
address
advanced_brep_shape_representation
advanced_face
and_expression
angle_taper
ap_lift_path_angle
ap_lift_path_tangent
approach_lift_path
approval
approval_status
assignment
axis_trajectory
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
b_spline_surface
b_spline_surface_with_knots
binary_boolean_expression
blind_bottom_condition
block
boolean_expression
boss
bounded_curve
bounded_pcurve
bounded_surface
calendar_date
cartesian_point
catalogue_thread
chamfer
channel
circular_closed_profile
circular_closed_shape_profile
circular_offset
circular_omit
circular_path
circular_pattern
closed_pocket
closed_profile
closed_shell
comparison_equal
comparison_expression
comparison_greater
comparison_greater_equal
comparison_less
comparison_less_equal
comparison_not_equal
```

complete_circular_path
composite_curve
composite_curve_segment
compound_feature
conical_hole_bottom
connect_direct
connect_secplane
connected_face_set
connector
coordinated_universal_time_offset
counterbore_hole
countersunk_hole
curve
curve_with_normal_vector
cutter_contact_trajectory
cutter_location_trajectory
cutting_component
cutting_edge_technological_data
cutting_tool
date
date_and_time
defined_thread
definitional_representation
descriptive_parameter
diameter_taper
direction
display_message
edge
edge_curve
edge_loop
edge_round
elementary_surface
executable
face
face_bound
face_outer_bound
face_surface
faceted_brep
feedstop
flat_hole_bottom
flat_slot_end_type
flat_with_radius_hole_bottom
founded_item
general_closed_profile
general_outside_profile
general_path
general_pattern
general_pocket_bottom_condition
general_profile
general_profile_floor
general_shape_profile
geometric_representation_context
geometric_representation_item
hole_bottom_condition
if_statement
in_process_geometry
limits_and_fits
linear_path
linear_profile
local_time
loop
loop_slot_end_type
machine_functions
machined_surface
machining_feature
machining_operation
machining_tool
machining_workingstep
manifold_solid_brep
manufacturing_feature
mapped_item
material
multiple_arity_boolean_expression
nc_constant
nc_function
nc_variable
ngon_profile
non_sequential
not_expression
numeric_parameter
offset_vector
open_pocket
open_profile
open_shell
open_slot_end_type
operation
optional_stop
or_expression
ordinal_date
oriented_closed_shell
oriented_edge
oriented_face
orineted_open_shell
oriented_path
parallel
parameterised_path
partial_area_definition
partial_circular_path
partial_circular_profile
partial_circular_shape_profile
path
pcurve
person
person_and_address
placement
planar_face
planar_pocket_bottom_condition
planar_profile_floor
plane
plus_minus_value
pocket
pocket_bottom_condition
point
polyline
poly_loop
profile
profile_feature
profile_floor
program_stop
program_structure
project
property_parameter

```
radiused_pocket_bottom_condition
radiused_slot_end_type
rapid_movement
rational_b_spline_curve
rational_b_spline_surface
rectangular_closed_profile
rectangular_closed_shape_profile
rectangular_offset
rectangular_omit
rectangular_open_shape_profile
rectangular_pattern
region
region_projection
region_surface_list
replicate_feature
representation_context
representation_item
representation_map
return_home
right_circular_cylinder
round_hole
rounded_end
rounded_u_profile
selective
set_mark
setup
setup_instruction
shape_profile
shape_representation
slot
slot_end_type
solid_model
specification
specification_usage_constraint
spherical_cap
spherical_hole_bottom
square_u_profile
step
surface
surface_texture_parameter
technology
tee_profile
thread
three_axes
through_bottom_condition
through_pocket_bottom_condition
through_profile_floor
toleranced_length_measure
tool_body
tool_direction
tool_length_probing
tool_probing
tool_radius_probing
toolpath
toolpath_feature
toolpath_list
toolpath_speed
topological_region
topological_representation_item
touch_probe
touch_probing
trajectory
transition_feature
travel_path
trimmed_curve
two_axes
two5D_manufacturing_feature
unary_boolean_expression
vee_profile
vertex
vertex_loop
vertex_point
wait_for_mark
week_of_year_and_day_date
while_statement
woodruff_slot_end_type
workingstep
workpiece
workpiece_complete_probing
workpiece_probing
workpiece_setup
workplan
xor_expression
```

5.6 Conformance class 6 entities

An implementation of conformance class 6 of this part of ISO 14649 shall support the following entities and related constructs:

```
address
advanced_brep_shape_representation
advanced_face
and_expression
angle_taper
ap_lift_path_angle
ap_lift_path_tangent
approach_lift_path
approval
approval_status
assignment
axis_trajectory
axis1_placement
axis2_placement_3d
b_spline_curve
b_spline_curve_with_knots
b_spline_surface
b_spline_surface_with_knots
bezier_curve
bezier_surface
binary_boolean_expression
blind_bottom_condition
```

```
  block
  boolean_expression
  boss
  bounded_curve
  bounded_pcurve
  bounded_surface
  calendar_date
  cartesian_point
  catalogue_thread
  chamfer
  channel
  circle
  circular_closed_profile
  circular_closed_shape_profile
  circular_offset
  circular_omit
  circular_path
  circular_pattern
  closed_pocket
  closed_profile
  closed_shell
  comparison_equal
  comparison_expression
  comparison_greater
  comparison_greater_equal
  comparison_less
  comparison_less_equal
  comparison_not_equal
  complete_circular_path
  composite_curve
  composite_curve_segment
  compound_feature
  conic
  conical_hole_bottom
  connect_direct
  connect_secplane
  connected_face_set
  connector
  coordinated_universal_time_offset
  counterbore_hole
  countersunk_hole
  curve
  curve_with_normal_vector
  cutter_contact_trajectory
  cutter_location_trajectory
  cutting_component
  cutting_edge_technological_data
  cutting_tool
  date
  date_and_time
  defined_thread
  definitional_representation
  descriptive_parameter
  diameter_taper
  direction
  display_message
  edge
  edge_curve
  edge_loop
  edge_round
  elementary_surface
  ellipse
  executable
  face
  face_bound
  face_outer_bound
  face_surface
  faceted_brep
  feedstop
  flat_hole_bottom
  flat_slot_end_type
  flat_with_radius_hole_bottom
  founded_item
  general_closed_profile
  general_outside_profile
  general_path
  general_pattern
  general_pocket_bottom_condition
  general_profile
  general_profile_floor
  general_shape_profile
  geometric_representation_context
  geometric_representation_item
  hole_bottom_condition
  hyperbola
  if_statement
  in_process_geometry
  limits_and_fits
  line
  linear_path
  linear_profile
  local_time
  loop
  loop_slot_end_type
  machine_functions
  machined_surface
  machining_feature
  machining_operation
  machining_tool
  machining_workingstep
  manifold_solid_brep
  manufacturing_feature
  mapped_item
  material
  multiple_arity_boolean_expression
  nc_constant
  nc_function
  nc_variable
  ngon_profile
  non_sequential
  not_expression
  numeric_parameter
  offset_vector
  open_pocket
  open_profile
  open_shell
  open_slot_end_type
  operation
  optional_stop
  or_expression
  ordinal_date
  oriented_closed_shell
```

oriented_edge
oriented_face
orineted_open_shell
oriented_path
oriented_surface
parabola
parallel
parameterised_path
partial_area_definition
partial_circular_path
partial_circular_profile
partial_circular_shape_profile
path
pcurve
person
person_and_address
placement
planar_face
planar_pocket_bottom_condition
planar_profile_floor
plane
plus_minus_value
pocket
pocket_bottom_condition
point
polyline
poly_loop
profile
profile_feature
profile_floor
program_stop
program_structure
project
property_parameter
quasi_uniform_curve
quasi_uniform_surface
radiused_pocket_bottom_condition
radiused_slot_end_type
rapid_movement
rational_b_spline_curve
rational_b_spline_surface
rectangular_closed_profile
rectangular_closed_shape_profile
rectangular_offset
rectangular_omit
rectangular_open_shape_profile
rectangular_pattern
region
region_projection
region_surface_list
replicate_feature
representation_context
representation_item
representation_map
return_home
right_circular_cylinder
round_hole
rounded_end
rounded_u_profile
selective
set_mark
setup
setup_instruction
shape_profile
shape_representation
slot
slot_end_type
solid_model
specification
specification_usage_constraint
spherical_cap
spherical_hole_bottom
spherical_surface
square_u_profile
step
surface
surface_of_linear_extrusion
surface_of_revolution
surface_texture_parameter
swept_surface
technology
tee_profile
thread
three_axes
through_bottom_condition
through_pocket_bottom_condition
through_profile_floor
toleranced_length_measure
tool_body
tool_direction
tool_length_probing
tool_probing
tool_radius_probing
toolpath
toolpath_feature
toolpath_list
toolpath_speed
topological_region
topological_representation_item
touch_probe
touch_probing
trajectory
transition_feature
travel_path
trimmed_curve
two_axes
two5D_manufacturing_feature
unary_boolean_expression
uniform_curve
uniform_surface
vector
vee_profile
vertex
vertex_loop
vertex_point
wait_for_mark
week_of_year_and_day_date
while_statement
woodruff_slot_end_type
workingstep
workpiece
workpiece_complete_probing

```
workpiece_probing
workpiece_setup
workplan
xor_expression
```

Annex A
(normative)

EXPRESS expanded listing

The following EXPRESS is the whole schema given in clause 5. In the event of any discrepancy between the short form and this expanded listing, the expanded listing shall be used. The two-character labels used for each entity indicate to which conformance class an entity belongs; please refer to Chapter 5.

```
SCHEMA machining_schema;
(* Version 5 date: 2002-02-06
 * Author: ISO TC184/SC1/WG7
 *)

   (* ********************************************************** *)
   (* CONSTANT from geometry_schema                 ISO 10303-42 *)
   (* ********************************************************** *)

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


   (* ********************************************************** *)
   (* Types from approval_schema                    ISO 10303-41 *)
   (* ********************************************************** *)

ENTITY approval;                                                   (* m0 *)
  status :                    approval_status;
  level  :                    label;
END_ENTITY;

ENTITY approval_status;                                            (* m0 *)
  name :                      label;
END_ENTITY;

   (* ********************************************************** *)
   (* Types from date_time_schema                   ISO 10303-41 *)
   (* ********************************************************** *)

ENTITY date_and_time;                                              (* m0 *)
  date_component :            date;
  time_component :            local_time;
END_ENTITY;

ENTITY date                                                        (* m0 *)
  SUPERTYPE OF (ONEOF (calendar_date, ordinal_date, week_of_year_and_day_date));
  year_component :            year_number;
END_ENTITY;

ENTITY calendar_date                                               (* m0 *)
```

```
  SUBTYPE OF (date);
  day_component   :           day_in_month_number;
  month_component :           month_in_year_number;
  WHERE
    WR1: valid_calendar_date (SELF);
END_ENTITY;

TYPE day_in_month_number = INTEGER;
  WHERE
    WR1: {1 <= SELF <= 31};
END_TYPE;

TYPE month_in_year_number = INTEGER;
  WHERE
    WR1: { 1 <= SELF <= 12 };
END_TYPE;

ENTITY ordinal_date                                                         (* m0 *)
  SUBTYPE OF (date);
  day_component : day_in_year_number;
  WHERE
    WR1: (NOT leap_year(SELF.year_component) AND { 1 <= day_component <= 365 })
         OR
         (leap_year(SELF.year_component) AND { 1 <= day_component <= 366 });
END_ENTITY;

TYPE day_in_year_number = INTEGER;
  WHERE
    WR1: {1 <= SELF <= 366};
END_TYPE;

ENTITY week_of_year_and_day_date                                            (* m0 *)
  SUBTYPE OF (date);
  week_component :            week_in_year_number;
  day_component  :            OPTIONAL day_in_week_number;
END_ENTITY;

TYPE week_in_year_number = INTEGER;
  WHERE
    WR1: { 1 <= SELF <= 53 };
END_TYPE;

TYPE day_in_week_number = INTEGER;
  WHERE
    WR1: { 1 <= SELF <= 7 };
END_TYPE; -- day_in_week_number

ENTITY local_time;                                                          (* m0 *)
  hour_component   :          hour_in_day;
  minute_component :          OPTIONAL minute_in_hour;
  second_component :          OPTIONAL second_in_minute;
  zone             :          coordinated_universal_time_offset;
  WHERE
    WR1: valid_time (SELF);
END_ENTITY;

TYPE hour_in_day = INTEGER;
  WHERE
    WR1: { 0 <= SELF < 24 };
END_TYPE;
```

```
TYPE minute_in_hour = INTEGER;
  WHERE
    WR1: { 0 <= SELF <= 59 };
END_TYPE;

TYPE second_in_minute = REAL;
  WHERE
    WR1: { 0 <= SELF <= 60 };
END_TYPE;

ENTITY coordinated_universal_time_offset;                                     (* m0 *)
  hour_offset   :             INTEGER;
  minute_offset :             OPTIONAL INTEGER;
  sense         :             ahead_or_behind;
  WHERE
    WR1: { 0 <= hour_offset <= 12 };
    WR2: { 0 <= minute_offset <= 59 };
    WR3: NOT ((hour_offset <> 0) AND (sense = exact));
END_ENTITY;

TYPE ahead_or_behind = ENUMERATION OF(ahead, exact, behind);
END_TYPE;

TYPE year_number = INTEGER;
END_TYPE;

FUNCTION valid_calendar_date (date : calendar_date) : LOGICAL;
  CASE date.month_component OF
    1  : RETURN({ 1 <= date.day_component <= 31 });
    2  : BEGIN
           IF (leap_year(date.year_component)) THEN
             RETURN({ 1 <= date.day_component <= 29 });
           ELSE
             RETURN({ 1 <= date.day_component <= 28 });
           END_IF;
         END;
    3  : RETURN({ 1 <= date.day_component <= 31 });
    4  : RETURN({ 1 <= date.day_component <= 30 });
    5  : RETURN({ 1 <= date.day_component <= 31 });
    6  : RETURN({ 1 <= date.day_component <= 30 });
    7  : RETURN({ 1 <= date.day_component <= 31 });
    8  : RETURN({ 1 <= date.day_component <= 31 });
    9  : RETURN({ 1 <= date.day_component <= 30 });
    10 : RETURN({ 1 <= date.day_component <= 31 });
    11 : RETURN({ 1 <= date.day_component <= 30 });
    12 : RETURN({ 1 <= date.day_component <= 31 });
  END_CASE;
       RETURN (FALSE);
END_FUNCTION;

FUNCTION leap_year (year : year_number) : BOOLEAN;
  IF ((((year MOD 4) = 0) AND ((year MOD 100) <> 0)) OR
       ((year MOD 400) = 0)) THEN
    RETURN(TRUE);
  ELSE
    RETURN(FALSE);
  END_IF;
END_FUNCTION;

FUNCTION valid_time (time: local_time) : BOOLEAN;
  IF EXISTS (time.second_component) THEN
```

```
    RETURN (EXISTS (time.minute_component));
  ELSE
    RETURN (TRUE);
  END_IF;
END_FUNCTION;

   (* ************************************************************ *)
   (* Types from person_organization_schema           ISO 10303-41 *)
   (* ************************************************************ *)

ENTITY person;                                                               (* m0 *)
  id            :             identifier;
  last_name     :             OPTIONAL label;
  first_name    :             OPTIONAL label;
  middle_names  :             OPTIONAL LIST [1:?] OF label;
  prefix_titles :             OPTIONAL LIST [1:?] OF label;
  suffix_titles :             OPTIONAL LIST [1:?] OF label;
  WHERE
    WR1: EXISTS(last_name) OR EXISTS(first_name);
END_ENTITY;

ENTITY address;                                                              (* m0 *)
  internal_location       :   OPTIONAL label;
  street_number           :   OPTIONAL label;
  street                  :   OPTIONAL label;
  postal_box              :   OPTIONAL label;
  town                    :   OPTIONAL label;
  region                  :   OPTIONAL label;
  postal_code             :   OPTIONAL label;
  country                 :   OPTIONAL label;
  facsimile_number        :   OPTIONAL label;
  telephone_number        :   OPTIONAL label;
  electronic_mail_address :   OPTIONAL label;
  telex_number            :   OPTIONAL label;
  WHERE
    WR1: EXISTS(internal_location)       OR
         EXISTS(street_number)           OR
         EXISTS(street)                  OR
         EXISTS(postal_box)              OR
         EXISTS(town)                    OR
         EXISTS(region)                  OR
         EXISTS(postal_code)             OR
         EXISTS(country)                 OR
         EXISTS(facsimile_number)        OR
         EXISTS(telephone_number)        OR
         EXISTS(electronic_mail_address) OR
         EXISTS(telex_number);
END_ENTITY;

   (* ************************************************************ *)
   (* Types from support_resource_schema              ISO 10303-41 *)
   (* ************************************************************ *)

TYPE identifier = STRING;
END_TYPE;

TYPE label = STRING;
END_TYPE;

TYPE text = STRING;
END_TYPE;
```

```
FUNCTION bag_to_set
  (the_bag : BAG OF GENERIC : intype) : SET OF GENERIC : intype;
  LOCAL
      the_set: SET OF GENERIC : intype := [];
  END_LOCAL;
  IF SIZEOF (the_bag) > 0 THEN
     REPEAT i := 1 to HIINDEX (the_bag);
       the_set := the_set + the_bag [i];
     END_REPEAT;
  END_IF;
  RETURN (the_set);
END_FUNCTION;


  (* ****************************************************** *)
  (* Types from measure_schema                ISO 10303-41 *)
  (* ****************************************************** *)

TYPE length_measure = REAL;
END_TYPE;

TYPE parameter_value = REAL;
END_TYPE;

TYPE plane_angle_measure = REAL;
END_TYPE;

TYPE positive_length_measure = length_measure;
  WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE positive_ratio_measure = ratio_measure;
  WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE ratio_measure = REAL;
END_TYPE;

TYPE time_measure = REAL;
END_TYPE;


  (* ******************************************************** *)
  (* Types from product_property_representation_schema ISO 10303-41 *)
  (* ******************************************************** *)

ENTITY shape_representation                                   (* t1 *)
  SUBTYPE OF (representation);
END_ENTITY;


  (* ****************************************************** *)
  (* Types from geometry_schema               ISO 10303-42 *)
  (* ****************************************************** *)

TYPE trimming_select = SELECT
    (cartesian_point,
    parameter_value);
```

```
END_TYPE;

TYPE trimming_preference = ENUMERATION OF
   (cartesian, parameter, unspecified);
END_TYPE;

TYPE transition_code = ENUMERATION OF
        (discontinuous,
        continuous,
        cont_same_gradient,
        cont_same_gradient_same_curvature);
END_TYPE;

ENTITY trimmed_curve    (* c0 *)
        SUBTYPE OF (bounded_curve);
        basis_curve           : curve;
        trim_1                : SET[1:2] OF trimming_select;
        trim_2                : SET[1:2] OF trimming_select;
        sense_agreement       : BOOLEAN;
        master_representation : trimming_preference;
 WHERE
        WR1: ('MACHINING_SCHEMA.PARAMETER_VALUE' IN TYPEOF(trim_1[1])) OR
             ('MACHINING_SCHEMA.PARAMETER_VALUE' IN TYPEOF(trim_1[2]));
        WR2: ('MACHINING_SCHEMA.PARAMETER_VALUE' IN TYPEOF(trim_2[1])) OR
             ('MACHINING_SCHEMA.PARAMETER_VALUE' IN TYPEOF(trim_2[2]));
        WR3: (''MACHINING_SCHEMA.PARAMETER' IN
  TYPEOF(SELF.master_representation));
END_ENTITY;

ENTITY composite_curve  (* c0 *)
        SUBTYPE OF (bounded_curve);
        segments       : LIST [1:?] OF composite_curve_segment;
        self_intersect : LOGICAL;
 DERIVE
        n_segments     : INTEGER := SIZEOF(segments);
        closed_curve   : LOGICAL
                       := segments[n_segments].transition <> discontinuous;
 WHERE
        WR1: ((NOT closed_curve) AND (SIZEOF(QUERY(temp <* segments |
                     temp.transition = discontinuous)) = 1)) OR
                 ((closed_curve) AND (SIZEOF(QUERY(temp <* segments |
                     temp.transition = discontinuous)) = 0));
END_ENTITY;

ENTITY composite_curve_segment;         (* c0 *)
        transition    : transition_code;
        same_sense    : BOOLEAN;
        parent_curve  : curve;
 INVERSE
         using_curves  : BAG [1:?] OF composite_curve FOR segments;
 WHERE
        WR1 : ('MACHINING_SCHEMA.BOUNDED_CURVE' IN
          TYPEOF(parent_curve));
END_ENTITY;

TYPE b_spline_curve_form = ENUMERATION OF (polyline_form, circular_arc,
  elliptic_arc, parabolic_arc, hyperbolic_arc, unspecified);
END_TYPE;
```

```
TYPE b_spline_surface_form = ENUMERATION OF (plane_surf, cylindrical_surf,
  conical_surf, spherical_surf, toroidal_surf, surf_of_revolution, ruled_surf,
  generalised_cone, quadric_surf, surf_of_linear_extrusion, unspecified);
END_TYPE;

TYPE dimension_count = INTEGER;
  WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE knot_type = ENUMERATION OF (uniform_knots, quasi_uniform_knots,
  piecewise_bezier_knots, unspecified);
END_TYPE;

TYPE pcurve_or_surface = SELECT (pcurve, surface);
END_TYPE;

TYPE vector_or_direction = SELECT (vector, direction);
END_TYPE;

ENTITY axis1_placement                                                  (* s1 *)
  SUBTYPE OF (placement);
  axis     : OPTIONAL direction;
  DERIVE
  z : direction := NVL(normalise(axis), dummy_gri ||
                     direction([0.0,0.0,1.0]));
  WHERE
  WR1: SELF\geometric_representation_item.dim  = 3;
END_ENTITY;

ENTITY axis2_placement_3d                                               (* m0 *)
  SUBTYPE OF (placement);
  axis          : OPTIONAL direction;
  ref_direction : OPTIONAL direction;
  DERIVE
  p             : LIST [3:3] OF direction := build_axes(axis, ref_direction);
  WHERE
  WR1: SELF\placement.location.dim = 3;
  WR2: (NOT (EXISTS (axis))) OR (axis.dim = 3);
  WR3: (NOT (EXISTS (ref_direction))) OR (ref_direction.dim = 3);
  WR4: (NOT (EXISTS (axis))) OR (NOT (EXISTS (ref_direction))) OR
       (cross_product(axis,ref_direction).magnitude > 0.0);
END_ENTITY;

ENTITY bezier_curve                                                     (* c1 *)
  SUBTYPE OF (b_spline_curve);
END_ENTITY;

ENTITY bezier_surface                                                   (* s1 *)
  SUBTYPE OF (b_spline_surface);
END_ENTITY;

ENTITY bounded_curve                                                    (* c0 *)
  SUPERTYPE OF (ONEOF(polyline, b_spline_curve, trimmed_curve,
                    bounded_pcurve, composite_curve))
  SUBTYPE OF (curve);
END_ENTITY;

ENTITY bounded_pcurve                                                   (* s0 *)
  SUBTYPE OF (pcurve, bounded_curve);
  WHERE
  WR1: ('MACHINING_SCHEMA.BOUNDED_CURVE' IN
        TYPEOF(SELF\pcurve.reference_to_curve.items[1]));
END_ENTITY;

ENTITY bounded_surface                                             (* s0 *)
  SUPERTYPE OF (b_spline_surface)
  SUBTYPE OF (surface);
END_ENTITY;

ENTITY b_spline_curve                                              (* c0 *)
  SUPERTYPE OF (ONEOF(uniform_curve, b_spline_curve_with_knots,
                  quasi_uniform_curve, bezier_curve) ANDOR
    rational_b_spline_curve)
  SUBTYPE OF (bounded_curve);
  degree:                      INTEGER;
  control_points_list:         LIST [2:?] OF cartesian_point;
  curve_form:                  b_spline_curve_form;
  closed_curve:                LOGICAL;
  self_intersect:              LOGICAL;
  DERIVE
  upper_index_on_control_points  : INTEGER
                                 := (SIZEOF(control_points_list) - 1);
  control_points       : ARRAY [0:upper_index_on_control_points]
                         OF cartesian_point
                       := list_to_array(control_points_list,0,
                          upper_index_on_control_points);
  WHERE
  WR1: (MACHINING_SCHEMA.UNIFORM_CURVE' IN TYPEOF(self)) OR
       ('MACHINING_SCHEMA.QUASI_UNIFORM_CURVE' IN TYPEOF(self)) OR
       ('MACHINING_SCHEMA.BEZIER_CURVE' IN TYPEOF(self)) OR
       ('MACHINING_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS' IN TYPEOF(self));
END_ENTITY;

ENTITY b_spline_curve_with_knots                                   (* c0 *)
  SUBTYPE OF (b_spline_curve);
  knot_multiplicities   : LIST [2:?] OF INTEGER;
  knots                 : LIST [2:?] OF parameter_value;
  knot_spec             : knot_type;
  DERIVE
  upper_index_on_knots : INTEGER := SIZEOF(knots);
  WHERE
  WR1: constraints_param_b_spline(degree, upper_index_on_knots,
                                  upper_index_on_control_points,
                                  knot_multiplicities, knots);
  WR2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY;

ENTITY b_spline_surface                                            (* s0 *)
  SUPERTYPE OF (ONEOF(b_spline_surface_with_knots, uniform_surface,
                   quasi_uniform_surface, bezier_surface) ANDOR
    rational_b_spline_surface)
  SUBTYPE OF (bounded_surface);
  u_degree              : INTEGER;
  v_degree              : INTEGER;
  control_points_list   : LIST [2:?] OF
                          LIST [2:?] OF cartesian_point;
  surface_form          : b_spline_surface_form;
  u_closed              : LOGICAL;
  v_closed              : LOGICAL;
  self_intersect        : LOGICAL;
  DERIVE
  u_upper              : INTEGER := SIZEOF(control_points_list) - 1;
  v_upper              : INTEGER := SIZEOF(control_points_list[1]) - 1;
  control_points       : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
                           cartesian_point
                       := make_array_of_array(control_points_list,
                                                0,u_upper,0,v_upper);
  WHERE
  WR1: ('MACHINING_SCHEMA.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
       ('MACHINING_SCHEMA.QUASI_UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
       ('MACHINING_SCHEMA.BEZIER_SURFACE' IN TYPEOF(SELF)) OR
       ('MACHINING_SCHEMA.B_SPLINE_SURFACE_WITH_KNOTS' IN TYPEOF(SELF));
END_ENTITY;

ENTITY b_spline_surface_with_knots                                    (* s0 *)
  SUBTYPE OF (b_spline_surface);
  u_multiplicities  : LIST [2:?] OF INTEGER;
  v_multiplicities  : LIST [2:?] OF INTEGER;
  u_knots           : LIST [2:?] OF parameter_value;
  v_knots           : LIST [2:?] OF parameter_value;
  knot_spec         : knot_type;
  DERIVE
  knot_u_upper      : INTEGER := SIZEOF(u_knots);
  knot_v_upper      : INTEGER := SIZEOF(v_knots);
  WHERE
  WR1: constraints_param_b_spline(SELF\b_spline_surface.u_degree,
                  knot_u_upper, SELF\b_spline_surface.u_upper,
                            u_multiplicities, u_knots);
  WR2: constraints_param_b_spline(SELF\b_spline_surface.v_degree,
                  knot_v_upper, SELF\b_spline_surface.v_upper,
                            v_multiplicities, v_knots);
  WR3: SIZEOF(u_multiplicities) = knot_u_upper;
    WR4: SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY;

ENTITY cartesian_point                                          (* m0 c0 s0 *)
  SUBTYPE OF (point);
  coordinates  : LIST [1:3] OF length_measure;
END_ENTITY;

ENTITY circle                                                         (* c1 *)
  SUBTYPE OF (conic);
  radius   : positive_length_measure;
END_ENTITY;

ENTITY conic                                                          (* c1 *)
  SUPERTYPE OF (ONEOF(circle, ellipse, hyperbola, parabola))
  SUBTYPE OF (curve);
  position: axis2_placement_3d;
END_ENTITY;

ENTITY curve                                                          (* c0 *)
  SUPERTYPE OF (ONEOF(line, conic, pcurve, bounded_curve))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY direction                                                      (* m0 *)
  SUBTYPE OF (geometric_representation_item);
  direction_ratios : LIST [2:3] OF REAL;
  WHERE
  WR1: SIZEOF(QUERY(tmp <* direction_ratios | tmp <> 0.0)) > 0;
```

```
END_ENTITY;

ENTITY elementary_surface                                                  (* m0 s0 *)
  SUPERTYPE OF (ONEOF(plane, spherical_surface))
  SUBTYPE OF (surface);
  position : axis2_placement_3d;
END_ENTITY;

ENTITY ellipse                                                                (* c1 *)
  SUBTYPE OF (conic);
  semi_axis_1 : positive_length_measure;
  semi_axis_2 : positive_length_measure;
END_ENTITY;

ENTITY geometric_representation_context                                       (* m0 *)
  SUBTYPE OF (representation_context);
  coordinate_space_dimension : dimension_count;
END_ENTITY;

ENTITY geometric_representation_item                                          (* m0 *)
  SUPERTYPE OF (ONEOF(point, direction, placement, curve, surface, edge_curve,
    face_surface, vertex_point, solid_model, poly_loop))
  SUBTYPE OF (representation_item);
  DERIVE
      dim : dimension_count := dimension_of(SELF);
  WHERE
  WR1: SIZEOF (QUERY (using_rep <* using_representations (SELF) |
       NOT ('MACHINING_SCHEMA.GEOMETRIC_REPRESENTATION_CONTEXT' IN
       TYPEOF (using_rep.context_of_items)))) = 0;
END_ENTITY;

ENTITY hyperbola                                                              (* c1 *)
  SUBTYPE OF (conic);
  semi_axis      : positive_length_measure;
  semi_imag_axis : positive_length_measure;
END_ENTITY;

ENTITY line                                                                   (* c1 *)
  SUBTYPE OF (curve);
  pnt : cartesian_point;
  dir : vector;
  WHERE
  WR1: dir.dim  = pnt.dim;
END_ENTITY;

ENTITY oriented_surface                                                       (* c1 *)
  SUBTYPE OF (surface);
  orientation : BOOLEAN;
END_ENTITY;

ENTITY parabola                                                               (* c1 *)
  SUBTYPE OF (conic);
  focal_dist : length_measure;
  WHERE
  WR1: focal_dist <> 0.0;
END_ENTITY;

ENTITY pcurve                                                                 (* s0 *)
  SUBTYPE OF (curve);
  basis_surface      : surface;
  reference_to_curve : definitional_representation;
  WHERE
  WR1: SIZEOF(reference_to_curve\representation.items) = 1;
  WR2: 'MACHINING_SCHEMA.CURVE' IN TYPEOF
                 (reference_to_curve\representation.items[1]);
  WR3: reference_to_curve\representation.items[1]\
                        geometric_representation_item.dim =2;
END_ENTITY;

ENTITY placement                                                   (* m0 *)
  SUPERTYPE OF (axis2_placement_3d)
  SUBTYPE OF (geometric_representation_item);
  location : cartesian_point;
END_ENTITY;

ENTITY plane                                                    (* m0 s0 *)
  SUBTYPE OF (elementary_surface);
END_ENTITY;

ENTITY point                                                 (* m0 c0 s0 *)
  SUPERTYPE OF (cartesian_point)
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY polyline                                                    (* c0 *)
  SUBTYPE OF (bounded_curve);
  points : LIST [2:?] OF cartesian_point;
END_ENTITY;

ENTITY quasi_uniform_curve                                         (* c1 *)
  SUBTYPE OF (b_spline_curve);
END_ENTITY;

ENTITY quasi_uniform_surface                                       (* s1 *)
  SUBTYPE OF (b_spline_surface);
END_ENTITY;

ENTITY rational_b_spline_curve                                     (* c0 *)
  SUBTYPE OF (b_spline_curve);
  weights_data : LIST [2:?] OF REAL;
  DERIVE
  weights : ARRAY [0:upper_index_on_control_points] OF REAL
          := list_to_array(weights_data,0,upper_index_on_control_points);
  WHERE
  WR1:  SIZEOF(weights_data) = SIZEOF(SELF\b_spline_curve.control_points_list);
  WR2:  curve_weights_positive(SELF);
END_ENTITY;

ENTITY rational_b_spline_surface                                   (* s0 *)
  SUBTYPE OF (b_spline_surface);
  weights_data : LIST [2:?] OF LIST [2:?] OF REAL;
  DERIVE
  weights       : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF REAL
                := make_array_of_array(weights_data,0,u_upper,0,v_upper);
  WHERE
  WR1: (SIZEOF(weights_data) = SIZEOF(SELF\b_spline_surface.control_points_list))
    AND (SIZEOF(weights_data[1]) =
    SIZEOF(SELF\b_spline_surface.control_points_list[1]));
  WR2: surface_weights_positive(SELF);
END_ENTITY;

ENTITY spherical_surface                                           (* s1 *)
```

```
  SUBTYPE OF (elementary_surface);
  radius   : positive_length_measure;
END_ENTITY;

ENTITY surface                                                           (* m0 s0 *)
  SUPERTYPE OF (ONEOF(swept_surface, elementary_surface, bounded_surface,
    oriented_surface))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY surface_of_linear_extrusion                                          (* s1 *)
  SUBTYPE OF (swept_surface);
  extrusion_axis      : vector;
END_ENTITY;

ENTITY surface_of_revolution                                                (* s1 *)
  SUBTYPE OF (swept_surface);
  axis_position       : axis1_placement;
  DERIVE
  axis_line : line := dummy_gri || curve() || line (axis_position.location,
                      dummy_gri || vector(axis_position.z, 1.0));
END_ENTITY;

ENTITY swept_surface                                                        (* s1 *)
  SUPERTYPE OF (ONEOF(surface_of_linear_extrusion, surface_of_revolution))
  SUBTYPE OF (surface);
  swept_curve : curve;
END_ENTITY;

ENTITY uniform_curve                                                        (* c1 *)
  SUBTYPE OF (b_spline_curve);
END_ENTITY;

ENTITY uniform_surface                                                      (* s1 *)
  SUBTYPE OF (b_spline_surface);
END_ENTITY;

ENTITY vector                                                            (* c1 s1 *)
  SUBTYPE OF (geometric_representation_item);
  orientation : direction;
  magnitude   : length_measure;
  WHERE
  WR1 : magnitude >= 0.0;
END_ENTITY;

RULE compatible_dimension FOR (cartesian_point, direction, representation_context,
  geometric_representation_context);
WHERE
    -- ensure that the count of coordinates of each cartesian_point
    -- matches the coordinate_space_dimension of each geometric_context in
    -- which it is geometrically founded
  WR1: SIZEOF(QUERY(x <* cartesian_point| SIZEOF(QUERY
          (y <* geometric_representation_context | item_in_context(x,y) AND
          (HIINDEX(x.coordinates) <> y.coordinate_space_dimension))) > 0 )) =0;
    -- ensure that the count of direction_ratios of each direction
    -- matches the coordinate_space_dimension of each geometric_context in
    -- which it is geometrically founded
  WR2: SIZEOF(QUERY(x <* direction | SIZEOF( QUERY
          (y <* geometric_representation_context | item_in_context(x,y) AND
          (HIINDEX(x.direction_ratios) <> y.coordinate_space_dimension)))
         > 0 )) = 0;
```

```
END_RULE;

FUNCTION build_axes(axis, ref_direction : direction): LIST [3:3] OF direction;
  LOCAL
  d1, d2 : direction;
  END_LOCAL;
  d1 := NVL(normalise(axis), dummy_gri || direction([0.0,0.0,1.0]));
  d2 := first_proj_axis(d1, ref_direction);
  RETURN([d2, normalise(cross_product(d1,d2)).orientation, d1]);
END_FUNCTION;

FUNCTION constraints_param_b_spline(degree, up_knots, up_cp : INTEGER;
                                            knot_mult : LIST OF INTEGER;
  knots : LIST OF parameter_value) : BOOLEAN;
  LOCAL
  result  : BOOLEAN := TRUE;
  k, sum  : INTEGER;
  END_LOCAL;
  (* Find sum of knot multiplicities. *)
  sum := knot_mult[1];
  REPEAT i := 2 TO up_knots;
  sum := sum + knot_mult[i];
  END_REPEAT;
  (* Check limits holding for all B-spline parametrisations *)
  IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR
        (sum <> (degree + up_cp + 2)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  k := knot_mult[1];
  IF (k < 1) OR (k > degree + 1) THEN
    result := FALSE;
    RETURN(result);
  END_IF;
  REPEAT i := 2 TO up_knots;
    IF (knot_mult[i] < 1) OR (knots[i] <= knots[i-1]) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    k := knot_mult[i];
    IF (i < up_knots) AND (k > degree) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
    IF (i = up_knots) AND (k > degree + 1) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
END_FUNCTION;

FUNCTION cross_product (arg1, arg2 : direction) : vector;
  LOCAL
  mag    : REAL;
  res    : direction;
  v1,v2  : LIST[3:3] OF REAL;
  result : vector;
  END_LOCAL;
  IF ( NOT EXISTS (arg1) OR (arg1.dim = 2)) OR
    ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
    RETURN(?);
   ELSE
    BEGIN
      v1  := normalise(arg1).direction_ratios;
      v2  := normalise(arg2).direction_ratios;
      res := dummy_gri || direction([(v1[2]*v2[3] - v1[3]*v2[2]),
            (v1[3]*v2[1] - v1[1]*v2[3]), (v1[1]*v2[2] - v1[2]*v2[1])]);
      mag := 0.0;
      REPEAT i := 1 TO 3;
        mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
      END_REPEAT;
      IF (mag > 0.0) THEN
        result := dummy_gri || vector(res, SQRT(mag));
      ELSE
        result := dummy_gri || vector(arg1, 0.0);
      END_IF;
      RETURN(result);
    END;
   END_IF;
 END_FUNCTION;

 FUNCTION curve_weights_positive(b: rational_b_spline_curve) : BOOLEAN;
   LOCAL
        result : BOOLEAN := TRUE;
   END_LOCAL;
   REPEAT i := 0 TO b.upper_index_on_control_points;
     IF b.weights[i] <= 0.0  THEN
       result := FALSE;
       RETURN(result);
     END_IF;
   END_REPEAT;
   RETURN(result);
 END_FUNCTION;

 FUNCTION dimension_of(item : geometric_representation_item): dimension_count;
   LOCAL
   x : SET OF representation;
   y : representation_context;
   END_LOCAL;
   -- Find the set of representation in which the item is used.
     x := using_representations(item);
   -- Determines the dimension_count of the
   -- geometric_representation_context. Note that the
   -- RULE compatible_dimension ensures that the context_of_items
   -- is of type geometric_representation_context and has
   -- the same dimension_count for all values of x.
   -- The SET x is non-empty since this is required by WR1 of
   -- representation_item.
     y := x[1].context_of_items;
   RETURN (y\geometric_representation_context.coordinate_space_dimension);
 END_FUNCTION;

 FUNCTION dot_product(arg1, arg2 : direction) : REAL;
   LOCAL
   scalar : REAL;
   vec1, vec2: direction;
   ndim : INTEGER;
   END_LOCAL;
   IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
     scalar := ?;
     (* When function is called with invalid data an indeterminate result
    is returned *)
  ELSE
    IF (arg1.dim <> arg2.dim) THEN
      scalar := ?;
    (* When function is called with invalid data an indeterminate result
    is returned *)
    ELSE
      BEGIN
        vec1   := normalise(arg1);
        vec2   := normalise(arg2);
        ndim   := arg1.dim;
        scalar := 0.0;
        REPEAT  i := 1 TO ndim;
          scalar := scalar +
                      vec1.direction_ratios[i]*vec2.direction_ratios[i];
        END_REPEAT;
      END;
    END_IF;
  END_IF;
  RETURN (scalar);
END_FUNCTION;

FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
  LOCAL
  x_axis : direction;
  v      : direction;
  z      : direction;
  x_vec  : vector;
  END_LOCAL;
  IF (NOT EXISTS(z_axis)) THEN
    RETURN (?) ;
  ELSE
    z := normalise(z_axis);
    IF NOT EXISTS(arg) THEN
      IF (z.direction_ratios <> [1.0,0.0,0.0]) THEN
        v :=  dummy_gri || direction([1.0,0.0,0.0]);
      ELSE
        v := dummy_gri || direction([0.0,1.0,0.0]);
      END_IF;
    ELSE
      IF  (arg.dim <> 3) THEN
        RETURN (?) ;
      END_IF;
      IF ((cross_product(arg,z).magnitude) = 0.0) THEN
        RETURN (?);
      ELSE
        v := normalise(arg);
      END_IF;
    END_IF;
    x_vec := scalar_times_vector(dot_product(v, z), z);
    x_axis := vector_difference(v, x_vec).orientation;
    x_axis := normalise(x_axis);
  END_IF;
  RETURN(x_axis);
END_FUNCTION;

FUNCTION list_to_array(lis : LIST [0:?] OF GENERIC : T; low,u : INTEGER) : ARRAY
  OF GENERIC : T;
  LOCAL
  n   : INTEGER;
  res : ARRAY [low:u] OF GENERIC : T;
  END_LOCAL;
  n := SIZEOF(lis);
  IF (n <> (u-low +1)) THEN
    RETURN(?);
  ELSE
    res := [lis[1] : n];
    REPEAT i := 2 TO n;
      res[low+i-1] := lis[i];
    END_REPEAT;
    RETURN(res);
  END_IF;
END_FUNCTION;

FUNCTION make_array_of_array(lis : LIST[1:?] OF LIST [1:?] OF GENERIC : T; low1,
  u1, low2, u2 : INTEGER): ARRAY OF ARRAY OF GENERIC : T;
  LOCAL
  res   : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
  END_LOCAL;
  (* Check input dimensions for consistency *)
   IF (u1-low1+1) <> SIZEOF(lis) THEN
     RETURN (?);
   END_IF;
   IF (u2 - low2 + 1 ) <> SIZEOF(lis[1]) THEN
     RETURN (?) ;
   END_IF;
  (* Initialise res with values from lis[1] *)
   res := [list_to_array(lis[1], low2, u2) : (u1-low1 + 1)];
   REPEAT i := 2 TO HIINDEX(lis);
     IF (u2-low2+1) <> SIZEOF(lis[i]) THEN
       RETURN (?);
     END_IF;
     res[low1+i-1] := list_to_array(lis[i], low2, u2);
   END_REPEAT;
  RETURN (res);
END_FUNCTION;

FUNCTION normalise (arg : vector_or_direction) : vector_or_direction;
  LOCAL
  ndim   : INTEGER;
  v      : direction;
  result : vector_or_direction;
  vec    : vector;
  mag    : REAL;
  END_LOCAL;
  IF NOT EXISTS (arg) THEN
    result := ?;
    (* When function is called with invalid data a NULL result is returned *)
  ELSE
    ndim := arg.dim;
    IF 'MACHINING_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
      BEGIN
            v := dummy_gri || direction(arg.orientation.direction_ratios);
        IF arg.magnitude = 0.0 THEN
          RETURN(?);
        ELSE
         vec := dummy_gri || vector (v, 1.0);
        END_IF;
      END;
    ELSE
      v := dummy_gri || direction (arg.direction_ratios);
    END_IF;
    mag := 0.0;
    REPEAT  i := 1 TO ndim;
      mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
    END_REPEAT;
    IF mag > 0.0 THEN
      mag := SQRT(mag);
      REPEAT  i := 1 TO ndim;
        v.direction_ratios[i] := v.direction_ratios[i]/mag;
      END_REPEAT;
      IF 'MACHINING_SCHEMA.VECTOR' IN TYPEOF(arg) THEN
        vec.orientation := v;
        result := vec;
      ELSE
        result := v;
      END_IF;
    ELSE
      RETURN(?);
    END_IF;
  END_IF;
  RETURN (result);
END_FUNCTION;

FUNCTION scalar_times_vector (scalar : REAL; vec : vector_or_direction)
                                                 : vector;
  LOCAL
  v      : direction;
  mag    : REAL;
  result : vector;
  END_LOCAL;
  IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
    RETURN (?) ;
   ELSE
    IF 'MACHINING_SCHEMA.VECTOR' IN TYPEOF (vec) THEN
      v   := dummy_gri || direction(vec.orientation.direction_ratios);
      mag := scalar * vec.magnitude;
    ELSE
      v   := dummy_gri || direction(vec.direction_ratios);
      mag := scalar;
    END_IF;
    IF (mag < 0.0 ) THEN
      REPEAT i := 1 TO SIZEOF(v.direction_ratios);
        v.direction_ratios[i] := -v.direction_ratios[i];
      END_REPEAT;
      mag := -mag;
    END_IF;
    result := dummy_gri || vector(normalise(v), mag);
  END_IF;
  RETURN (result);
END_FUNCTION;

FUNCTION surface_weights_positive(b: rational_b_spline_surface) : BOOLEAN;
  LOCAL
  result        : BOOLEAN := TRUE;
  END_LOCAL;
  REPEAT i := 0 TO b.u_upper;
    REPEAT j := 0 TO b.v_upper;
      IF (b.weights[i][j] <= 0.0)  THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
    END_REPEAT;
    RETURN(result);
  END_FUNCTION;

  FUNCTION vector_difference(arg1, arg2 : vector_or_direction) : vector;
    LOCAL
    result          : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim            : INTEGER;
    END_LOCAL;
    IF ((NOT EXISTS (arg1)) OR (NOT EXISTS (arg2))) OR (arg1.dim <> arg2.dim)
        THEN
      RETURN (?) ;
     ELSE
      BEGIN
        IF 'MACHINING_SCHEMA.VECTOR' IN TYPEOF(arg1) THEN
          mag1 := arg1.magnitude;
          vec1 := arg1.orientation;
        ELSE
          mag1 := 1.0;
          vec1 := arg1;
        END_IF;
        IF 'MACHINING_SCHEMA.VECTOR' IN TYPEOF(arg2) THEN
          mag2 := arg2.magnitude;
          vec2 := arg2.orientation;
        ELSE
          mag2 := 1.0;
          vec2 := arg2;
        END_IF;
        vec1 := normalise (vec1);
        vec2 := normalise (vec2);
        ndim := SIZEOF(vec1.direction_ratios);
        mag := 0.0;
        res := dummy_gri || direction(vec1.direction_ratios);
        REPEAT i := 1 TO ndim;
          res.direction_ratios[i] := mag1*vec1.direction_ratios[i] +
                                     mag2*vec2.direction_ratios[i];
          mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
        END_REPEAT;
        IF (mag > 0.0 ) THEN
        result := dummy_gri || vector( res, SQRT(mag));
        ELSE
          result := dummy_gri || vector( vec1,  0.0);
        END_IF;
      END;
    END_IF;
    RETURN (result);
  END_FUNCTION;


    (* ********************************************************* *)
    (* Types from topology_schema                   ISO 10303-42 *)
    (* ********************************************************* *)

  TYPE list_of_reversible_topology_item = LIST [0:?] of reversible_topology_item;
  END_TYPE;

  TYPE set_of_reversible_topology_item = SET [0:?] of reversible_topology_item;
  END_TYPE;
```

```
TYPE reversible_topology = SELECT (reversible_topology_item,
     list_of_reversible_topology_item, set_of_reversible_topology_item);
END_TYPE;

TYPE reversible_topology_item = SELECT (edge, path, face,
      face_bound, closed_shell, open_shell);
END_TYPE;

TYPE shell = SELECT (open_shell, closed_shell);
END_TYPE;

ENTITY closed_shell                                                         (* t0 *)
  SUBTYPE OF (connected_face_set, oriented_closed_shell);
END_ENTITY;

ENTITY connected_face_set                                                   (* t0 *)
  SUPERTYPE OF (ONEOF(open_shell, closed_shell))
  SUBTYPE OF (topological_representation_item);
  cfs_faces : SET [1:?] OF face;
END_ENTITY;

ENTITY edge                                                                 (* t0 *)
  SUPERTYPE OF(ONEOF(edge_curve, oriented_edge))
  SUBTYPE OF (topological_representation_item);
  edge_start : vertex;
  edge_end   : vertex;
END_ENTITY;

ENTITY edge_curve                                                           (* t0 *)
  SUBTYPE OF(edge, geometric_representation_item);
  edge_geometry : curve;
  same_sense    : BOOLEAN;
END_ENTITY;

ENTITY edge_loop                                                            (* t0 *)
  SUBTYPE OF (loop, path);
DERIVE
  ne : INTEGER := SIZEOF(SELF\path.edge_list);
WHERE
  WR1: (SELF\path.edge_list[1].edge_start) :=:
       (SELF\path.edge_list[ne].edge_end);
END_ENTITY;

ENTITY face                                                                 (* t0 *)
  SUPERTYPE OF(face_surface, oriented_face)
  SUBTYPE OF (topological_representation_item);
  bounds : SET[1:?] OF face_bound;
  WHERE
  WR1: NOT (mixed_loop_type_set(list_to_set(list_face_loops(SELF))));
  WR2: SIZEOF(QUERY(temp <* bounds | 'MACHINING_SCHEMA.FACE_OUTER_BOUND' IN
                                             TYPEOF(temp))) <= 1;
END_ENTITY;

ENTITY face_bound                                                           (* t1 *)
  SUPERTYPE OF(face_outer_bound)
  SUBTYPE OF(topological_representation_item);
  bound       :  loop;
  orientation :  BOOLEAN;
END_ENTITY;
```

```
ENTITY face_outer_bound                                              (* t1 *)
  SUBTYPE OF (face_bound);
END_ENTITY;

ENTITY face_surface                                                  (* t0 *)
  SUPERTYPE OF(advanced_face)
  SUBTYPE OF(face, geometric_representation_item);
  face_geometry :  surface;
  same_sense    :  BOOLEAN;
WHERE
  WR1: NOT ('MACHINING_SCHEMA.ORIENTED_SURFACE' IN TYPEOF(face_geometry));
END_ENTITY;

ENTITY loop                                                          (* t0 *)
  SUPERTYPE OF (ONEOF(vertex_loop, edge_loop, poly_loop))
  SUBTYPE OF (topological_representation_item);
END_ENTITY;

ENTITY open_shell                                                    (* t0 *)
  SUPERTYPE OF (topological_region, oriented_open_shell)
  SUBTYPE OF (connected_face_set);
END_ENTITY;

ENTITY oriented_closed_shell                                         (* t1 *)
 SUBTYPE OF (closed_shell);
  closed_shell_element : closed_shell;
  orientation          : BOOLEAN;
DERIVE
  SELF\connected_face_set.cfs_faces : SET [1:?] OF face
                                 := conditional_reverse(SELF.orientation,
                                    SELF.closed_shell_element.cfs_faces);
WHERE
  WR1: NOT ('MACHINING_SCHEMA.ORIENTED_CLOSED_SHELL'
             IN TYPEOF (SELF.closed_shell_element));
END_ENTITY;

ENTITY oriented_edge                                                 (* t0 *)
  SUBTYPE OF (edge);
  edge_element : edge;
  orientation  : BOOLEAN;
DERIVE
  SELF\edge.edge_start : vertex := boolean_choose (SELF.orientation,
                                   SELF.edge_element.edge_start,
    SELF.edge_element.edge_end);
  SELF\edge.edge_end   : vertex := boolean_choose (SELF.orientation,
  SELF.edge_element.edge_end, SELF.edge_element.edge_start);
WHERE
  WR1: NOT ('MACHINING_SCHEMA.ORIENTED_EDGE' IN TYPEOF (SELF.edge_element));
END_ENTITY;

ENTITY oriented_face                                                 (* t1 *)
  SUBTYPE OF (face);
  face_element : face;
  orientation  : BOOLEAN;
 DERIVE
   SELF\face.bounds : SET[1:?] OF face_bound
          := conditional_reverse(SELF.orientation,SELF.face_element.bounds);
 WHERE
   WR1: NOT ('MACHINING_SCHEMA.ORIENTED_FACE' IN TYPEOF (SELF.face_element));
 END_ENTITY;
```

```
ENTITY oriented_open_shell                                                    (* t1 *)
  SUBTYPE OF (open_shell);
  open_shell_element : open_shell;
  orientation        : BOOLEAN;
DERIVE
  SELF\connected_face_set.cfs_faces : SET [1:?] OF face
                                    := conditional_reverse(SELF.orientation,
                                       SELF.open_shell_element.cfs_faces);
WHERE
  WR1: NOT ('MACHINING_SCHEMA.ORIENTED_OPEN_SHELL'
                IN TYPEOF (SELF.open_shell_element));
END_ENTITY;


ENTITY oriented_path                                                          (* t1 *)
  SUBTYPE OF (path);
  path_element : path;
  orientation  : BOOLEAN;
 DERIVE
  SELF\path.edge_list : LIST [1:?] OF UNIQUE oriented_edge
                      := conditional_reverse(SELF.orientation,
                                   SELF.path_element.edge_list);
 WHERE
  WR1: NOT ('MACHINING_SCHEMA.ORIENTED_PATH' IN TYPEOF (SELF.path_element));
END_ENTITY;

ENTITY path                                                                   (* t0 *)
  SUPERTYPE OF (edge_loop, oriented_path)
  SUBTYPE OF (topological_representation_item);
  edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
WHERE
  WR1: path_head_to_tail(SELF);
END_ENTITY;

ENTITY poly_loop                                                              (* t0 *)
  SUBTYPE OF (loop, geometric_representation_item);
  polygon : LIST [3:?] OF UNIQUE cartesian_point;
END_ENTITY;

ENTITY topological_representation_item                                        (* t0 *)
  SUPERTYPE OF (ONEOF(face_bound, connected_face_set, edge, vertex, face,
                face_bound, (loop ANDOR path)))
  SUBTYPE OF (representation_item);
END_ENTITY;

ENTITY vertex                                                                 (* t0 *)
  SUBTYPE OF (topological_representation_item);
END_ENTITY;

ENTITY vertex_loop                                                            (* t1 *)
  SUBTYPE OF (loop);
  loop_vertex : vertex;
END_ENTITY;

ENTITY vertex_point                                                           (* t0 *)
  SUBTYPE OF(vertex, geometric_representation_item);
  vertex_geometry : point;
END_ENTITY;

FUNCTION boolean_choose (b:boolean; choice1, choice2:generic:item):generic:item;
```

```
    IF b THEN
      RETURN (choice1);
    ELSE
      RETURN (choice2);
    END_IF;
  END_FUNCTION;

  FUNCTION closed_shell_reversed (a_shell : closed_shell) :
                                         oriented_closed_shell;
    LOCAL
      the_reverse : oriented_closed_shell;
    END_LOCAL;
     IF ('MACHINING_SCHEMA.ORIENTED_CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
        the_reverse := dummy_tri ||
                       connected_face_set (
                          a_shell\connected_face_set.cfs_faces) ||
                       closed_shell () || oriented_closed_shell(
                          a_shell\oriented_closed_shell.closed_shell_element,
                          NOT(a_shell\oriented_closed_shell.orientation));
     ELSE
        the_reverse := dummy_tri ||
                      connected_face_set (
                        a_shell\connected_face_set.cfs_faces) ||
                      closed_shell () || oriented_closed_shell (a_shell, FALSE);
     END_IF;
     RETURN (the_reverse);
  END_FUNCTION;

  FUNCTION conditional_reverse (p       : BOOLEAN;
                                an_item : reversible_topology)
                                        : reversible_topology;
     IF p THEN
       RETURN (an_item);
     ELSE
       RETURN (topology_reversed (an_item));
     END_IF;
  END_FUNCTION;

  FUNCTION face_bound_reversed (a_face_bound : face_bound) : face_bound;
     LOCAL
       the_reverse : face_bound ;
     END_LOCAL;
     IF ('MACHINING_SCHEMA.FACE_OUTER_BOUND' IN TYPEOF (a_face_bound) ) THEN
       the_reverse := dummy_tri ||
                        face_bound(a_face_bound\face_bound.bound,
                             NOT (a_face_bound\face_bound.orientation))
                              || face_outer_bound() ;
     ELSE
       the_reverse := dummy_tri ||
                  face_bound(a_face_bound.bound, NOT(a_face_bound.orientation));
     END_IF;
    RETURN (the_reverse);
  END_FUNCTION;

  FUNCTION face_reversed (a_face : face) : oriented_face;
     LOCAL
       the_reverse : oriented_face ;
     END_LOCAL;
     IF ('MACHINING_SCHEMA.ORIENTED_FACE' IN TYPEOF (a_face) ) THEN
       the_reverse := dummy_tri ||
         face(set_of_topology_reversed(a_face.bounds)) ||
          oriented_face(a_face\oriented_face.face_element,
                           NOT (a_face\oriented_face.orientation)) ;
   ELSE
     the_reverse := dummy_tri ||
       face(set_of_topology_reversed(a_face.bounds)) ||
                                 oriented_face(a_face, FALSE) ;
   END_IF;
      RETURN (the_reverse);
END_FUNCTION;

FUNCTION list_face_loops(f: face) : LIST[0:?] OF loop;
  LOCAL
  loops : LIST[0:?] OF loop := [];
  END_LOCAL;
  REPEAT i := 1 TO SIZEOF(f.bounds);
    loops := loops +(f.bounds[i].bound);
  END_REPEAT;
  RETURN(loops);
END_FUNCTION;

FUNCTION edge_reversed (an_edge : edge) : oriented_edge;
   LOCAL
     the_reverse : oriented_edge;
   END_LOCAL;
   IF ('MACHINING_SCHEMA.ORIENTED_EDGE' IN TYPEOF (an_edge) ) THEN
     the_reverse  := dummy_tri ||
             edge(an_edge.edge_end, an_edge.edge_start) ||
             oriented_edge(an_edge\oriented_edge.edge_element,
                        NOT (an_edge\oriented_edge.orientation)) ;
   ELSE
     the_reverse := dummy_tri ||
             edge(an_edge.edge_end, an_edge.edge_start) ||
             oriented_edge(an_edge, FALSE);
   END_IF;
   RETURN (the_reverse);
END_FUNCTION;

FUNCTION list_of_topology_reversed (a_list
                                 : list_of_reversible_topology_item)
                                 : list_of_reversible_topology_item;
   LOCAL
     the_reverse : list_of_reversible_topology_item;
   END_LOCAL;
   the_reverse := [];
   REPEAT i := 1 TO SIZEOF (a_list);
     the_reverse := topology_reversed (a_list [i]) + the_reverse;
   END_REPEAT;
   RETURN (the_reverse);
END_FUNCTION;

FUNCTION list_to_set(l : LIST [0:?] OF GENERIC:T) : SET OF GENERIC:T;
LOCAL
  s : SET OF GENERIC:T := [];
END_LOCAL;
  REPEAT i := 1 TO SIZEOF(l);
    s := s + l[i];
  END_REPEAT;
  RETURN(s);
END_FUNCTION;

FUNCTION mixed_loop_type_set(l: SET[0:?] OF loop): LOGICAL;
```

```
LOCAL
  poly_loop_type: LOGICAL;
END_LOCAL;
  IF(SIZEOF(l) <= 1) THEN
    RETURN(FALSE);
  END_IF;
  poly_loop_type := ('MACHINING_SCHEMA.POLY_LOOP' IN TYPEOF(l[1]));
  REPEAT i := 2 TO SIZEOF(l);
    IF(('MACHINING_SCHEMA.POLY_LOOP' IN TYPEOF(l[i])) <> poly_loop_type) THEN
       RETURN(TRUE);
     END_IF;
  END_REPEAT;
  RETURN(FALSE);
END_FUNCTION;

FUNCTION open_shell_reversed ( a_shell : open_shell) :
                                         oriented_open_shell;
   LOCAL
     the_reverse : oriented_open_shell;
   END_LOCAL;
   IF ('MACHINING_SCHEMA.ORIENTED_OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
     the_reverse := dummy_tri ||
                    connected_face_set (
                        a_shell\connected_face_set.cfs_faces) ||
                    open_shell () || oriented_open_shell(
                      a_shell\oriented_open_shell.open_shell_element,
                        (NOT (a_shell\oriented_open_shell.orientation)));
   ELSE
     the_reverse := dummy_tri ||
                    connected_face_set (
                        a_shell\connected_face_set.cfs_faces) ||
                    open_shell () ||  oriented_open_shell (a_shell, FALSE);
   END_IF;
   RETURN (the_reverse);
END_FUNCTION;

FUNCTION path_head_to_tail(a_path : path) : BOOLEAN;
LOCAL
  n : INTEGER;
  p : BOOLEAN := TRUE;
END_LOCAL;
  n := SIZEOF (a_path.edge_list);
  REPEAT i := 2 TO n;
    p := p AND (a_path.edge_list[i-1].edge_end :=:
                a_path.edge_list[i].edge_start);
  END_REPEAT;
  RETURN (p);
END_FUNCTION;

FUNCTION path_reversed (a_path : path) : oriented_path;
  LOCAL
    the_reverse : oriented_path ;
  END_LOCAL;
  IF ('MACHINING_SCHEMA.ORIENTED_PATH' IN TYPEOF (a_path) ) THEN
    the_reverse := dummy_tri ||
       path(list_of_topology_reversed (a_path.edge_list)) ||
          oriented_path(a_path\oriented_path.path_element,
                       NOT(a_path\oriented_path.orientation)) ;
  ELSE
    the_reverse := dummy_tri ||
                   path(list_of_topology_reversed (a_path.edge_list)) ||
                       oriented_path(a_path, FALSE);
  END_IF;
  RETURN (the_reverse);
END_FUNCTION;

FUNCTION set_of_topology_reversed (a_set : set_of_reversible_topology_item)
                                          : set_of_reversible_topology_item;
   LOCAL
     the_reverse : set_of_reversible_topology_item;
   END_LOCAL;
   the_reverse := [];
   REPEAT i := 1 TO SIZEOF (a_set);
     the_reverse := the_reverse + topology_reversed (a_set [i]);
   END_REPEAT;
   RETURN (the_reverse);
END_FUNCTION;

FUNCTION shell_reversed (a_shell : shell) : shell;
   IF ('MACHINING_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
     RETURN (open_shell_reversed (a_shell));
   ELSE
     IF ('MACHINING_SCHEMA.CLOSED_SHELL' IN TYPEOF (a_shell) ) THEN
       RETURN (closed_shell_reversed (a_shell));
     ELSE
       RETURN (?);
     END_IF;
   END_IF;
END_FUNCTION;

FUNCTION topology_reversed (an_item : reversible_topology)
                                     : reversible_topology;
   IF ('MACHINING_SCHEMA.EDGE' IN TYPEOF (an_item)) THEN
     RETURN (edge_reversed (an_item));
   END_IF;
   IF ('MACHINING_SCHEMA.PATH' IN TYPEOF (an_item)) THEN
     RETURN (path_reversed (an_item));
   END_IF;
   IF ('MACHINING_SCHEMA.FACE_BOUND' IN TYPEOF (an_item)) THEN
     RETURN (face_bound_reversed (an_item));
   END_IF;
   IF ('MACHINING_SCHEMA.FACE' IN TYPEOF (an_item)) THEN
     RETURN (face_reversed (an_item));
   END_IF;
   IF ('MACHINING_SCHEMA.SHELL' IN TYPEOF (an_item)) THEN
     RETURN (shell_reversed (an_item));
   END_IF;
   IF ('SET' IN TYPEOF (an_item)) THEN
     RETURN (set_of_topology_reversed (an_item));
   END_IF;
   IF ('LIST' IN TYPEOF (an_item)) THEN
     RETURN (list_of_topology_reversed (an_item));
   END_IF;
   RETURN (?);
END_FUNCTION;


   (* ************************************************************ *)
   (* Types from geometric_model_schema              ISO 10303-42 *)
   (* ************************************************************ *)

ENTITY block                                                            (* m1 *)
```

```
   SUBTYPE OF (geometric_representation_item);
   position : axis2_placement_3d;
   x        : positive_length_measure;
   y        : positive_length_measure;
   z        : positive_length_measure;
END_ENTITY;

ENTITY faceted_brep                                                           (* t0 *)
  SUBTYPE OF (manifold_solid_brep);
END_ENTITY;

ENTITY manifold_solid_brep                                                    (* t0 *)
  SUPERTYPE OF (faceted_brep)
  SUBTYPE OF (solid_model);
  outer : closed_shell;
END_ENTITY;

ENTITY right_circular_cylinder                                                (* m1 *)
   SUBTYPE OF (geometric_representation_item);
   position   : axis1_placement;
   height     : positive_length_measure;
   radius     : positive_length_measure;
END_ENTITY;

ENTITY solid_model                                                            (* t0 *)
  SUPERTYPE OF (manifold_solid_brep)
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

   (* ************************************************************* *)
   (* Types from representation_schema                 ISO 10303-43 *)
   (* ************************************************************* *)

TYPE founded_item_select = SELECT
  (founded_item,
  representation_item);
END_TYPE;

ENTITY definitional_representation                                            (* s0 *)
  SUBTYPE OF (representation);
END_ENTITY;

ENTITY founded_item;                                                          (* t1 *)
END_ENTITY;

ENTITY mapped_item                                                            (* t1 *)
  SUBTYPE OF (representation_item);
  mapping_source : representation_map;
  mapping_target : representation_item;
  WHERE
  WR1: acyclic_mapped_representation(using_representations(SELF), [SELF]);
END_ENTITY;

ENTITY representation;
  name             : label;
  items            : SET[1:?] OF representation_item;
  context_of_items : representation_context;
  (* DERIVE
  id               : identifier := get_id_value (SELF);
  description      : text := get_description_value (SELF);
  WHERE
  WR1: SIZEOF (USEDIN (SELF, 'MACHINING_SCHEMA.' +
    'ID_ATTRIBUTE.IDENTIFIED_ITEM'))
         <= 1;
  WR2: SIZEOF (USEDIN (SELF, 'MACHINING_SCHEMA.' +
    'DESCRIPTION_ATTRIBUTE.DESCRIBED_ITEM')) <= 1; *)
END_ENTITY;

ENTITY representation_context;                                     (* t1 *)
  context_identifier : identifier;
  context_type       : text;
  INVERSE
  representations_in_context : SET [1:?] OF representation FOR context_of_items;
END_ENTITY;

ENTITY representation_item                                      (* c0 s0 *)
  SUPERTYPE OF (ONEOF(geometric_representation_item,
    topological_representation_item, mapped_item));
  name : label;
  WHERE
  WR1: SIZEOF(using_representations(SELF)) > 0;
END_ENTITY;

ENTITY representation_map;                                         (* t1 *)
  mapping_origin        : representation_item;
  mapped_representation : representation;
  INVERSE
  map_usage : SET[1:?] OF mapped_item FOR mapping_source;
  WHERE
  WR1: item_in_context(SELF.mapping_origin,
       SELF.mapped_representation.context_of_items);
END_ENTITY;

FUNCTION acyclic_mapped_representation
  (parent_set   : SET OF representation;
  children_set : SET OF representation_item) : BOOLEAN;
  LOCAL
    x,y : SET OF representation_item;
  END_LOCAL;
  -- Determine the subset of children_set that are mapped_items
    x := QUERY(z <* children_set | 'MACHINING_SCHEMA.MAPPED_ITEM'
      IN TYPEOF(z));
  -- Determine that the subset has elements
    IF SIZEOF(x) > 0 THEN
  -- Check each element of the set
    REPEAT i := 1 TO HIINDEX(x);
     -- If the selected element maps a representation in the
     -- parent_set, then return false
      IF x[i]\mapped_item.mapping_source.mapped_representation
        IN parent_set THEN
        RETURN (FALSE);
      END_IF;
     -- Recursive check of the items of mapped_representation
      IF NOT acyclic_mapped_representation
        (parent_set +
        x[i]\mapped_item.mapping_source.mapped_representation,
        x[i]\mapped_item.mapping_source.mapped_representation.items) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
```

```
    -- Determine the subset of children_set that are not
    -- mapped_items
    x := children_set - x;
    -- Determine that the subset has elements
    IF SIZEOF(x) > 0 THEN
      -- For each element of the set:
      REPEAT i := 1 TO HIINDEX(x);
        -- Determine the set of representation_items referenced
        y := QUERY(z <* bag_to_set( USEDIN(x[i], '')) |
             'MACHINING_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));
        -- Recursively check for an offending mapped_item
        -- Return false for any errors encountered
        IF NOT acyclic_mapped_representation(parent_set, y) THEN
          RETURN (FALSE);
        END_IF;
      END_REPEAT;
    END_IF;
    -- Return true when all elements are checked and
    -- no error conditions found
    RETURN (TRUE);
  END_FUNCTION;

  FUNCTION item_in_context
    (item  : representation_item;
     cntxt : representation_context) : BOOLEAN;
    LOCAL
      y : BAG OF representation_item;
    END_LOCAL;
    -- If there is one or more representation using both the item
    -- and cntxt return true.
    IF SIZEOF(USEDIN(item,'MACHINING_SCHEMA.REPRESENTATION.ITEMS')
      * cntxt.representations_in_context) > 0 THEN
      RETURN (TRUE);
      -- Determine the bag of representation_items that reference
      -- item
      ELSE y := QUERY(z <* USEDIN (item , '') |
             'MACHINING_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));
        -- Ensure that the bag is not empty
        IF SIZEOF(y) > 0 THEN
        -- For each element in the bag
        REPEAT i := 1 TO HIINDEX(y);
          -- Check to see it is an item in the input cntxt.
          IF item_in_context(y[i], cntxt) THEN
            RETURN (TRUE);
          END_IF;
        END_REPEAT;
      END_IF;
    END_IF;
    -- Return false when all possible branches have been checked
    -- with no success.
    RETURN (FALSE);
  END_FUNCTION;

  FUNCTION using_items (item : founded_item_select;
                        checked_items: SET OF founded_item_select)
                      : SET OF founded_item_select;
    LOCAL
      new_check_items    : SET OF founded_item_select;
      result_items       : SET OF founded_item_select;
      next_items         : SET OF founded_item_select;
    END_LOCAL;
    result_items := [];
    new_check_items := checked_items + item;
    -- Find the set of representation_items or founded_items
    -- in which item is used directly.
    next_items := QUERY(z <* bag_to_set( USEDIN(item , '')) |
      ('MACHINING_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z)) OR
      ('MACHINING_SCHEMA.FOUNDED_ITEM'        IN TYPEOF(z)));
    -- If the set of next_items is not empty;
    IF SIZEOF(next_items) > 0 THEN
      -- For each element in the set, find the using_items recursively
      REPEAT i := 1 TO HIINDEX(next_items);
        -- Check for loop in data model, i.e. one of the next_items
        -- occurred earlier in the set of check_items;
        IF NOT(next_items[i] IN new_check_items) THEN
          result_items := result_items + next_items[i] +
                          using_items(next_items[i],new_check_items);
        END_IF;
      END_REPEAT;
    END_IF;
    -- return the set of representation_items or founded_items
    -- in which the input item is used directly and indirectly.
    RETURN (result_items);
  END_FUNCTION;

  FUNCTION using_representations (item : founded_item_select)
    : SET OF representation;
    LOCAL
      results            : SET OF representation;
      result_bag         : BAG OF representation;
      intermediate_items : SET OF founded_item_select;
    END_LOCAL;
    -- Find the representations in which the item is used and add to the
    -- results set.
    results := [];
    result_bag := USEDIN(item,'MACHINING_SCHEMA.REPRESENTATION.ITEMS');
    IF SIZEOF(result_bag) > 0 THEN
      REPEAT i := 1 TO HIINDEX(result_bag);
        results := results + result_bag[i];
      END_REPEAT;
    END_IF;
    -- Find all representation_items or founded_items
    -- by which item is referenced directly or indirectly.
    intermediate_items := using_items(item,[]);
    -- If the set of intermediate items is not empty;
    IF SIZEOF(intermediate_items) > 0 THEN
      -- For each element in the set, add the
      -- representations of that element.
      REPEAT i := 1 TO HIINDEX(intermediate_items);
        result_bag := USEDIN(intermediate_items[i],
                      'MACHINING_SCHEMA.REPRESENTATION.ITEMS');
        IF SIZEOF(result_bag) > 0 THEN
          REPEAT j := 1 TO HIINDEX(result_bag);
            results := results + result_bag[j];
          END_REPEAT;
        END_IF;
      END_REPEAT;
    END_IF;
    -- Return the set of representation in which the input item is
    -- used directly and indirectly (through intervening
    -- representation_items or founded items).
    RETURN (results);
```

```
END_FUNCTION;


   (* ******************************************************** *)
   (* Types from aic_topologically_bounded_surface   ISO 10303-511 *)
   (* ******************************************************** *)

ENTITY advanced_face                                            (* t0 *)
  SUBTYPE OF (face_surface);
  WHERE
  WR1 : SIZEOF (['MACHINING_SCHEMA.ELEMENTARY_SURFACE',
       'MACHINING_SCHEMA.B_SPLINE_SURFACE',
       'MACHINING_SCHEMA.SWEPT_SURFACE'] *
        TYPEOF(face_geometry)) = 1;
  WR2 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
       'MACHINING_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT('MACHINING_SCHEMA.EDGE_CURVE' IN
        TYPEOF(oe\oriented_edge.edge_element)))) = 0))) = 0;
  WR3 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
       'MACHINING_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
        NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT (SIZEOF (['MACHINING_SCHEMA.LINE',
       'MACHINING_SCHEMA.CONIC',
       'MACHINING_SCHEMA.POLYLINE',
  (*   'MACHINING_SCHEMA.SURFACE_CURVE',*)
       'MACHINING_SCHEMA.B_SPLINE_CURVE'] *
        TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
        )) = 0))) = 0;
  WR4 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
       'MACHINING_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
        NOT(SIZEOF(QUERY (oe <* elp_fbnds.bound\path.edge_list |
        NOT(((('MACHINING_SCHEMA.VERTEX_POINT' IN
        TYPEOF(oe\edge.edge_start)) AND
       ( 'MACHINING_SCHEMA.CARTESIAN_POINT' IN
        TYPEOF(oe\edge.edge_start\vertex_point.vertex_geometry)))) AND
       (('MACHINING_SCHEMA.VERTEX_POINT' IN
        TYPEOF(oe\edge.edge_end)) AND
       ( 'MACHINING_SCHEMA.CARTESIAN_POINT' IN
        TYPEOF(oe\edge.edge_end\vertex_point.vertex_geometry)))
        ))) = 0))) = 0;
  WR5 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
       'MACHINING_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
       'MACHINING_SCHEMA.ORIENTED_PATH' IN
        TYPEOF(elp_fbnds.bound))) = 0;
  WR6 : (NOT ('MACHINING_SCHEMA.SWEPT_SURFACE' IN
        TYPEOF(face_geometry))) OR
       (SIZEOF (['MACHINING_SCHEMA.LINE',
        'MACHINING_SCHEMA.CONIC',
        'MACHINING_SCHEMA.POLYLINE',
        'MACHINING_SCHEMA.B_SPLINE_CURVE'] *
         TYPEOF(face_geometry\swept_surface.swept_curve)) = 1);
  WR7 : SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* bounds |
        'MACHINING_SCHEMA.VERTEX_LOOP' IN TYPEOF(bnds.bound)) |
         NOT(('MACHINING_SCHEMA.VERTEX_POINT' IN
         TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.loop_vertex)) AND
        ('MACHINING_SCHEMA.CARTESIAN_POINT' IN
         TYPEOF(vlp_fbnds\face_bound.bound\vertex_loop.
         loop_vertex\vertex_point.vertex_geometry))
        ))) = 0;
  WR8 : SIZEOF (QUERY (bnd <* bounds |
         NOT (SIZEOF(['MACHINING_SCHEMA.EDGE_LOOP',
         'MACHINING_SCHEMA.VERTEX_LOOP'] * TYPEOF(bnd.bound))
         = 1))) = 0;
   (*
   WR9 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
        'MACHINING_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
        ('MACHINING_SCHEMA.SURFACE_CURVE' IN
         TYPEOF(oe\oriented_edge.edge_element\edge_curve.edge_geometry))
         AND (NOT (SIZEOF (QUERY (sc_ag <*
     oe.edge_element\edge_curve.edge_geometry\
         surface_curve.associated_geometry |
         NOT ('MACHINING_SCHEMA.PCURVE' IN
         TYPEOF(sc_ag)))) = 0)))) = 0))) = 0;
   *)
   WR10 : ((NOT ('MACHINING_SCHEMA.SWEPT_SURFACE' IN
          TYPEOF(face_geometry))) OR
          ((NOT ('MACHINING_SCHEMA.POLYLINE' IN
          TYPEOF(face_geometry\swept_surface.swept_curve))) OR
          (SIZEOF(face_geometry\swept_surface.swept_curve\polyline.points)
          >= 3))) AND
          (SIZEOF (QUERY (elp_fbnds <* QUERY (bnds <* bounds |
          'MACHINING_SCHEMA.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
          NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
          ('MACHINING_SCHEMA.POLYLINE' IN
          TYPEOF(oe\oriented_edge.edge_element\edge_curve.edge_geometry)) AND
          (NOT (SIZEOF (oe\oriented_edge.edge_element\
          edge_curve.edge_geometry\polyline.points) >= 3)))) = 0))) = 0);
END_ENTITY;


   (* ************************************************************ *)
   (* Types from aic_advanced_brep                  ISO 10303-514 *)
   (* ************************************************************ *)

ENTITY advanced_brep_shape_representation                           (* t1 *)
   SUBTYPE OF (shape_representation);
   WHERE
   WR1: SIZEOF(QUERY ( it <* SELF.items | (NOT (SIZEOF([
       'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP',
       'AIC_ADVANCED_BREP.FACETED_BREP',
       'AIC_ADVANCED_BREP.MAPPED_ITEM',
       'AIC_ADVANCED_BREP.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1)) )) = 0;
   WR2: SIZEOF(QUERY ( it <* SELF.items | (SIZEOF([
       'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP',
       'AIC_ADVANCED_BREP.MAPPED_ITEM'] * TYPEOF(it)) = 1) )) > 0;
   (*
   WR3: SIZEOF(QUERY ( msb <* QUERY ( it <* SELF.items |
       ('AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) |
       ( NOT (SIZEOF(QUERY ( csh <* msb_shells(msb) |
        (NOT (SIZEOF(QUERY ( fcs <* csh\
        connected_face_set.cfs_faces | (NOT (
       'AIC_ADVANCED_BREP.ADVANCED_FACE' IN TYPEOF(fcs))) )) = 0)) ))  = 0)) )) =
     0;
   *)
   WR4: SIZEOF(QUERY ( msb <* QUERY ( it <* items |
       ( 'AIC_ADVANCED_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) ) |
       (  'AIC_ADVANCED_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF(msb\
      manifold_solid_brep.outer)) )) = 0;
   (*
   WR5: SIZEOF(QUERY ( brv <* QUERY ( it <* items |
        ( 'AIC_ADVANCED_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) ) | (NOT
        (SIZEOF(QUERY ( csh <* brv\brep_with_voids.voids |
        ( csh\oriented_closed_shell.orientation)))  = 0)) )) = 0;
   *)
   (*
   WR6: SIZEOF(QUERY ( mi <* QUERY ( it <* items |
        ( 'AIC_ADVANCED_BREP.MAPPED_ITEM' IN TYPEOF(it)) ) | (NOT
        ( 'AIC_ADVANCED_BREP.ADVANCED_BREP_SHAPE_REPRESENTATION' IN
        TYPEOF(mi\mapped_item.mapping_source.mapped_representation))) )) = 0;
   *)
 END_ENTITY;


   (* ******************************************************** *)
   (* ******************************************************** *)
   (*                                                          *)
   (* General types and definitions                            *)
   (*                                                          *)
   (* ******************************************************** *)
   (* ******************************************************** *)


   (* ******************************************************** *)
   (* Measure units                                            *)
   (* ******************************************************** *)

 ENTITY toleranced_length_measure;                                  (* m1 *)
   theoretical_size:   positive_length_measure;
   implicit_tolerance: tolerance_select;
 END_ENTITY;

 TYPE tolerance_select = SELECT(plus_minus_value, limits_and_fits);
 END_TYPE;

 ENTITY plus_minus_value;                                           (* m1 *)
   upper_limit:           positive_length_measure;
   lower_limit:           positive_length_measure;
   significant_digits:    INTEGER;
 END_ENTITY;

 ENTITY limits_and_fits;                                            (* m1 *)
   deviation:             length_measure;
   grade:                 length_measure;
   its_fitting_type:      OPTIONAL fitting_type;
 END_ENTITY;

 TYPE fitting_type = ENUMERATION OF (shaft,hole);
 END_TYPE;

 TYPE speed_measure = REAL;
 END_TYPE;

 TYPE rot_speed_measure = REAL;
 END_TYPE;

 TYPE pressure_measure = REAL;
 END_TYPE;
```

```
  (* ************************************************************ *)
  (* Other general types                                          *)
  (* ************************************************************ *)

TYPE rot_direction = ENUMERATION OF (cw,ccw);
END_TYPE;

TYPE shape_tolerance = length_measure;
END_TYPE;

  (* ************************************************************ *)
  (* ************************************************************ *)
  (*                                                              *)
  (*  PROJECT                                                     *)
  (*                                                              *)
  (* ************************************************************ *)
(* ************************************************************ *)


ENTITY project;                                                             (* m0 *)
  its_id:          identifier;
  main_workplan:   workplan;
  its_workpieces:  SET [0:?] OF workpiece;
  its_owner:        OPTIONAL person_and_address;
  its_release:      OPTIONAL date_and_time;
  its_status:      OPTIONAL approval;
  (*
  Informal proposition:
  its_id shall be unique within the part programme.
  *)
END_ENTITY;

ENTITY person_and_address;                                                  (* m0 *)
  its_person:             person;
  its_address:            OPTIONAL address;
END_ENTITY;

  (* ************************************************************ *)
  (* ************************************************************ *)
  (*                                                              *)
  (* Workpiece and manufacturing feature                          *)
  (*                                                              *)
  (* ************************************************************ *)
  (* ************************************************************ *)


ENTITY workpiece;                                                           (* m1 *)
  its_id:                  identifier;
  its_material:            OPTIONAL material;
  global_tolerance:        OPTIONAL shape_tolerance;
  its_rawpiece:            OPTIONAL workpiece;
  its_geometry:            OPTIONAL advanced_brep_shape_representation;
  its_bounding_geometry:   OPTIONAL bounding_geometry_select;
  clamping_positions:       SET [0:?] OF cartesian_point;
END_ENTITY;
```

```
   (* ************************************************************ *)
   (* Material                                                     *)
   (* ************************************************************ *)

ENTITY material;                                                             (* m1 *)
  standard_identifier:        label;
  material_identifier:        label;
  material_property:          SET [0:?] OF property_parameter;
END_ENTITY;


   (* ************************************************************ *)
   (* Property parameter                                           *)
   (* ************************************************************ *)

ENTITY property_parameter                                                    (* m0 *)
  SUPERTYPE OF (ONEOF (descriptive_parameter,numeric_parameter));
  parameter_name:                 label;
END_ENTITY;

ENTITY descriptive_parameter                                                 (* m0 *)
  SUBTYPE OF (property_parameter);
  descriptive_string:             text;
END_ENTITY;

ENTITY numeric_parameter                                                     (* m0 *)
  SUBTYPE OF (Property_parameter);
  its_parameter_value:                parameter_value;
  its_parameter_unit:                label;
END_ENTITY;

   (* ************************************************************ *)
   (* Bounding geometry                                            *)
   (* ************************************************************ *)

TYPE bounding_geometry_select = SELECT (block, right_circular_cylinder,      (* m1 *)
                                 advanced_brep_shape_representation);        (* t1 *)
END_TYPE;

   (* ************************************************************ *)
   (* Manufacturing feature                                        *)
   (* ************************************************************ *)

ENTITY manufacturing_feature                                                 (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF(region, two5D_manufacturing_feature,
    transition_feature));
  its_id:              identifier;
  its_workpiece:       workpiece;
  its_operations:      SET [0:?] OF machining_operation;
END_ENTITY;
```

```
   (* ************************************************************ *)
   (* ************************************************************ *)
   (*                                                              *)
   (* Catalogue of manufacturing feature                           *)
   (*                                                              *)
   (* ************************************************************ *)
   (* ************************************************************ *)


   (* ************************************************************ *)
   (* Region                                                       *)
   (* ************************************************************ *)

ENTITY region                                                               (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF (region_surface_list, region_projection,
    topological_region))
  SUBTYPE OF (manufacturing_feature);
  feature_placement: OPTIONAL axis2_placement_3d;
END_ENTITY;

ENTITY region_projection                                                    (* m1 *)
  SUBTYPE OF (region);
  proj_curve : bounded_curve;
  proj_dir   : direction;
  depth      : toleranced_length_measure;
END_ENTITY;

ENTITY region_surface_list                                                  (* m1 *)
  SUBTYPE OF (region);
  surface_list : LIST [1:?] OF bounded_surface;
END_ENTITY;

ENTITY topological_region                                                (* t0 t1 *)
  SUBTYPE OF (region, open_shell);
WHERE
  WR1: SIZEOF(QUERY(it <* SELF.cfs_faces | NOT('MACHINING_SCHEMA.ADVANCED_FACE' IN
    TYPEOF(it)))) = 0;
END_ENTITY;


   (* ************************************************************ *)
   (* 2.5D manufacturing feature                                   *)
   (* ************************************************************ *)

ENTITY two5D_manufacturing_feature                                          (* m1 *)
ABSTRACT SUPERTYPE OF (ONEOF(machining_feature, replicate_feature,
  compound_feature))
SUBTYPE OF (manufacturing_feature);
  feature_placement:        axis2_placement_3d;
END_ENTITY;


   (* ************************************************************ *)
   (* Machining feature                                            *)
   (* ************************************************************ *)

ENTITY machining_feature                                                    (* m1 *)
ABSTRACT SUPERTYPE OF (ONEOF(planar_face, pocket, slot, step, round_hole,
  toolpath_feature, profile_feature, boss, spherical_cap, rounded_end, thread))
SUBTYPE OF (two5D_manufacturing_feature);
  depth:                      elementary_surface;
```

```
END_ENTITY;


   (* ************************************************************* *)
   (* Planer face                                                   *)
   (* ************************************************************* *)

ENTITY planar_face                                                            (* m1 *)
  SUBTYPE OF (machining_feature);
  course_of_travel:       linear_path;
  removal_boundary:       linear_profile;
  face_boundary:          OPTIONAL closed_profile;
  its_boss:                    SET [0:?] OF boss;
(*
Informal propositions:
- The entire profile lies in the local xy plane.
- The profile is not self-intersecting.
*)
END_ENTITY;


   (* ************************************************************* *)
   (* Pocket                                                        *)
   (* ************************************************************* *)

ENTITY pocket                                                                 (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF(closed_pocket, open_pocket))
  SUBTYPE OF (machining_feature);
  its_boss:                    SET [0:?] OF boss;
  slope:                       OPTIONAL plane_angle_measure;
  bottom_condition:            pocket_bottom_condition;
  planar_radius:               OPTIONAL toleranced_length_measure;
  orthogonal_radius:           OPTIONAL toleranced_length_measure;
END_ENTITY;

ENTITY closed_pocket                                                          (* m1 *)
  SUBTYPE OF (pocket);
  feature_boundary:            closed_profile;
(*
Informal propositions:
- The entire profile lies in the feature's local xy plane.
- The profile is closed and not self-intersecting.
*)
END_ENTITY;

ENTITY open_pocket                                                            (* m1 *)
  SUBTYPE OF (pocket);
  open_boundary:               open_profile;
  wall_boundary:               OPTIONAL open_profile;
(*
Informal propositions:
- The entire open_boundary profile lies in the local xy plane.
- The profile are is not self-intersecting.
- Together the two open_profiles form a closed profile.
- wall_boundary is for information only.
*)
END_ENTITY;

ENTITY pocket_bottom_condition                                                (* m1 *)
  ABSTRACT SUPERTYPE OF
```

```
   (ONEOF (through_pocket_bottom_condition, planar_pocket_bottom_condition,
     radiused_pocket_bottom_condition, general_pocket_bottom_condition));
END_ENTITY;

ENTITY through_pocket_bottom_condition                                (* m1 *)
  SUBTYPE OF (pocket_bottom_condition);
END_ENTITY;

ENTITY planar_pocket_bottom_condition                                 (* m1 *)
  SUBTYPE OF (pocket_bottom_condition);
END_ENTITY;

ENTITY radiused_pocket_bottom_condition                               (* m1 *)
  SUBTYPE OF (pocket_bottom_condition);
  floor_radius_center:       cartesian_point;
  floor_radius:              toleranced_length_measure;
END_ENTITY;

ENTITY general_pocket_bottom_condition                                (* m1 *)
  SUBTYPE OF (pocket_bottom_condition);
  shape:                     region;
WHERE
  WR1: SIZEOF(shape\manufacturing_feature.its_operations) = 0;
END_ENTITY;


  (* ****************************************************** *)
  (* Slot                                                   *)
  (* ****************************************************** *)

ENTITY slot                                                           (* m1 *)
  SUBTYPE OF (machining_feature);
  course_of_travel:          travel_path;
  swept_shape:               open_profile;
  end_conditions:            LIST[0:2] OF slot_end_type;
WHERE
  WR1: ( ( SIZEOF(QUERY (it <* SELF.end_conditions |
          ('MACHINING_SCHEMA.LOOP_SLOT_END_TYPE' IN TYPEOF(it)) )) = 1)
        AND
        (SIZEOF(end_conditions) = 1) )
       OR
       (SIZEOF(end_conditions) <> 1);
(*
Informal propositions:
- The entire travel_path lies in the local xy plane.
- The travel_path is not self-intersecting.
*)
END_ENTITY;

ENTITY slot_end_type                                                  (* m1 *)
ABSTRACT SUPERTYPE OF (ONEOF (woodruff_slot_end_type, radiused_slot_end_type,
  flat_slot_end_type, loop_slot_end_type, open_slot_end_type));
END_ENTITY;

ENTITY woodruff_slot_end_type                                         (* m1 *)
  SUBTYPE OF (slot_end_type);
  radius:                    toleranced_length_measure;
END_ENTITY;

ENTITY radiused_slot_end_type                                         (* m1 *)
  SUBTYPE OF (slot_end_type);
```

```
END_ENTITY;

ENTITY flat_slot_end_type                                                   (* m1 *)
  SUBTYPE OF (slot_end_type);
  corner_radius1:           toleranced_length_measure;
  corner_radius2:           toleranced_length_measure;
END_ENTITY;

ENTITY loop_slot_end_type                                                   (* m1 *)
  SUBTYPE OF (slot_end_type);
END_ENTITY;

ENTITY open_slot_end_type                                                   (* m1 *)
  SUBTYPE OF (slot_end_type);
END_ENTITY;


   (* ************************************************************ *)
   (* Step                                                         *)
   (* ************************************************************ *)

ENTITY step                                                                 (* m1 *)
  SUBTYPE OF (machining_feature);
  open_boundary:          linear_path;
  wall_boundary:          OPTIONAL vee_profile;
  its_boss:               SET[0:?] OF boss;
(*
Informal propositions:
- The entire linear_path lies in the same plane.
*)
END_ENTITY;

   (* ************************************************************ *)
   (* Profile_feature                                              *)
   (* ************************************************************ *)
ENTITY profile_feature                                                      (* m1 *)
ABSTRACT SUPERTYPE OF(ONEOF(general_outside_profile, shape_profile))
  SUBTYPE OF (machining_feature);
     profile_swept_shape : linear_path;
END_ENTITY;

ENTITY general_outside_profile                                              (* m1 *)
  SUBTYPE OF (profile_feature);
  feature_boundary:       profile;
(*
Informal propositions:
- The entire profile lies in the local xy plane.
- The profile is not self-intersecting.
*)
END_ENTITY;

ENTITY shape_profile                                                        (* m1 *)
  ABSTRACT SUPERTYPE
    OF(ONEOF(general_shape_profile,partial_circular_shape_profile,
    circular_closed_shape_profile,rectangular_open_shape_profile,
    rectangular_closed_shape_profile))
  SUBTYPE OF (profile_feature);
  floor_condition:         profile_select;
  removal_direction:       direction;
END_ENTITY;
```

```
TYPE profile_select = SELECT (through_profile_floor, profile_floor);
END_TYPE;

ENTITY through_profile_floor;                                               (* m1 *)
END_ENTITY;

ENTITY profile_floor                                                        (* m1 *)
ABSTRACT SUPERTYPE OF(ONEOF(general_profile_floor, planar_profile_floor));
  floor_radius :             OPTIONAL numeric_parameter;
  start_or_end :             BOOLEAN;
END_ENTITY;

ENTITY general_profile_floor                                                (* m1 *)
SUBTYPE OF (profile_floor);
  floor :                 face;
END_ENTITY;

ENTITY planar_profile_floor                                                 (* m1 *)
SUBTYPE OF (profile_floor);
  floor :                plane;
END_ENTITY;

ENTITY general_shape_profile                                                (* m1 *)
SUBTYPE OF (shape_profile);
  profile_boundary :     profile;
END_ENTITY;

ENTITY partial_circular_shape_profile                                       (* m1 *)
SUBTYPE OF (shape_profile);
  open_boundary :        partial_circular_profile;
END_ENTITY;

ENTITY circular_closed_shape_profile                                        (* m1 *)
SUBTYPE OF (shape_profile);
  closed_boundary :      circular_closed_profile;
END_ENTITY;

ENTITY rectangular_open_shape_profile                                       (* m1 *)
SUBTYPE OF (shape_profile);
  open_boundary :        square_u_profile;
END_ENTITY;

ENTITY rectangular_closed_shape_profile                                     (* m1 *)
SUBTYPE OF (shape_profile);
  closed_boundary :      rectangular_closed_profile;
END_ENTITY;

  (* ************************************************************ *)
  (* Round hole                                                   *)
  (* ************************************************************ *)

ENTITY round_hole                                                           (* m1 *)
  SUBTYPE OF (machining_feature);
  diameter:               toleranced_length_measure;
  change_in_diameter:     OPTIONAL taper_select;
  bottom_condition:       hole_bottom_condition;
END_ENTITY;

TYPE taper_select = SELECT (diameter_taper, angle_taper);
END_TYPE;
```

```
ENTITY diameter_taper;                                                      (* m1 *)
  final_diameter: toleranced_length_measure;
END_ENTITY;

ENTITY angle_taper;                                                         (* m1 *)
  angle: plane_angle_measure;
END_ENTITY;

ENTITY hole_bottom_condition                                                (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF (blind_bottom_condition,
    through_bottom_condition));
END_ENTITY;

ENTITY through_bottom_condition                                             (* m1 *)
  SUBTYPE OF (hole_bottom_condition);
END_ENTITY;

ENTITY blind_bottom_condition                                               (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF(flat_hole_bottom, flat_with_radius_hole_bottom,
    spherical_hole_bottom, conical_hole_bottom))
  SUBTYPE OF (hole_bottom_condition);
END_ENTITY;

ENTITY flat_hole_bottom                                                     (* m1 *)
  SUBTYPE OF (blind_bottom_condition);
END_ENTITY;

ENTITY flat_with_radius_hole_bottom                                         (* m1 *)
  SUBTYPE OF (blind_bottom_condition);
  corner_radius: toleranced_length_measure;
END_ENTITY;

ENTITY spherical_hole_bottom                                                (* m1 *)
  SUBTYPE OF (blind_bottom_condition);
  radius: toleranced_length_measure;
END_ENTITY;

ENTITY conical_hole_bottom                                                  (* m1 *)
  SUBTYPE OF (blind_bottom_condition);
  tip_angle:      plane_angle_measure;
  tip_radius:     OPTIONAL toleranced_length_measure;
END_ENTITY;


   (* ****************************************************** *)
   (* Toolpath feature                                      *)
   (* ****************************************************** *)

ENTITY toolpath_feature                                                     (* m1 *)
  SUBTYPE OF (machining_feature);
END_ENTITY;

   (* ****************************************************** *)
   (* Boss                                                   *)
   (* ****************************************************** *)

ENTITY boss                                                                 (* m1 *)
SUBTYPE OF(machining_feature);
  its_boundary:              closed_profile;
  slope:                     OPTIONAL plane_angle_measure;
(*
Informal propositions:
- The entire profile lies in the same plane.
- The profile is not self-intersecting.
*)
END_ENTITY;

   (* ******************************************************* *)
   (* Spherical_cap                                           *)
   (* ******************************************************* *)

ENTITY spherical_cap                                              (* m1 *)
SUBTYPE OF (machining_feature);
  internal_angle:        numeric_parameter;
  radius:                numeric_parameter;
END_ENTITY;

   (* ******************************************************* *)
   (* Rounded_end                                             *)
   (* ******************************************************* *)

ENTITY rounded_end                                                (* m1 *)
SUBTYPE OF (machining_feature);
  course_of_travel:          linear_path;
  partial_circular_boundary: partial_circular_profile;
END_ENTITY;

   (* ******************************************************* *)
   (* Compound feature                                        *)
   (* ******************************************************* *)

ENTITY compound_feature                                           (* m1 *)
SUPERTYPE OF (ONEOF(counterbore_hole, countersunk_hole))
SUBTYPE OF (two5D_manufacturing_feature);
  elements: SET [2:?] OF compound_feature_select;
WHERE
  WR1: SIZEOF(QUERY(e <* elements | SIZEOF(e\manufacturing_feature.its_operations)
    <> 0)) = 0;
END_ENTITY;

TYPE compound_feature_select = SELECT(
  machining_feature, transition_feature
);
END_TYPE;

ENTITY counterbore_hole                                           (* m1 *)
  SUBTYPE OF (compound_feature);
WHERE
  WR1: SIZEOF(elements) =2;
  WR2: (SIZEOF(QUERY ( it <* SELF.elements |
       (('MACHINING_SCHEMA.ROUND_HOLE' IN TYPEOF(it))) )) = 2);
  WR3: SELF.elements[1].diameter.theoretical_size <>
       SELF.elements[2].diameter.theoretical_size;
END_ENTITY;

ENTITY countersunk_hole                                           (* m1 *)
 SUBTYPE OF (compound_feature);
WHERE
  WR1: SIZEOF(elements) =2;
  WR2: (SIZEOF(QUERY ( it <* SELF.elements |
       (('MACHINING_SCHEMA.ROUND_HOLE' IN TYPEOF(it))) )) = 2);
    WR3: SELF.elements[1].diameter.theoretical_size <>
         SELF.elements[2].diameter.theoretical_size;
    WR4: NOT EXISTS(SELF.elements[1].change_in_diameter) AND
         EXISTS(SELF.elements[2].change_in_diameter);
  END_ENTITY;


   (* ********************************************************* *)
   (* Replicate feature                                         *)
   (* ********************************************************* *)

  ENTITY replicate_feature                                                 (* m1 *)
    ABSTRACT SUPERTYPE OF (ONEOF(rectangular_pattern, circular_pattern,
      general_pattern))
    SUBTYPE OF (two5D_manufacturing_feature);
    replicate_base_feature:     two5D_manufacturing_feature;
  END_ENTITY;

  ENTITY circular_pattern                                                  (* m1 *)
    SUBTYPE OF(replicate_feature);
    angle_increment:            plane_angle_measure;
    number_of_feature:          INTEGER;
    relocated_base_feature:     SET[0:?] OF circular_offset;
    missing_base_feature:       SET[0:?] OF circular_omit;
    base_feature_diameter:      OPTIONAL toleranced_length_measure;
    base_feature_rotation :     plane_angle_measure;
  END_ENTITY;

  ENTITY circular_offset;                                                  (* m1 *)
    angular_offset:             plane_angle_measure;
    index:                      INTEGER;
  END_ENTITY;

  ENTITY circular_omit;                                                    (* m1 *)
    index:                      INTEGER;
  END_ENTITY;

  ENTITY rectangular_pattern                                               (* m1 *)
    SUBTYPE OF(replicate_feature);
    spacing :                   toleranced_length_measure;
    its_direction :             direction;
    number_of_rows:             OPTIONAL INTEGER;
    number_of_columns:          INTEGER;
    row_spacing:                OPTIONAL toleranced_length_measure;
    row_layout_direction:       OPTIONAL direction;
    relocated_base_feature:     SET[0:?] OF rectangular_offset;
    missing_base_feature:       SET[0:?] OF rectangular_omit;
    WHERE
     WR1: (
           (SELF.number_of_rows > 1 )
           AND EXISTS(SELF.row_spacing)
           AND EXISTS(SELF.row_layout_direction)
           );
  END_ENTITY;

  ENTITY rectangular_offset;                                               (* m1 *)
    offset_direction:   direction;
    offset_distance:    length_measure;
    row_index:          INTEGER;
    column_index:       INTEGER;
  END_ENTITY; -- rectangular_offset
```

```
ENTITY rectangular_omit;                                                    (* m1 *)
  row_index:          INTEGER;
  column_index:       INTEGER;
END_ENTITY; -- rectangular_omit

ENTITY general_pattern                                                      (* m1 *)
  SUBTYPE OF(replicate_feature);
  replicate_locations:  LIST [2:?] OF axis2_placement_3d;
END_ENTITY;


   (* ********************************************************** *)
   (* Transition feature                                          *)
   (* ********************************************************** *)

ENTITY transition_feature                                                   (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF(chamfer, edge_round))
  SUBTYPE OF (manufacturing_feature);
  first_feature:              machining_feature;
  second_feature:             machining_feature;
END_ENTITY;

ENTITY chamfer                                                              (* m1 *)
  SUBTYPE OF (transition_feature);
  angle_to_plane:             plane_angle_measure;
  first_offset_amount:        toleranced_length_measure;
END_ENTITY;

ENTITY edge_round                                                           (* m1 *)
  SUBTYPE OF (transition_feature);
  radius:                     toleranced_length_measure;
  first_offset_amount:        OPTIONAL toleranced_length_measure;
  second_offset_amount:       OPTIONAL toleranced_length_measure;
END_ENTITY;

   (* ********************************************************** *)
   (* Tread                                                       *)
   (* ********************************************************** *)

ENTITY thread                                                               (* m1 *)
ABSTRACT SUPERTYPE OF(ONEOF(catalogue_thread, defined_thread))
SUBTYPE OF(machining_feature);
  partial_profile:       partial_area_definition;
  applied_shape:         SET[1:?] OF machining_feature;
  inner_or_outer_thread: BOOLEAN;
  qualifier:             OPTIONAL descriptive_parameter;
  fit_class:             descriptive_parameter;
  form:                  descriptive_parameter;
  major_diameter:        length_measure;
  number_of_threads:     numeric_parameter;
  thread_hand:           descriptive_parameter;
 WHERE
  WR1: ('MACHINING_SCHEMA.ROUND_HOLE' IN TYPEOF(applied_shape)) OR
       ('MACHINING_SCHEMA.CIRCULAR_CLOSED_SHAPE_PROFILE' IN TYPEOF(applied_shape))
       OR ('MACHINING_SCHEMA.BOSS' IN TYPEOF(applied_shape));
END_ENTITY;

ENTITY partial_area_definition;                                             (* m1 *)
```

```
  effective_length:      length_measure;
  placement:             axis2_placement_3D;
  maximum_length:        OPTIONAL length_measure;
END_ENTITY;

ENTITY catalogue_thread                                                    (* m1 *)
SUBTYPE OF (thread);
  documentation:         specification;
END_ENTITY;

ENTITY specification;                                                      (* m1 *)
  constraint:                SET [0:?] OF specification_usage_constraint;
  specification_id:          text;
  specification_description: OPTIONAL text;
  specification_class:       OPTIONAL text;
END_ENTITY;

ENTITY specification_usage_constraint;                                     (* m1 *)
  element:               text;
  class_id:              text;
END_ENTITY;

ENTITY defined_thread                                                      (* m1 *)
SUBTYPE OF (thread);
  pitch_diameter:        length_measure;
  minor_diameter:        OPTIONAL length_measure;
  crest:                 OPTIONAL length_measure;
END_ENTITY;

  (* *********************************************************** *)
  (* Profile                                                     *)
  (* *********************************************************** *)

ENTITY profile                                                             (* m1 *)
  ABSTRACT SUPERTYPE OF(ONEOF(closed_profile, open_profile));
  placement:                  OPTIONAL axis2_placement_3d;
END_ENTITY;

ENTITY open_profile                                                        (* m1 *)
  ABSTRACT SUPERTYPE OF
  (ONEOF (linear_profile, square_u_profile, rounded_u_profile, tee_profile,
    vee_profile, partial_circular_profile, general_profile))
  SUBTYPE OF(profile);
END_ENTITY;

ENTITY linear_profile                                                      (* m1 *)
SUBTYPE OF (open_profile);
  profile_length:        numeric_parameter;
END_ENTITY;

ENTITY square_u_profile                                                    (* m1 *)
  SUBTYPE OF (open_profile);
  width:                      toleranced_length_measure;
  first_radius:               toleranced_length_measure;
  first_angle:                plane_angle_measure;
  second_radius:              toleranced_length_measure;
  second_angle:               plane_angle_measure;
END_ENTITY;

ENTITY rounded_u_profile                                                   (* m1 *)
  SUBTYPE OF (open_profile);
  width:                      toleranced_length_measure;
END_ENTITY;

ENTITY tee_profile                                                     (* m1 *)
  SUBTYPE OF (open_profile);
  first_angle:                plane_angle_measure;
  second_angle:               plane_angle_measure;
  cross_bar_width:            toleranced_length_measure;
  cross_bar_depth:            toleranced_length_measure;
  radius:                     toleranced_length_measure;
  width:                      toleranced_length_measure;
  first_offset:               toleranced_length_measure;
  second_offset:              toleranced_length_measure;
END_ENTITY;

ENTITY vee_profile                                                     (* m1 *)
  SUBTYPE OF (open_profile);
  profile_radius:             toleranced_length_measure;
  profile_angle:              plane_angle_measure;
  tilt_angle:                 plane_angle_measure;
END_ENTITY;

ENTITY partial_circular_profile                                        (* m1 *)
  SUBTYPE OF (open_profile);
  radius:               toleranced_length_measure;
  sweep_angle:          plane_angle_measure;
END_ENTITY;

ENTITY general_profile                                                 (* m1 *)
  SUBTYPE OF (open_profile);
  its_profile:                bounded_curve;
(*
Informal propositions:
- The entire profile lies in the local yz plane.
- The profile is not self-intersecting.
*)
END_ENTITY;

ENTITY closed_profile                                                  (* m1 *)
  ABSTRACT SUPERTYPE OF
  (ONEOF (rectangular_closed_profile, circular_closed_profile, ngon_profile,
    general_closed_profile))
  SUBTYPE OF(profile);
END_ENTITY;
ENTITY rectangular_closed_profile                                      (* m1 *)
  SUBTYPE OF(closed_profile);
  profile_width:         toleranced_length_measure;
  profile_length:        toleranced_length_measure;
END_ENTITY;

ENTITY circular_closed_profile                                         (* m1 *)
  SUBTYPE OF(closed_profile);
  diameter:              toleranced_length_measure;
END_ENTITY;

ENTITY ngon_profile                                                    (* m1 *)
  SUBTYPE OF(closed_profile);
  diameter:              toleranced_length_measure;
  number_of_sides:       INTEGER;
  circumscribed_or_across_flats: BOOLEAN;
END_ENTITY;
```

```
ENTITY general_closed_profile                                    (* m1 *)
  SUBTYPE OF(closed_profile);
  closed_profile_shape:  bounded_curve;
END_ENTITY;


   (* ***************************************************** *)
   (* Travel path                                            *)
   (* ***************************************************** *)

ENTITY travel_path                                               (* m1 *)
ABSTRACT SUPERTYPE OF(ONEOF(general_path, linear_path, circular_path));
  placement:                 OPTIONAL axis2_placement_3d;
END_ENTITY;

ENTITY general_path                                              (* m1 *)
SUBTYPE OF(travel_path);
  swept_path:            bounded_curve;
END_ENTITY;

ENTITY linear_path                                               (* m1 *)
SUBTYPE OF(travel_path);
  distance:              toleranced_length_measure;
  its_direction:         direction;
END_ENTITY;

ENTITY circular_path                                             (* m1 *)
ABSTRACT SUPERTYPE OF(ONEOF(complete_circular_path, partial_circular_path))
SUBTYPE OF(travel_path);
  radius:                toleranced_length_measure;
END_ENTITY;

ENTITY complete_circular_path                                    (* m1 *)
SUBTYPE OF(circular_path);
END_ENTITY;

ENTITY partial_circular_path                                     (* m1 *)
SUBTYPE OF(circular_path);
  sweep_angle:           plane_angle_measure;
END_ENTITY;

   (* ***************************************************** *)
   (* Surface texture parameter                              *)
   (* ***************************************************** *)

ENTITY surface_texture_parameter;                                (* m1 *)
  its_value:       parameter_value;
  parameter_name:        label;
  measuring_method:      identifier;
  parameter_index:       OPTIONAL identifier;
  applied_surfaces:      SET [1:?] OF machined_surface;
END_ENTITY;

ENTITY machined_surface;                                         (* m1 *)
  its_machining_feature: machining_feature;
  surface_element:       bottom_or_side;
END_ENTITY;

TYPE bottom_or_side = ENUMERATION OF (bottom, side, bottom_and_side);
END_TYPE;
```

```
   (* ************************************************************ *)
   (* ************************************************************ *)
   (*                                                              *)
   (* Executable                                                   *)
   (*                                                              *)
   (* ************************************************************ *)
   (* ************************************************************ *)


ENTITY executable                                                           (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF( workingstep, nc_function,
                               program_structure));
  its_id:                     identifier;
(*
Informal proposition:
its_id shall be unique within the part programme.
*)
END_ENTITY;


   (* ************************************************************ *)
   (* Workingstep                                                  *)
   (* ************************************************************ *)

ENTITY workingstep                                                          (* m0 *)
ABSTRACT SUPERTYPE OF (ONEOF (machining_workingstep, rapid_movement,
  touch_probing))
SUBTYPE OF (executable);
  its_secplane :              elementary_surface;
END_ENTITY;


   (* ************************************************************ *)
   (* Machining workingstep                                        *)
   (* ************************************************************ *)

ENTITY machining_workingstep                                                (* m0 *)
  SUBTYPE OF (workingstep);
its_feature:                manufacturing_feature;
its_operation:              machining_operation;
its_effect:                 OPTIONAL in_process_geometry;
END_ENTITY;

ENTITY in_process_geometry;                                                 (* m1 *)
  as_is:                     OPTIONAL advanced_brep_shape_representation;
  to_be:                     OPTIONAL advanced_brep_shape_representation;
  removal:                   OPTIONAL advanced_brep_shape_representation;
WHERE
  WR1: EXISTS (as_is) OR EXISTS (to_be) OR EXISTS (removal);
END_ENTITY;

   (* ************************************************************ *)
   (* Rapid movement                                               *)
   (* ************************************************************ *)

ENTITY rapid_movement                                                       (* m0 *)
  SUPERTYPE OF (return_home)
  SUBTYPE OF (workingstep, operation);
END_ENTITY;
```

```
ENTITY return_home                                                 (* m0 *)
  SUBTYPE OF (rapid_movement);
END_ENTITY;


   (* ****************************************************** *)
   (* Touch probing                                          *)
   (* ****************************************************** *)

ENTITY touch_probing                                               (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF (workpiece_probing,
    workpiece_complete_probing, tool_probing))
  SUBTYPE OF (workingstep, operation);
  measured_offset: nc_variable;
END_ENTITY;

ENTITY workpiece_probing                                           (* m1 *)
  SUBTYPE OF (touch_probing);
  start_position:        axis2_placement_3d;
  its_workpiece:         workpiece;
  its_direction:         direction;
  expected_value:        toleranced_length_measure;
  its_probe:             touch_probe;
END_ENTITY;

ENTITY workpiece_complete_probing                                  (* m1 *)
  SUBTYPE OF (touch_probing);
  its_workpiece:             workpiece;
  probing_distance:          toleranced_length_measure;
  its_probe:                 touch_probe;
  computed_offset:           offset_vector;
END_ENTITY;

ENTITY touch_probe;                                                (* m1 *)
  its_id: identifier;
END_ENTITY;

ENTITY offset_vector;                                              (* m1 *)
  translate:                 LIST [3:3] OF nc_variable;
  rotate:                    OPTIONAL LIST [3:3] OF nc_variable;
WHERE
  WR1: (SIZEOF(QUERY(i <* translate | NOT EXISTS(i.initial_value))) = 0)
    AND (NOT EXISTS(rotate) OR (SIZEOF(QUERY(i <* rotate |
    NOT EXISTS(i.initial_value))) = 0));
END_ENTITY;

ENTITY tool_probing                                                (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF (tool_length_probing, tool_radius_probing))
  SUBTYPE OF (touch_probing);
  offset:                    cartesian_point;
  max_wear:                  length_measure;
  its_tool:                  machining_tool;
END_ENTITY;

   (* ****************************************************** *)
   (* machining_tool                                          *)
   (* ****************************************************** *)

ENTITY machining_tool                                              (* m0 *)
  ABSTRACT SUPERTYPE;
  its_id:                          label;
```

```
END_ENTITY;

ENTITY cutting_tool                                                   (* m0 *)
 ABSTRACT SUPERTYPE
 SUBTYPE OF (machining_tool);
 its_tool_body:           tool_body;
 its_cutting_edge:        LIST [1:?] OF cutting_component;
 overall_assembly_length: OPTIONAL length_measure;
END_ENTITY;

ENTITY tool_body                                                      (* m0 *)
  ABSTRACT SUPERTYPE;
END_ENTITY;

ENTITY cutting_component;                                             (* m0 *)
  tool_offset_length:   length_measure;
  its_material:          OPTIONAL material;
  technological_data:    OPTIONAL cutting_edge_technological_data;
  expected_tool_life:    OPTIONAL time_measure;
  its_technology:        OPTIONAL technology;
END_ENTITY;

ENTITY cutting_edge_technological_data;                               (* m0 *)
 cutting_angle:         OPTIONAL plane_angle_measure;
 free_angle:            OPTIONAL plane_angle_measure;
 aux_angle:             OPTIONAL plane_angle_measure;
END_ENTITY;

ENTITY tool_length_probing                                            (* m1 *)
  SUBTYPE OF (tool_probing);
END_ENTITY;

ENTITY tool_radius_probing                                            (* m1 *)
  SUBTYPE OF (tool_probing);
END_ENTITY;


  (* ******************************************************** *)
  (* NC_function                                              *)
  (* ******************************************************** *)

ENTITY nc_function                                                    (* m0 *)
  ABSTRACT SUPERTYPE SUBTYPE OF (executable);
END_ENTITY;

ENTITY display_message                                                (* m0 *)
  SUBTYPE OF (nc_function);
  its_text:                  text;
END_ENTITY;

ENTITY optional_stop                                                  (* m0 *)
  SUBTYPE OF (nc_function);
END_ENTITY;

ENTITY program_stop                                                   (* m0 *)
  SUBTYPE OF (nc_function);
END_ENTITY;


ENTITY set_mark                                                       (* m0 *)
  SUBTYPE OF (nc_function);
```

```
END_ENTITY;

ENTITY wait_for_mark                                                  (* m0 *)
  SUBTYPE OF (nc_function);
  its_channel:                channel;
END_ENTITY;


   (* ********************************************************** *)
   (* Program structure                                          *)
   (* ********************************************************** *)

ENTITY program_structure                                              (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF(workplan, parallel, non_sequential, selective,
    if_statement, while_statement, assignment))
  SUBTYPE OF (executable);
END_ENTITY;

   (* ********************************************************** *)
   (* Workplan                                                   *)
   (* ********************************************************** *)

ENTITY workplan                                                       (* m0 *)
  SUBTYPE OF (program_structure);
  its_elements: LIST[0:?] OF executable;
  its_channel:                OPTIONAL channel;
  its_setup:                  OPTIONAL setup;
  its_effect:                 OPTIONAL in_process_geometry;
WHERE
  WR1: SIZEOF(QUERY(it <* its_elements | it = SELF)) = 0;
END_ENTITY;

ENTITY channel;                                                       (* m0 *)
  its_id: identifier;
END_ENTITY;

ENTITY setup;                                                         (* m0 *)
  its_id:                     identifier;
  its_origin:                 OPTIONAL axis2_placement_3d;
  its_secplane:               elementary_surface;
  its_workpiece_setup:    LIST [0:?] OF workpiece_setup;
(*
Informal proposition:
If its_origin is not set, the default for the origin
of the setup is identical with the machine origin.
*)
END_ENTITY;

ENTITY workpiece_setup;                                               (* m0 *)
  its_workpiece:              workpiece;
  its_origin:                 axis2_placement_3d;
  its_offset:                 OPTIONAL offset_vector;
  its_restricted_area:        OPTIONAL restricted_area_select;
  its_instructions:           LIST [0:?] OF setup_instruction;
END_ENTITY;

TYPE restricted_area_select = SELECT (
                      bounded_surface,                          (* ISO 10303-42 *)
                      bounding_geometry_select);
END_TYPE;
```

```
ENTITY setup_instruction;                                              (* m1 *)
  description:         OPTIONAL text;
  external_document:   OPTIONAL identifier;
  WHERE
  WR1: EXISTS (description) OR EXISTS (external_document);
END_ENTITY;

ENTITY parallel                                                        (* m1 *)
  SUBTYPE OF (program_structure);
  branches:                  SET [2:?] OF executable;
END_ENTITY;

ENTITY non_sequential                                                  (* m1 *)
SUBTYPE OF (program_structure);
  its_elements:          SET[2:?] OF executable;
END_ENTITY;

ENTITY selective                                                       (* m1 *)
SUBTYPE OF (program_structure);
  its_elements:          SET[2:?] OF executable;
END_ENTITY;

ENTITY if_statement                                                    (* m1 *)
SUBTYPE OF (program_structure);
  condition:                 boolean_expression;
  true_branch:               executable;
  false_branch:              OPTIONAL executable;
END_ENTITY;

ENTITY while_statement                                                 (* m1 *)
  SUBTYPE OF (program_structure);
  condition:                 boolean_expression;
  body:                      executable;
END_ENTITY;

ENTITY assignment                                                      (* m1 *)
  SUBTYPE OF (program_structure);
  its_lvalue:                nc_variable;
  its_rvalue:                rvalue;
END_ENTITY;

ENTITY nc_variable;                                                    (* m1 *)
  its_name:                  LABEL;
  initial_value:             OPTIONAL NUMBER;
END_ENTITY;

ENTITY nc_constant;                                                    (* m1 *)
  its_name:                  LABEL;
  its_value:                 OPTIONAL NUMBER;
END_ENTITY;

TYPE rvalue = SELECT(nc_constant, nc_variable);
END_TYPE;

ENTITY boolean_expression                                              (* m1 *)
  ABSTRACT SUPERTYPE OF(ONEOF(unary_boolean_expression, binary_boolean_expression,
    multiple_arity_boolean_expression, comparison_expression));
END_ENTITY;


ENTITY unary_boolean_expression                                        (* m1 *)
```

```
    ABSTRACT SUPERTYPE OF(not_expression)
    SUBTYPE OF (boolean_expression);
    operand: boolean_expression;
  END_ENTITY;

  ENTITY not_expression                                                    (* m1 *)
    SUBTYPE OF (unary_boolean_expression);
  END_ENTITY;

  ENTITY binary_boolean_expression                                         (* m1 *)
    ABSTRACT SUPERTYPE OF(xor_expression)
    SUBTYPE OF (boolean_expression);
    operand1: boolean_expression;
    operand2: boolean_expression;
  END_ENTITY;

  ENTITY xor_expression                                                    (* m1 *)
    SUBTYPE OF (binary_boolean_expression);
  END_ENTITY;

  ENTITY multiple_arity_boolean_expression                                 (* m1 *)
  ABSTRACT SUPERTYPE OF(ONEOF(and_expression, or_expression))
  SUBTYPE OF (boolean_expression);
    operands: LIST [2:?] OF boolean_expression;
  END_ENTITY;

  ENTITY and_expression                                                    (* m1 *)
  SUBTYPE OF (multiple_arity_boolean_expression);
  END_ENTITY;

  ENTITY or_expression                                                     (* m1 *)
  SUBTYPE OF (multiple_arity_boolean_expression);
  END_ENTITY;

  ENTITY comparison_expression                                             (* m1 *)
    ABSTRACT SUPERTYPE OF(ONEOF(comparison_equal, comparison_not_equal,
      comparison_greater, comparison_greater_equal, comparison_less,
      comparison_less_equal))SUBTYPE OF (boolean_expression);
    operand1:                   nc_variable;
    operand2:                   rvalue;
  END_ENTITY;

  ENTITY comparison_equal                                                  (* m1 *)
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_not_equal                                              (* m1 *)
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_greater                                                (* m1 *)
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_greater_equal                                          (* m1 *)
    SUBTYPE OF (comparison_expression);
  END_ENTITY;

  ENTITY comparison_less                                                   (* m1 *)
    SUBTYPE OF (comparison_expression);
  END_ENTITY;
```

```
ENTITY comparison_less_equal                                        (* m1 *)
  SUBTYPE OF (comparison_expression);
END_ENTITY;


   (* ********************************************************* *)
   (* ********************************************************* *)
   (*                                                           *)
   (* Operation                                                 *)
   (*                                                           *)
   (* ********************************************************* *)
   (* ********************************************************* *)


ENTITY operation                                                    (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF (machining_operation, rapid_movement,
    touch_probing));
  its_toolpath:               OPTIONAL toolpath_list;
  its_tool_direction:         OPTIONAL tool_direction;
END_ENTITY;

ENTITY toolpath_list;                                               (* m0 *)
  its_list: LIST [1:?] OF toolpath;
END_ENTITY;

ENTITY tool_direction                                               (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF (two_axes, three_axes));
END_ENTITY;

ENTITY two_axes                                                     (* m0 *)
  SUBTYPE OF (tool_direction);
END_ENTITY;

ENTITY three_axes                                                   (* m0 *)
  SUBTYPE OF (tool_direction);
END_ENTITY;


   (* ********************************************************* *)
   (* Machining operation                                       *)
   (* ********************************************************* *)

ENTITY machining_operation                                          (* m0 *)
  ABSTRACT SUPERTYPE
  SUBTYPE OF (operation);
  its_id:                 identifier;
  retract_plane:          OPTIONAL length_measure;
  start_point:        OPTIONAL cartesian_point;
  its_tool:               machining_tool;
  its_technology:         technology;
  its_machine_functions:  machine_functions;
(*
Informal proposition:
If attribute SELF\operation.its_toolpath exists, then attributes
its_machining_strategy, retract_plane and start_point, if present, are for
information only.
*)
END_ENTITY;
```

```
   (* ************************************************************ *)
   (* Technology                                                   *)
   (* ************************************************************ *)

ENTITY technology                                                           (* m0 *)
  ABSTRACT SUPERTYPE;
  feedrate:                      OPTIONAL speed_measure;
  feedrate_reference:            tool_reference_point;
END_ENTITY;

TYPE tool_reference_point = ENUMERATION OF (tcp, ccp);
END_TYPE;


   (* ************************************************************ *)
   (* Machine functions                                            *)
   (* ************************************************************ *)

ENTITY machine_functions                                                    (* m0 *)
  ABSTRACT SUPERTYPE;
END_ENTITY;


   (* ************************************************************ *)
   (* ************************************************************ *)
   (*                                                              *)
   (* Toolpath                                                     *)
   (*                                                              *)
   (* ************************************************************ *)
   (* ************************************************************ *)


ENTITY toolpath                                                             (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF(feedstop, trajectory, parameterised_path));
  its_priority:  BOOLEAN;
  its_type     : toolpath_type;
  its_speed    : OPTIONAL toolpath_speedprofile;
  its_technology: OPTIONAL technology;
  its_machine_functions: OPTIONAL machine_functions;
END_ENTITY;


   (* ************************************************************ *)
   (* Toolpath type                                                *)
   (* ************************************************************ *)

TYPE toolpath_type = ENUMERATION OF (
  approach,
  lift,
  connect,
  non_contact,
  contact,
  trajectory_path
);
END_TYPE;
```

```
   (* ********************************************************* *)
   (* Toolpath speedprofile                                      *)
   (* ********************************************************* *)

TYPE toolpath_speedprofile = SELECT (
  toolpath_speed,
  positive_ratio_measure,
  speed_name
);
END_TYPE;

ENTITY toolpath_speed;                                            (* m1 *)
  speed:                       b_spline_curve;
WHERE
  WR1: speed\geometric_representation_item.dim = 1;
(*
Informal proposition:
- speed shall have only values greater than zero.
*)
END_ENTITY;

TYPE speed_name = ENUMERATION OF(RAPID);
END_TYPE;


   (* ********************************************************* *)
   (* Feedstop                                                   *)
   (* ********************************************************* *)

ENTITY feedstop                                                   (* m0 *)
  SUBTYPE OF (toolpath);
  dwell:                       time_measure;
END_ENTITY;


   (* ********************************************************* *)
   (* Trajectory                                                 *)
   (* ********************************************************* *)

ENTITY trajectory                                                 (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF(cutter_location_trajectory,
    cutter_contact_trajectory, axis_trajectory))
  SUBTYPE OF (toolpath);
  its_direction:               OPTIONAL BOOLEAN;
END_ENTITY;


   (* ********************************************************* *)
   (* Cutter location trajectory                                 *)
   (* ********************************************************* *)

ENTITY cutter_location_trajectory                                 (* m0 *)
  SUBTYPE OF (trajectory);
  basiccurve:                  bounded_curve;
  its_toolaxis:                OPTIONAL bounded_curve;
  surface_normal:              OPTIONAL bounded_curve;
(*
Information proposition:
its_toolaxis must have the same must have the same parameter range as basiccurve.
*)
END_ENTITY;
```

```
   (* ************************************************************ *)
   (* Cutter contact trajectory                                    *)
   (* ************************************************************ *)

ENTITY cutter_contact_trajectory                                            (* m1 *)
  SUBTYPE OF (trajectory);
  basiccurve:                 curve_with_surface_normal;
  its_toolaxis:               OPTIONAL bounded_curve;
  its_contact_type:           OPTIONAL contact_type;
(*
Informal proposition:
its_toolaxis must have the same must have the same parameter range as basiccurve.
*)
END_ENTITY;

TYPE curve_with_surface_normal = SELECT (
  bounded_pcurve, curve_with_normal_vector
);
END_TYPE;

ENTITY curve_with_normal_vector;                                            (* m1 *)
  basiccurve:                 bounded_curve;
  surface_normal:             bounded_curve;
(*
Informal proposition:
basiccurve and surface_normal must have the same parameter range
*)
END_ENTITY;

TYPE contact_type = ENUMERATION OF (side, front);
END_TYPE;


   (* ************************************************************ *)
   (* Axis trajectory                                              *)
   (* ************************************************************ *)

ENTITY axis_trajectory                                                      (* m0 *)
  SUBTYPE OF (trajectory);
  axis_list:                  LIST [1:?] OF identifier;
  commands:                   LIST [1:?] OF bounded_curve;
WHERE
  WR1: SIZEOF(QUERY(cmd <* commands |
    cmd\geometric_representation_item.dim <> 1)) = 0;
END_ENTITY;


   (* ************************************************************ *)
   (* Parameterised path                                           *)
   (* ************************************************************ *)

ENTITY parameterised_path                                                   (* m0 *)
  ABSTRACT SUPERTYPE OF (ONEOF (approach_lift_path, connector))
  SUBTYPE OF (toolpath);
END_ENTITY;
```

```
  (* ************************************************************ *)
  (* Connector                                                    *)
  (* ************************************************************ *)

ENTITY connector                                                            (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF(connect_secplane, connect_direct))
  SUBTYPE OF (parameterised_path);
END_ENTITY;

ENTITY connect_secplane                                                     (* m1 *)
  SUBTYPE OF (connector);
  up_dir  : OPTIONAL direction;
  down_dir: OPTIONAL direction;
END_ENTITY;

ENTITY connect_direct                                                       (* m1 *)
  SUBTYPE OF (connector);
END_ENTITY;


  (* ************************************************************ *)
  (* Approach and lift movement                                   *)
  (* ************************************************************ *)

ENTITY approach_lift_path                                                   (* m1 *)
  ABSTRACT SUPERTYPE OF (ONEOF (ap_lift_path_angle, ap_lift_path_tangent))
  SUBTYPE OF (parameterised_path);
  fix_point:                  cartesian_point;
  fix_point_dir:              OPTIONAL direction;
END_ENTITY;

ENTITY ap_lift_path_angle                                                   (* m1 *)
  SUBTYPE OF (approach_lift_path);
  angle:                      plane_angle_measure;
  benddist:                   positive_length_measure;
END_ENTITY;

ENTITY ap_lift_path_tangent                                                 (* m1 *)
  SUBTYPE OF (approach_lift_path);
  radius:                     positive_length_measure;
END_ENTITY;


  (* ************************************************************ *)
  (* Rules                                                        *)
  (* ************************************************************ *)

RULE dependent_instantiable_representation_item FOR (representation_item);
WHERE
  WR1: SIZEOF (QUERY (ri <* representation_item |
        NOT (SIZEOF (USEDIN (ri, '')) >= 1))) = 0;
END_RULE;

RULE dependent_instantiable_shape_representation FOR (shape_representation);
WHERE
  WR1: SIZEOF (QUERY (sr <* shape_representation |
       NOT (SIZEOF(USEDIN(sr, '')) >= 1))) = 0;
END_RULE;

RULE geometric_representation_item_3d FOR (geometric_representation_item);
WHERE
```

```
   WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
        NOT ((dimension_of (gri) = 3) OR
        (SIZEOF (bag_to_set (USEDIN (gri, '')) - bag_to_set (USEDIN (gri,
        'MACHINING_SCHEMA.DEFINITIONAL_REPRESENTATION.ITEMS'))) = 0)
        ))) = 0;
END_RULE;

END_SCHEMA; (* machining_schema *)
```

Annex B
(normative)

Short names of entities

Entity names	Short names
ADDRESS	ADDRSS
ADVANCED_BREP_SHAPE_REPRESENTATION	ABSR
ADVANCED_FACE	ADVFC
AND_EXPRESSION	ANDEXP
ANGLE_TAPER	ANGTPR
AP_LIFT_PATH_ANGLE	ALPA
AP_LIFT_PATH_TANGENT	ALPT
APPROACH_LIFT_PATH	APLFPT
APPROVAL	APPRVL
APPROVAL_STATUS	APPSTT
ASSIGNMENT	ASSGNM
AXIS_TRAJECTORY	AXSTRJ
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_3D	A2PL3D
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BINARY_BOOLEAN_EXPRESSION	BNBLEX
BLIND_BOTTOM_CONDITION	BLBTCN
BLOCK	BLOCK
BOOLEAN_EXPRESSION	BLNEXP
BOSS	BOSS
BOUNDED_CURVE	BNDCRV
BOUNDED_PCURVE	BNDPCR
BOUNDED_SURFACE	BNDSRF
CALENDAR_DATE	CLNDT
CARTESIAN_POINT	CRTPNT
CATALOGUE_THREAD	CTLTHR
CHAMFER	CHMFR

Entity names	Short names
CHANNEL	CHNNL
CIRCLE	CIRCLE
CIRCULAR_CLOSED_PROFILE	CRCLPR
CIRCULAR_CLOSED_SHAPE_PROFILE	CCSP
CIRCULAR_OFFSET	CRCOFF
CIRCULAR_OMIT	CRCOMT
CIRCULAR_PATH	CRCPTH
CIRCULAR_PATTERN	CRCPTT
CLOSED_POCKET	CLSPCK
CLOSED_PROFILE	CLSPRF
CLOSED_SHELL	CLSSHL
COMPARISON_EQUAL	CMPEQL
COMPARISON_EXPRESSION	CMPEXP
COMPARISON_GREATER	CMPGRT
COMPARISON_GREATER_EQUAL	CMGREQ
COMPARISON_LESS	CMPLSS
COMPARISON_LESS_EQUAL	CMLSEQ
COMPARISON_NOT_EQUAL	CMNTEQ
COMPLETE_CIRCULAR_PATH	CMCRPT
COMPOSITE_CURVE	CMPCRV
COMPOSITE_CURVE_SEGMENT	CMCRSG
COMPOUND_FEATURE	CMPFTR
CONIC	CONIC
CONICAL_HOLE_BOTTOM	CNHLBT
CONNECT_DIRECT	CNNDRC
CONNECT_SECPLANE	CNNSCP
CONNECTED_FACE_SET	CNFCST
CONNECTOR	CNNCTR
COORDINATED_UNIVERSAL_TIME_OFFSET	CUTO
COUNTERBORE_HOLE	CNTHL
COUNTERSUNK_HOLE	CNT0
CURVE	CURVE
CURVE_WITH_NORMAL_VECTOR	CWNV
CUTTER_CONTACT_TRAJECTORY	CTCNTR
CUTTER_LOCATION_TRAJECTORY	CTLCTR
CUTTING_COMPONENT	CTTCMP
CUTTING_EDGE_TECHNOLOGICAL_DATA	CETD

Entity names	Short names
CUTTING_TOOL	CTTTL
DATE	DATE
DATE_AND_TIME	DTANTM
DEFINED_THREAD	DFNTHR
DEFINITIONAL_REPRESENTATION	DFNRPR
DESCRIPTIVE_PARAMETER	DSCPRM
DIAMETER_TAPER	DMTTPR
DIRECTION	DRCTN
DISPLAY_MESSAGE	DSPMSS
EDGE	EDGE
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
EDGE_ROUND	EDGRND
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
EXECUTABLE	EXCTBL
FACE	FACE
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FACETED_BREP	FCTBR
FEEDSTOP	FDSTP
FLAT_HOLE_BOTTOM	FLHLBT
FLAT_SLOT_END_TYPE	FSET
FLAT_WITH_RADIUS_HOLE_BOTTOM	FWRHB
FOUNDED_ITEM	FNDITM
GENERAL_CLOSED_PROFILE	GNCLPR
GENERAL_OUTSIDE_PROFILE	GNOTPR
GENERAL_PATH	GNRPTH
GENERAL_PATTERN	GNRPTT
GENERAL_POCKET_BOTTOM_CONDITION	GPBC
GENERAL_PROFILE	GNRPRF
GENERAL_PROFILE_FLOOR	GNPRFL
GENERAL_SHAPE_PROFILE	GNSHPR
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
HOLE_BOTTOM_CONDITION	HLBTCN

Entity names	Short names
HYPERBOLA	HYPRBL
IF_STATEMENT	IFSTT
IN_PROCESS_GEOMETRY	INPRGM
LIMITS_AND_FITS	LMANFT
LINE	LINE
LINEAR_PATH	LNRPTH
LINEAR_PROFILE	LNRPRF
LOCAL_TIME	LCLTM
LOOP	LOOP
LOOP_SLOT_END_TYPE	LSET
MACHINE_FUNCTIONS	MCHFNC
MACHINED_SURFACE	MCHSRF
MACHINING_FEATURE	MCHFTR
MACHINING_OPERATION	MCHOPR
MACHINING_TOOL	MCHTL
MACHINING_WORKINGSTEP	MCHWRK
MANIFOLD_SOLID_BREP	MNSLBR
MANUFACTURING_FEATURE	MNFFTR
MAPPED_ITEM	MPPITM
MATERIAL	MTRL
MULTIPLE_ARITY_BOOLEAN_EXPRESSION	MABE
NC_CONSTANT	NCCNS
NC_FUNCTION	NCFNC
NC_VARIABLE	NCVRB
NGON_PROFILE	NGNPRF
NON_SEQUENTIAL	NNSQN
NOT_EXPRESSION	NTEXP
NUMERIC_PARAMETER	NMRPRM
OFFSET_VECTOR	OFFVCT
OPEN_POCKET	OPNPCK
OPEN_PROFILE	OPNPRF
OPEN_SHELL	OPNSHL
OPEN_SLOT_END_TYPE	OSET
OPERATION	OPRTN
OPTIONAL_STOP	OPTSTP
OR_EXPRESSION	OREXP
ORDINAL_DATE	ORDDT

Entity names	Short names
ORIENTED_CLOSED_SHELL	ORCLSH
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
ORIENTED_OPEN_SHELL	OROPSH
ORIENTED_PATH	ORNPTH
ORIENTED_SURFACE	ORNSRF
PARABOLA	PRBL
PARALLEL	PRLLL
PARAMETERISED_PATH	PRMPTH
PARTIAL_AREA_DEFINITION	PRARDF
PARTIAL_CIRCULAR_PATH	PRCRPT
PARTIAL_CIRCULAR_PROFILE	PRCRPR
PARTIAL_CIRCULAR_SHAPE_PROFILE	PCSP
PATH	PATH
PCURVE	PCURVE
PERSON	PERSON
PERSON_AND_ADDRESS	PRANAD
PLACEMENT	PLCMNT
PLANAR_FACE	PLNFC
PLANAR_POCKET_BOTTOM_CONDITION	PPBC
PLANAR_PROFILE_FLOOR	PLPRFL
PLANE	PLANE
PLUS_MINUS_VALUE	PLMNVL
POCKET	POCKET
POCKET_BOTTOM_CONDITION	PCBTCN
POINT	POINT
POLYLINE	PLYLN
POLY_LOOP	PLYLP
PROFILE	PRFL
PROFILE_FEATURE	PRFFTR
PROFILE_FLOOR	PRFFLR
PROGRAM_STOP	PRGSTP
PROGRAM_STRUCTURE	PRGSTR
PROJECT	PRJCT
PROPERTY_PARAMETER	PRPPRM
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR

Entity names	Short names
RADIUSED_POCKET_BOTTOM_CONDITION	RPBC
RADIUSED_SLOT_END_TYPE	RSET
RAPID_MOVEMENT	RPDMVM
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
RECTANGULAR_CLOSED_PROFILE	RCCLPR
RECTANGULAR_CLOSED_SHAPE_PROFILE	RCSP
RECTANGULAR_OFFSET	RCTOFF
RECTANGULAR_OMIT	RCTOMT
RECTANGULAR_OPEN_SHAPE_PROFILE	ROSP
RECTANGULAR_PATTERN	RCTPTT
REGION	REGION
REGION_PROJECTION	RGNPRJ
REGION_SURFACE_LIST	RGSRLS
REPLICATE_FEATURE	RPLFTR
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_MAP	RPRMP
RETURN_HOME	RTRHM
RIGHT_CIRCULAR_CYLINDER	RGCRCY
ROUND_HOLE	RNDHL
ROUNDED_END	RNDEND
ROUNDED_U_PROFILE	RNUPR
SELECTIVE	SLCTV
SET_MARK	STMRK
SETUP	SETUP
SETUP_INSTRUCTION	STPINS
SHAPE_PROFILE	SHPPRF
SHAPE_REPRESENTATION	SHPRPR
SLOT	SLOT
SLOT_END_TYPE	SLENTY
SOLID_MODEL	SLDMDL
SPECIFICATION	SPCFCT
SPECIFICATION_USAGE_CONSTRAINT	SPUSCN
SPHERICAL_CAP	SPHCP
SPHERICAL_HOLE_BOTTOM	SPHLBT

Entity names	Short names
SPHERICAL_SURFACE	SPHSRF
SQUARE_U_PROFILE	SQUPR
STEP	STEP
SURFACE	SRFC
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_TEXTURE_PARAMETER	SRTXPR
SWEPT_SURFACE	SWPSRF
TECHNOLOGY	TCHNLG
TEE_PROFILE	TPRF
THREAD	THREAD
THREE_AXES	THRAXS
THROUGH_BOTTOM_CONDITION	THBTCN
THROUGH_POCKET_BOTTOM_CONDITION	TPBC
THROUGH_PROFILE_FLOOR	THPRFL
TOLERANCED_LENGTH_MEASURE	TLLNMS
TOOL_BODY	TLBDY
TOOL_DIRECTION	TLDRC
TOOL_LENGTH_PROBING	TLLNPR
TOOL_PROBING	TLPRB
TOOL_RADIUS_PROBING	TLRDPR
TOOLPATH	TLPTH
TOOLPATH_FEATURE	TLPFTR
TOOLPATH_LIST	TLPLST
TOOLPATH_SPEED	TLPSPD
TOPOLOGICAL_REGION	TPLRGN
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOUCH_PROBE	TCH0
TOUCH_PROBING	TCHPRB
TRAJECTORY	TRJCTR
TRANSITION_FEATURE	TRNFTR
TRAVEL_PATH	TRVPTH
TRIMMED_CURVE	TRMCRV
TWO_AXES	TWAXS
TWO5D_MANUFACTURING_FEATURE	TWMNFT
UNARY_BOOLEAN_EXPRESSION	UNBLEX
UNIFORM_CURVE	UNFCRV

Entity names	Short names
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VEE_PROFILE	VPRF
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
WAIT_FOR_MARK	WTFRMR
WEEK_OF_YEAR_AND_DAY_DATE	WOYADD
WHILE_STATEMENT	WHLSTT
WOODRUFF_SLOT_END_TYPE	WSET
WORKINGSTEP	WRKNGS
WORKPIECE	WRKPC
WORKPIECE_COMPLETE_PROBING	WRCMPR
WORKPIECE_PROBING	WRKPRB
WORKPIECE_SETUP	WRKSTP
WORKPLAN	WRKPLN
XOR_EXPRESSION	XREXP

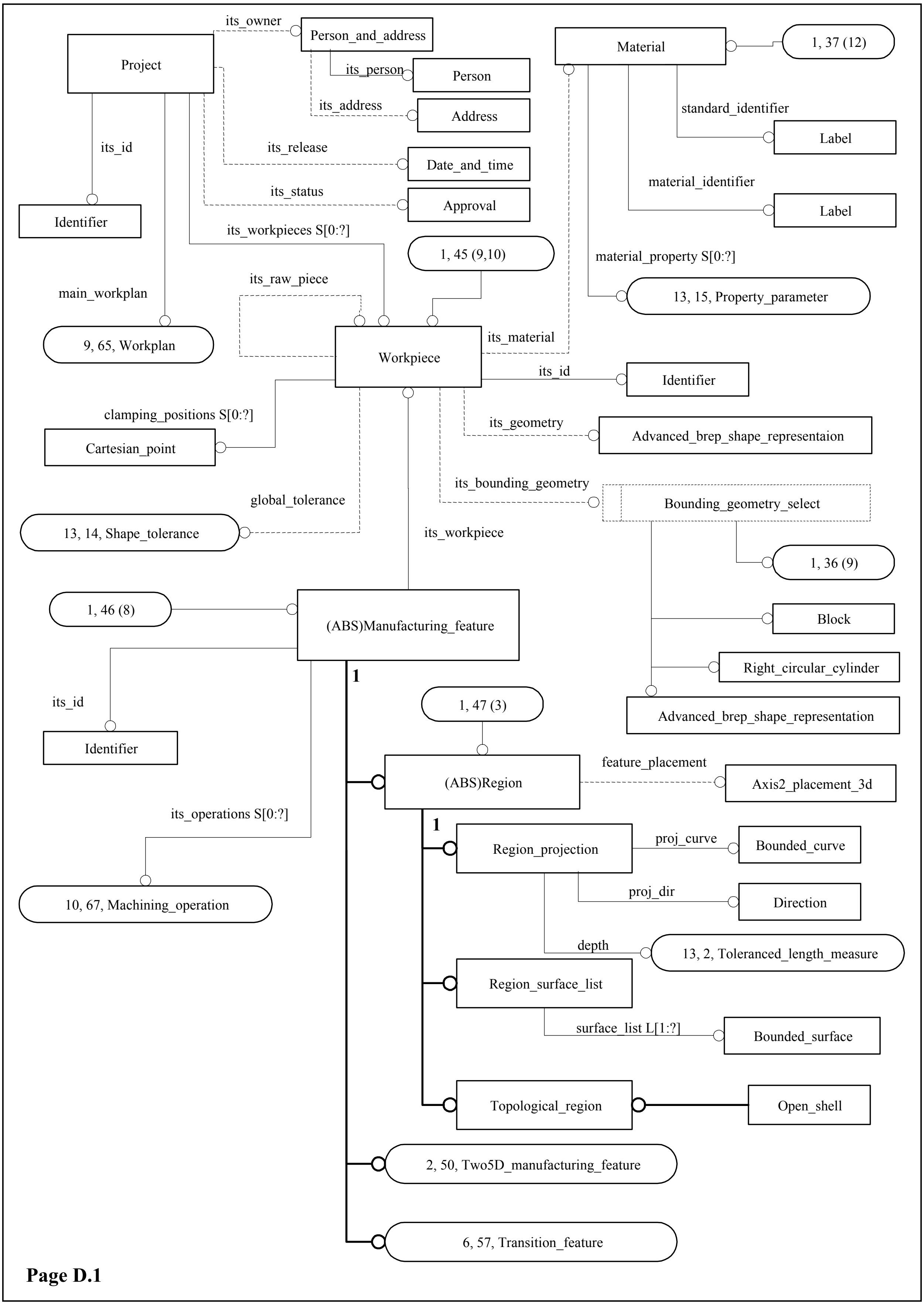
Annex C
(normative)

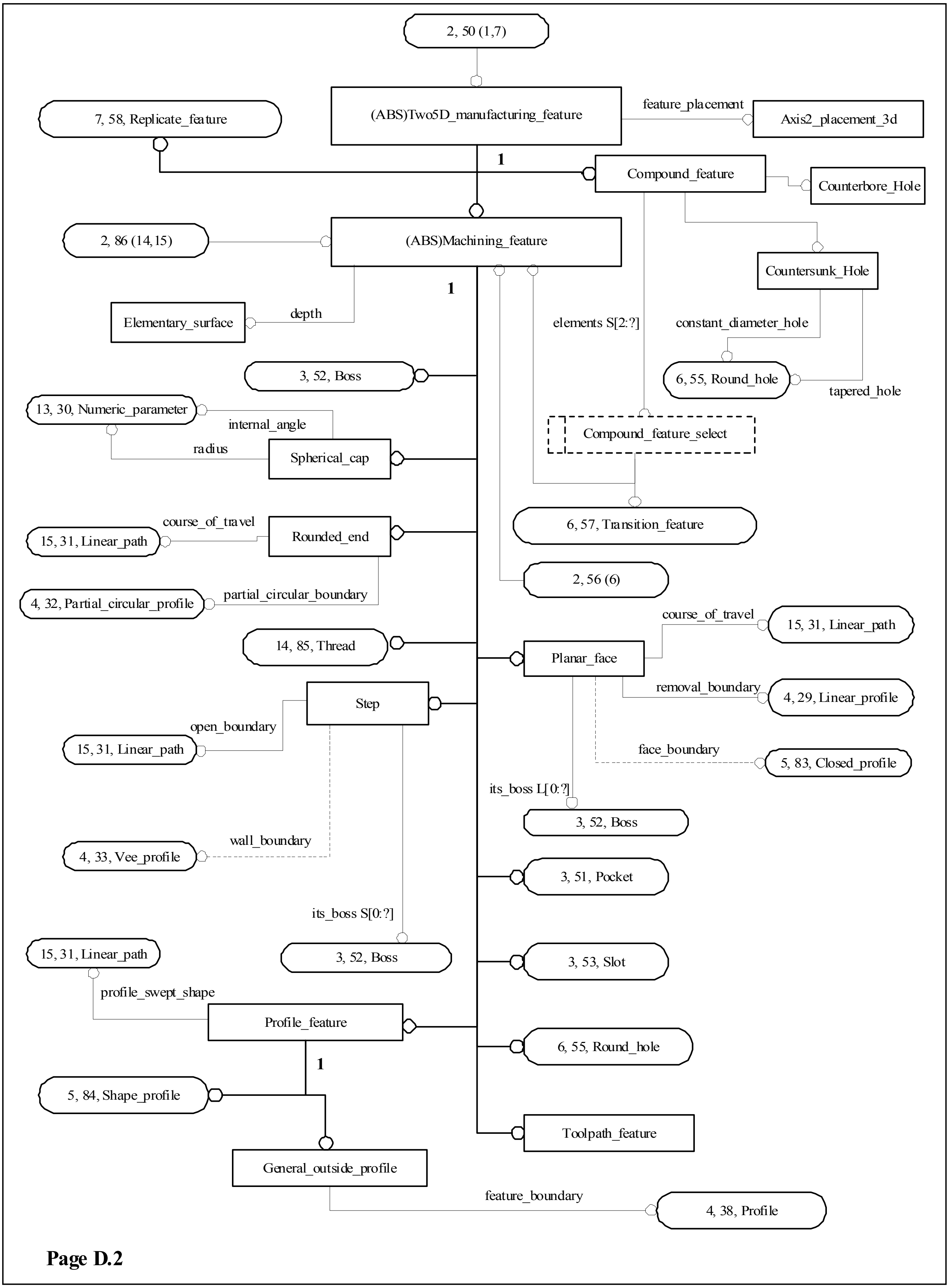
Implementation method specific requirements

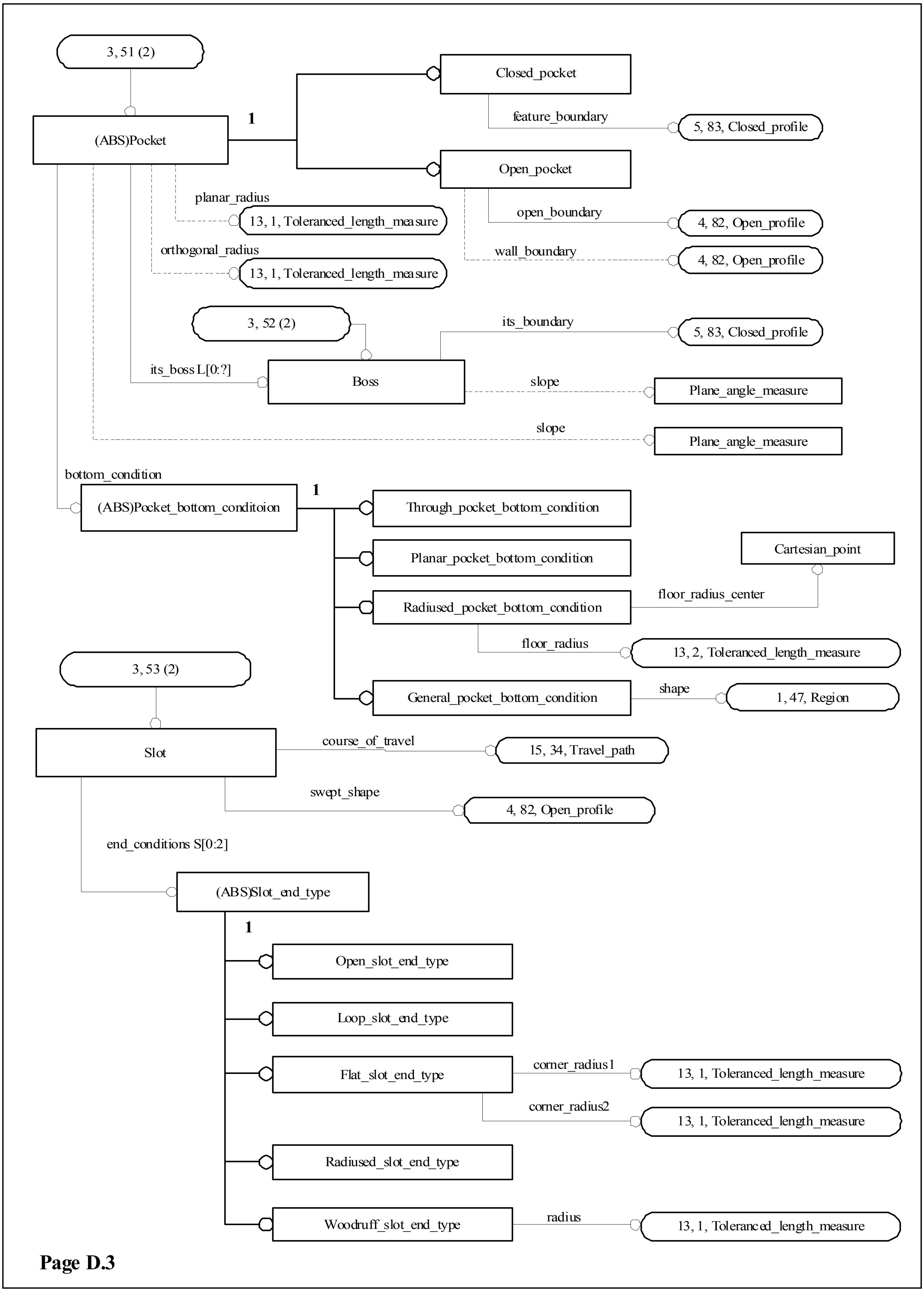
The implementation method defines what type of exchange behaviour is required to this part of ISO 14649. Conformance to this part of ISO 14649 shall be realised in an exchange structure. The file format shall be encoded according to the syntax and EXPRESS language mapping defined in ISO 10303-21 and annotated listing defined in Annex A of this part of ISO 14649. The header of the exchange structure shall identify use of this part of ISO 14649 by the schema name 'machining_schema'.

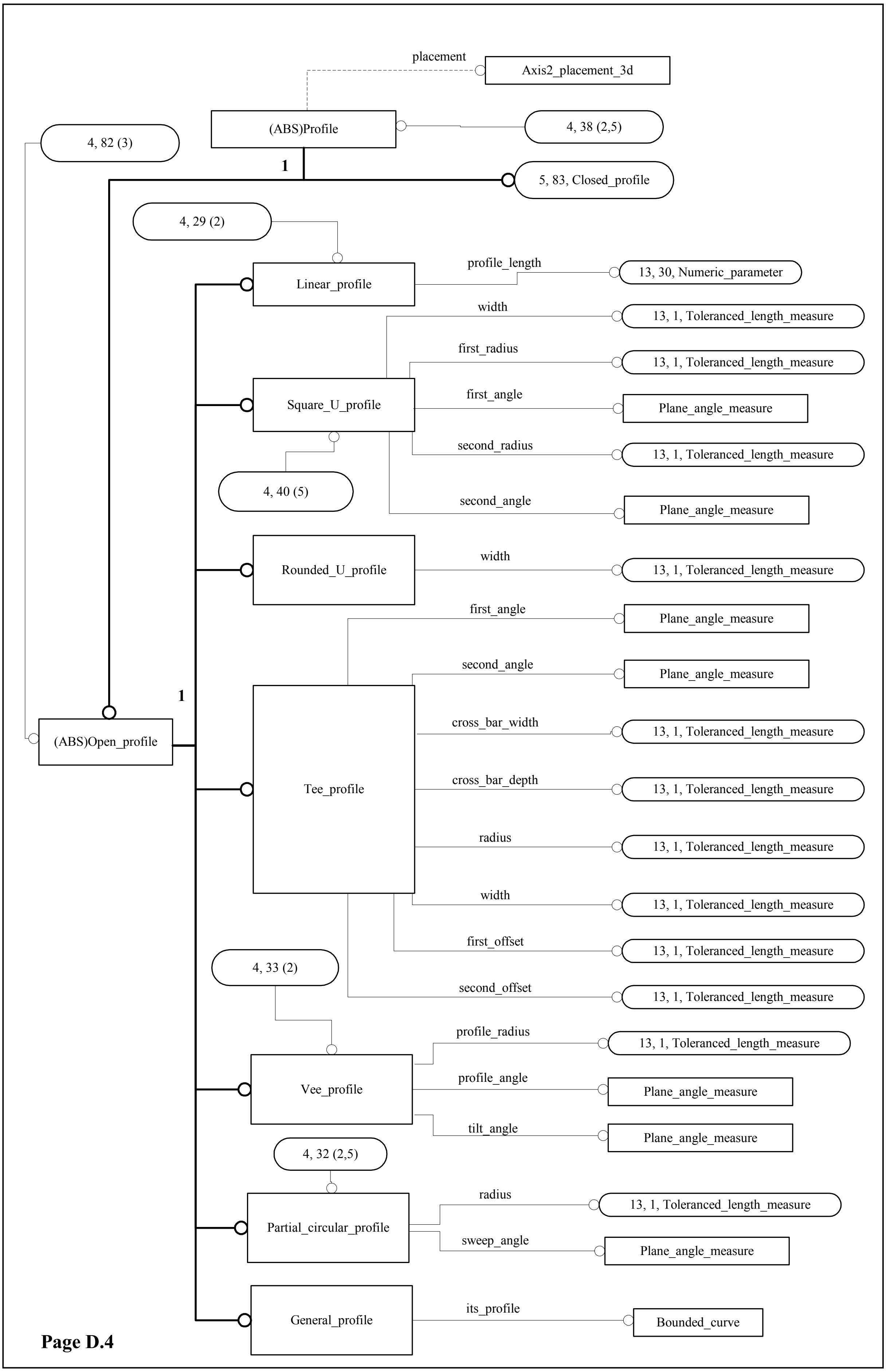
Annex D
(informative)

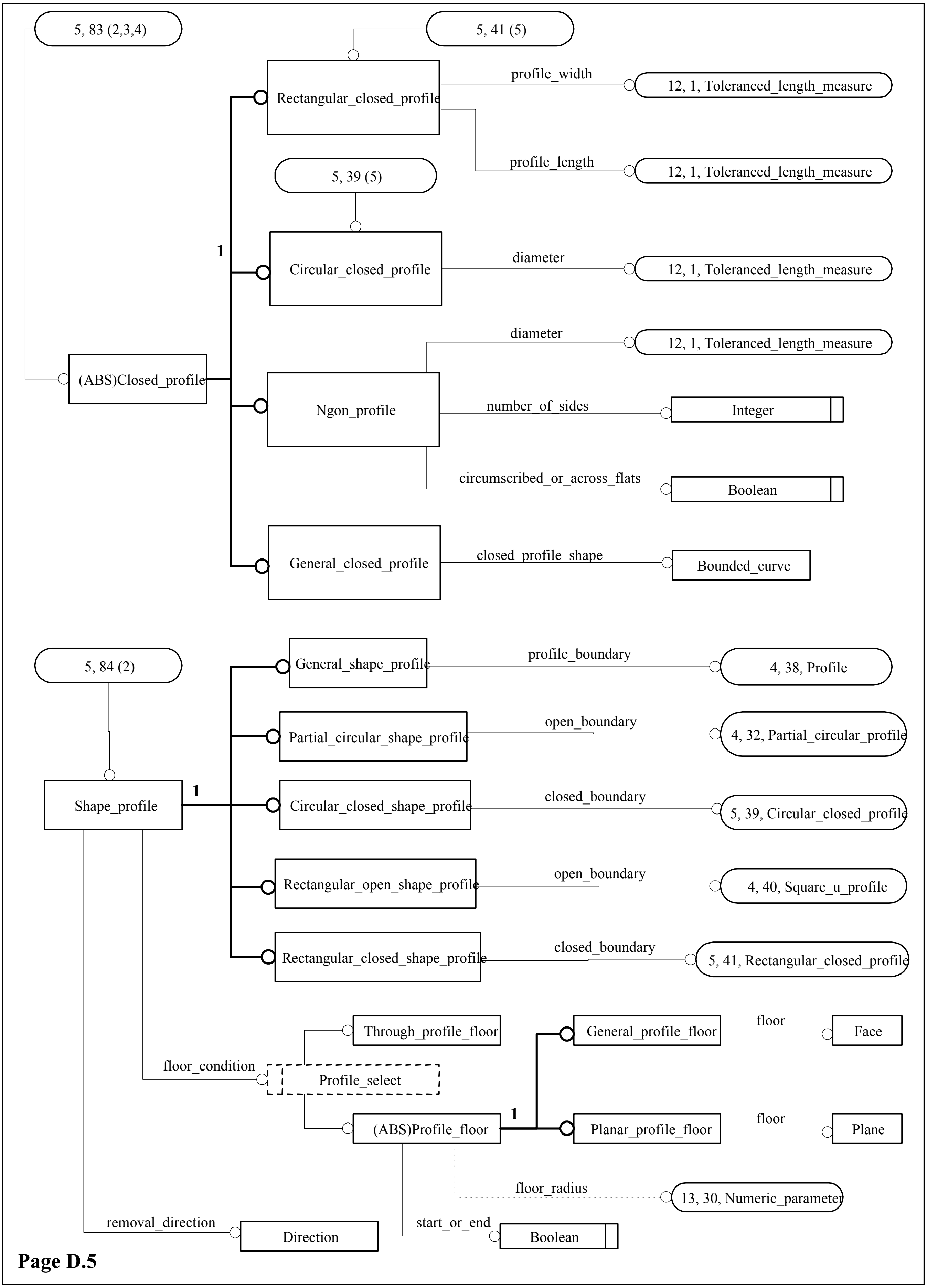
EXPRESS-G figures

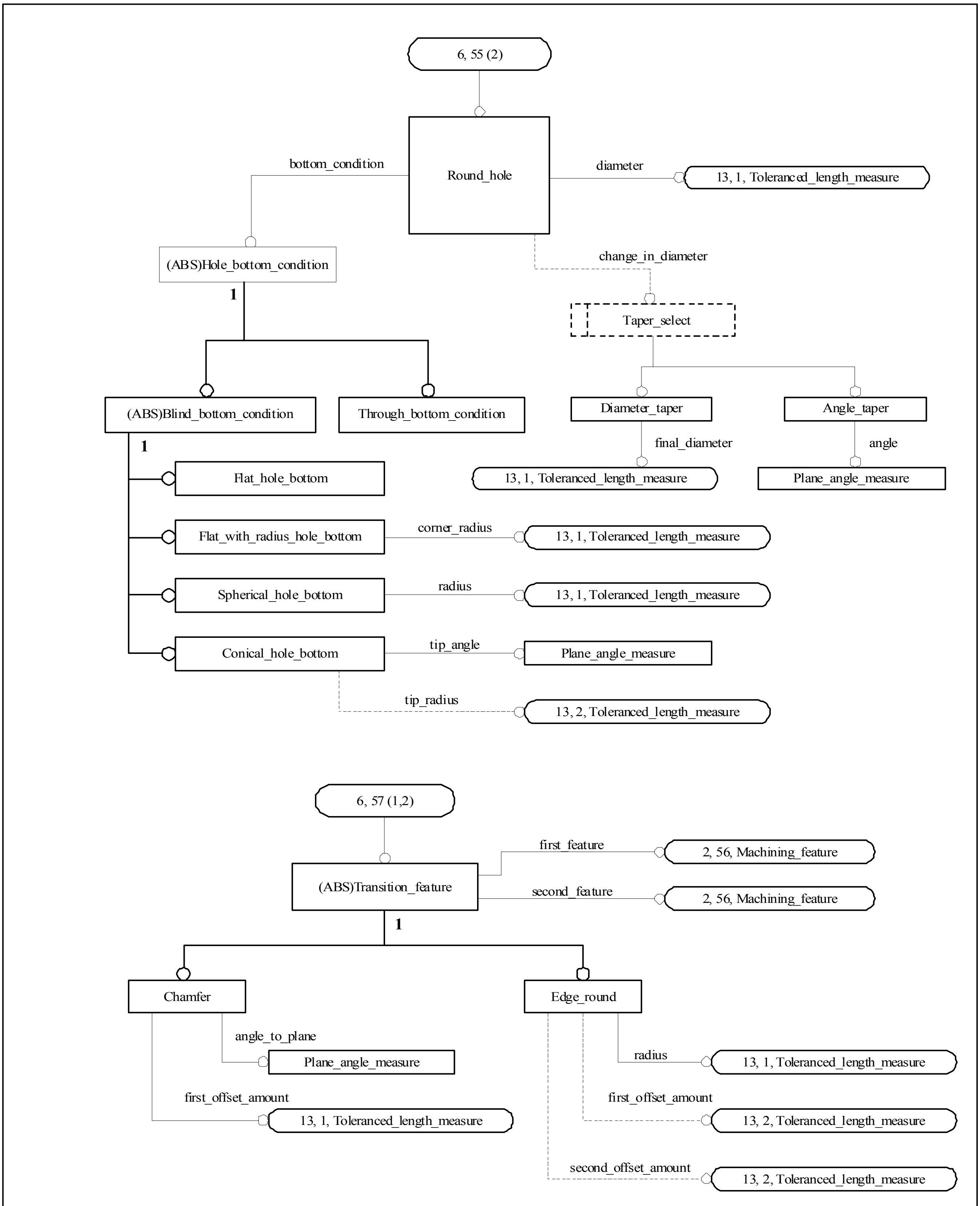

its_owner
Person_and_address
Project
its_person
Person
its_address
Address
its_id
its_release
Date_and_time
its_status
Approval
Identifier
its_workpieces S[0:?]
1, 45 (9,10)
its_raw_piece
main_workplan
9, 65, Workplan
Workpiece
its_material
its_id
Identifier
clamping_positions S[0:?]
Cartesian_point
its_geometry
Advanced_brep_shape_representaion
its_bounding_geometry
Bounding_geometry_select
global_tolerance
13, 14, Shape_tolerance
its_workpiece
1, 36 (9)
Block
Right_circular_cylinder
Advanced_brep_shape_representation
Material
1, 37 (12)
standard_identifier
Label
material_identifier
Label
material_property S[0:?]
13, 15, Property_parameter
1, 46 (8)
(ABS)Manufacturing_feature
1
its_id
Identifier
1, 47 (3)
(ABS)Region
feature_placement
Axis2_placement_3d
its_operations S[0:?]
1
Region_projection
proj_curve
Bounded_curve
proj_dir
Direction
10, 67, Machining_operation
depth
13, 2, Toleranced_length_measure
Region_surface_list
surface_list L[1:?]
Bounded_surface
Topological_region
Open_shell
2, 50, Two5D_manufacturing_feature
6, 57, Transition_feature
Page D.1


2, 50 (1,7)
(ABS)Two5D_manufacturing_feature
feature_placement
Axis2_placement_3d
7, 58, Replicate_feature
1
Compound_feature
Counterbore_Hole
2, 86 (14,15)
(ABS)Machining_feature
Countersunk_Hole
1
Elementary_surface
depth
elements S[2:?]
constant_diameter_hole
3, 52, Boss
6, 55, Round_hole
tapered_hole
13, 30, Numeric_parameter
internal_angle
Compound_feature_select
radius
Spherical_cap
6, 57, Transition_feature
course_of_travel
15, 31, Linear_path
Rounded_end
2, 56 (6)
4, 32, Partial_circular_profile
partial_circular_boundary
course_of_travel
15, 31, Linear_path
14, 85, Thread
Planar_face
removal_boundary
4, 29, Linear_profile
Step
open_boundary
15, 31, Linear_path
face_boundary
5, 83, Closed_profile
its_boss L[0:?]
3, 52, Boss
wall_boundary
4, 33, Vee_profile
3, 51, Pocket
its_boss S[0:?]
3, 52, Boss
15, 31, Linear_path
3, 53, Slot
profile_swept_shape
Profile_feature
6, 55, Round_hole
1
5, 84, Shape_profile
Toolpath_feature
General_outside_profile
feature_boundary
4, 38, Profile
Page D.2

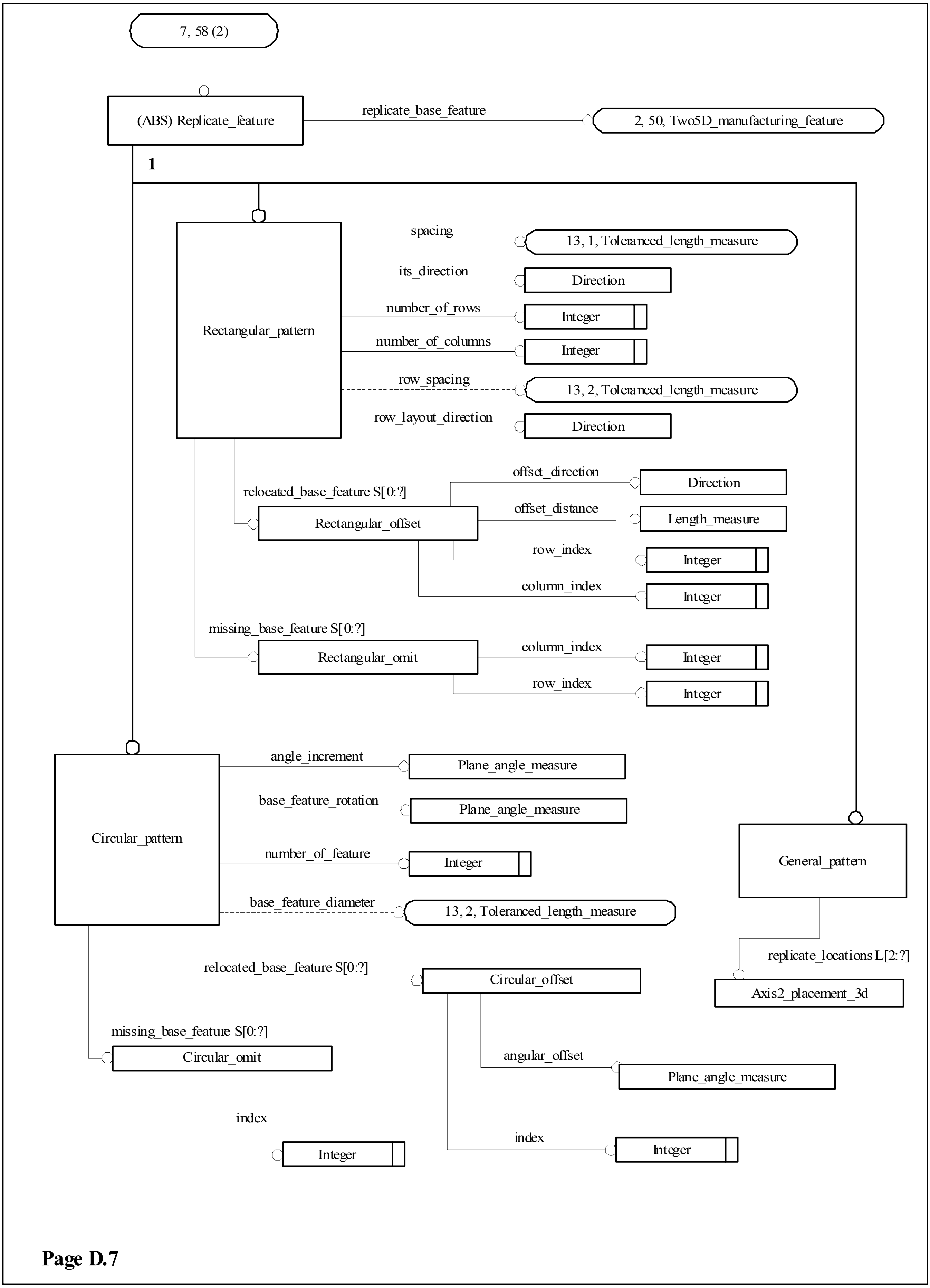

3, 51 (2)
Closed_pocket
1
feature_boundary
5, 83, Closed_profile
(ABS)Pocket
Open_pocket
planar_radius
open_boundary
13, 1, Toleranced_length_measure
4, 82, Open_profile
orthogonal_radius
wall_boundary
13, 1, Toleranced_length_measure
4, 82, Open_profile
3, 52 (2)
its_boundary
5, 83, Closed_profile
its_boss L[0:?]
Boss
slope
Plane_angle_measure
slope
Plane_angle_measure
bottom_condition
1
(ABS)Pocket_bottom_conditoion
Through_pocket_bottom_condition
Planar_pocket_bottom_condition
Cartesian_point
floor_radius_center
Radiused_pocket_bottom_condition
floor_radius
13, 2, Toleranced_length_measure
3, 53 (2)
shape
General_pocket_bottom_condition
1, 47, Region
course_of_travel
Slot
15, 34, Travel_path
swept_shape
4, 82, Open_profile
end_conditions S[0:2]
(ABS)Slot_end_type
1
Open_slot_end_type
Loop_slot_end_type
corner_radius1
Flat_slot_end_type
13, 1, Toleranced_length_measure
corner_radius2
13, 1, Toleranced_length_measure
Radiused_slot_end_type
radius
Woodruff_slot_end_type
13, 1, Toleranced_length_measure
Page D.3


placement
Axis2_placement_3d
(ABS)Profile
4, 38 (2,5)
4, 82 (3)
1
5, 83, Closed_profile
4, 29 (2)
Linear_profile
profile_length
13, 30, Numeric_parameter
width
13, 1, Toleranced_length_measure
first_radius
13, 1, Toleranced_length_measure
Square_U_profile
first_angle
Plane_angle_measure
second_radius
13, 1, Toleranced_length_measure
4, 40 (5)
second_angle
Plane_angle_measure
Rounded_U_profile
width
13, 1, Toleranced_length_measure
first_angle
Plane_angle_measure
second_angle
Plane_angle_measure
1
(ABS)Open_profile
cross_bar_width
13, 1, Toleranced_length_measure
Tee_profile
cross_bar_depth
13, 1, Toleranced_length_measure
radius
13, 1, Toleranced_length_measure
width
13, 1, Toleranced_length_measure
first_offset
13, 1, Toleranced_length_measure
4, 33 (2)
second_offset
13, 1, Toleranced_length_measure
profile_radius
13, 1, Toleranced_length_measure
Vee_profile
profile_angle
Plane_angle_measure
tilt_angle
Plane_angle_measure
4, 32 (2,5)
radius
13, 1, Toleranced_length_measure
Partial_circular_profile
sweep_angle
Plane_angle_measure
General_profile
its_profile
Bounded_curve
Page D.4

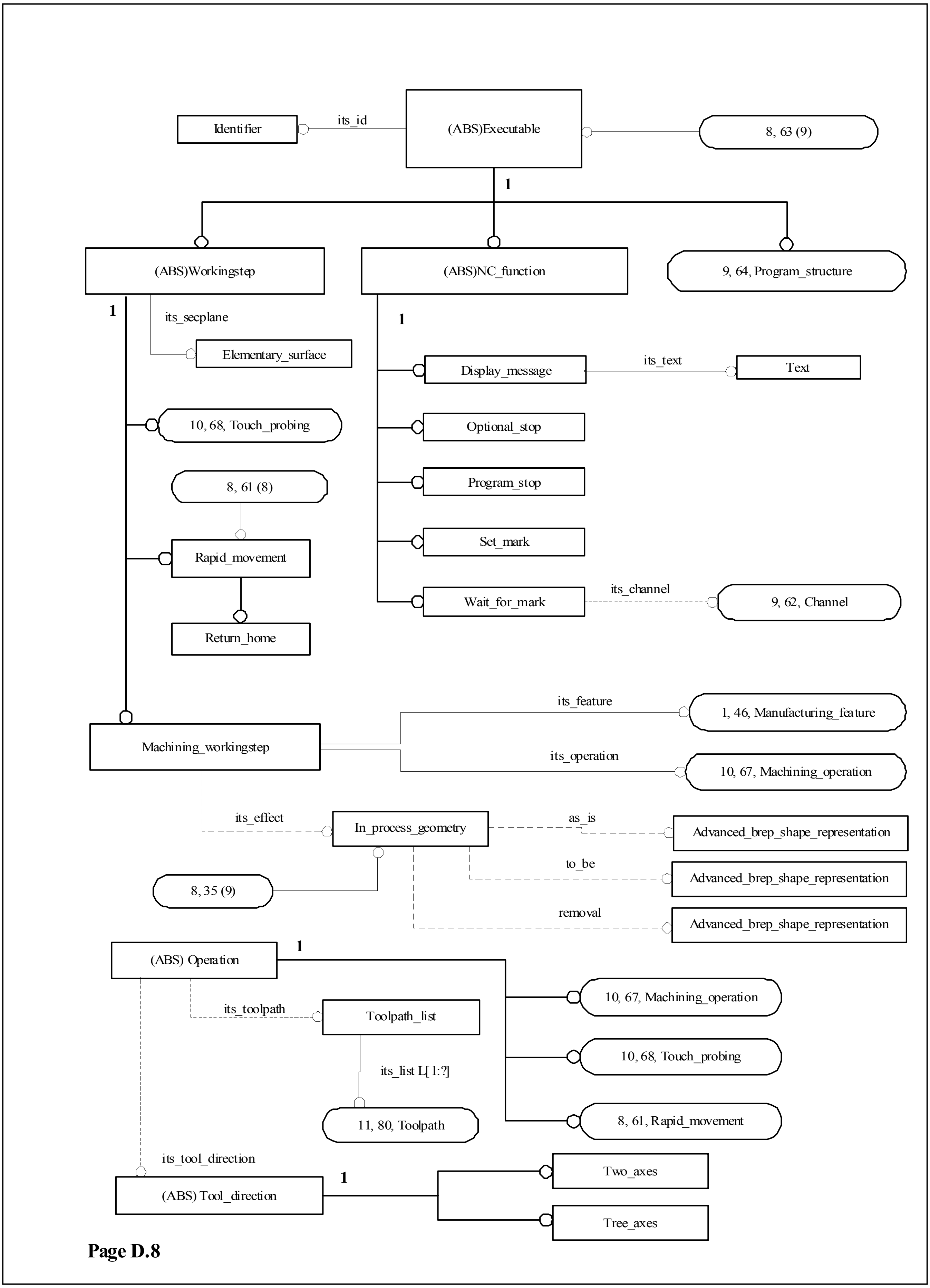

5, 83 (2,3,4)
5, 41 (5)
Rectangular_closed_profile
profile_width
12, 1, Toleranced_length_measure
profile_length
12, 1, Toleranced_length_measure
5, 39 (5)
1
Circular_closed_profile
diameter
12, 1, Toleranced_length_measure
diameter
12, 1, Toleranced_length_measure
(ABS)Closed_profile
Ngon_profile
number_of_sides
Integer
circumscribed_or_across_flats
Boolean
General_closed_profile
closed_profile_shape
Bounded_curve
5, 84 (2)
General_shape_profile
profile_boundary
4, 38, Profile
Partial_circular_shape_profile
open_boundary
4, 32, Partial_circular_profile
Shape_profile
1
Circular_closed_shape_profile
closed_boundary
5, 39, Circular_closed_profile
Rectangular_open_shape_profile
open_boundary
4, 40, Square_u_profile
Rectangular_closed_shape_profile
closed_boundary
5, 41, Rectangular_closed_profile
Through_profile_floor
General_profile_floor
floor
Face
floor_condition
Profile_select
1
(ABS)Profile_floor
Planar_profile_floor
floor
Plane
floor_radius
13, 30, Numeric_parameter
removal_direction
Direction
start_or_end
Boolean
Page D.5


6, 55 (2)
Round_hole
bottom_condition
diameter
13, 1, Toleranced_length_measure
(ABS)Hole_bottom_condition
change_in_diameter
1
Taper_select
(ABS)Blind_bottom_condition
Through_bottom_condition
Diameter_taper
Angle_taper
1
final_diameter
angle
Flat_hole_bottom
13, 1, Toleranced_length_measure
Plane_angle_measure
Flat_with_radius_hole_bottom
corner_radius
13, 1, Toleranced_length_measure
Spherical_hole_bottom
radius
13, 1, Toleranced_length_measure
Conical_hole_bottom
tip_angle
Plane_angle_measure
tip_radius
13, 2, Toleranced_length_measure
6, 57 (1,2)
first_feature
2, 56, Machining_feature
(ABS)Transition_feature
second_feature
2, 56, Machining_feature
1
Chamfer
Edge_round
angle_to_plane
Plane_angle_measure
radius
13, 1, Toleranced_length_measure
first_offset_amount
first_offset_amount
13, 1, Toleranced_length_measure
13, 2, Toleranced_length_measure
second_offset_amount
13, 2, Toleranced_length_measure

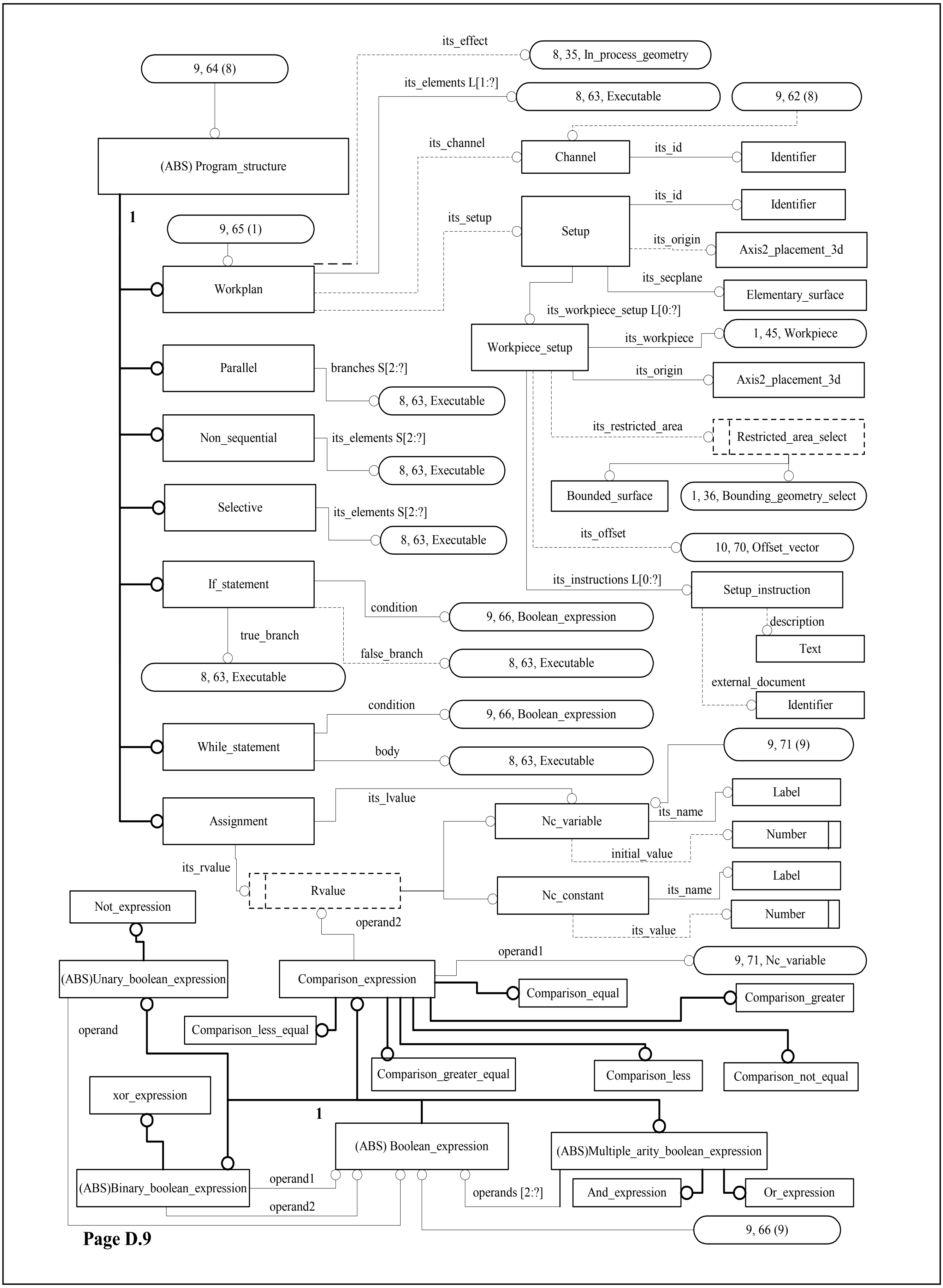
Page D.6

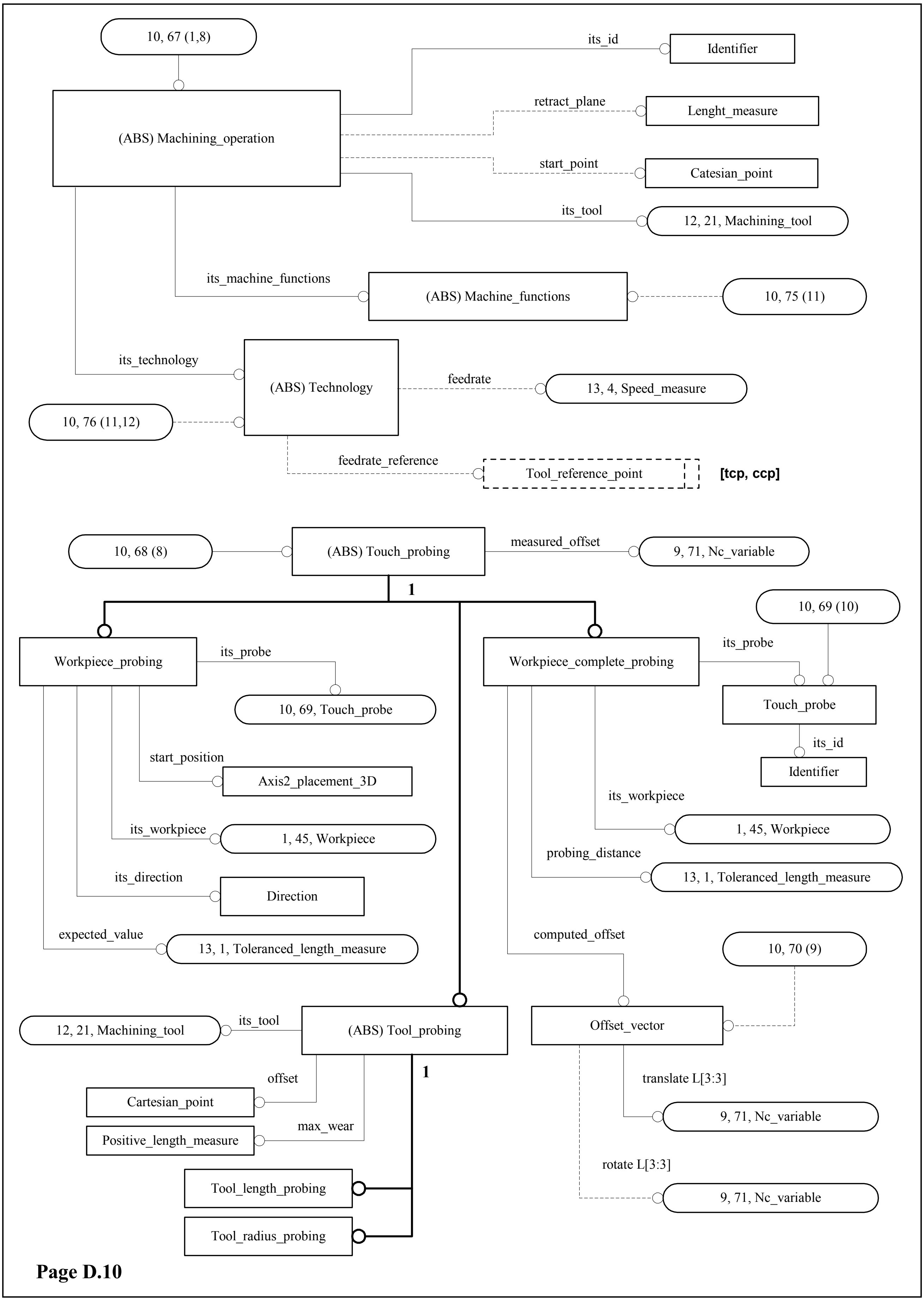

7, 58 (2)
(ABS) Replicate_feature
replicate_base_feature
2, 50, Two5D_manufacturing_feature
1
Rectangular_pattern
spacing
13, 1, Toleranced_length_measure
its_direction
Direction
number_of_rows
Integer
number_of_columns
Integer
row_spacing
13, 2, Toleranced_length_measure
row_layout_direction
Direction
relocated_base_feature S[0:?]
Rectangular_offset
offset_direction
Direction
offset_distance
Length_measure
row_index
Integer
column_index
Integer
missing_base_feature S[0:?]
Rectangular_omit
column_index
Integer
row_index
Integer
Circular_pattern
angle_increment
Plane_angle_measure
base_feature_rotation
Plane_angle_measure
number_of_feature
Integer
base_feature_diameter
13, 2, Toleranced_length_measure
General_pattern
replicate_locations L[2:?]
Axis2_placement_3d
relocated_base_feature S[0:?]
Circular_offset
missing_base_feature S[0:?]
Circular_omit
angular_offset
Plane_angle_measure
index
Integer
index
Integer

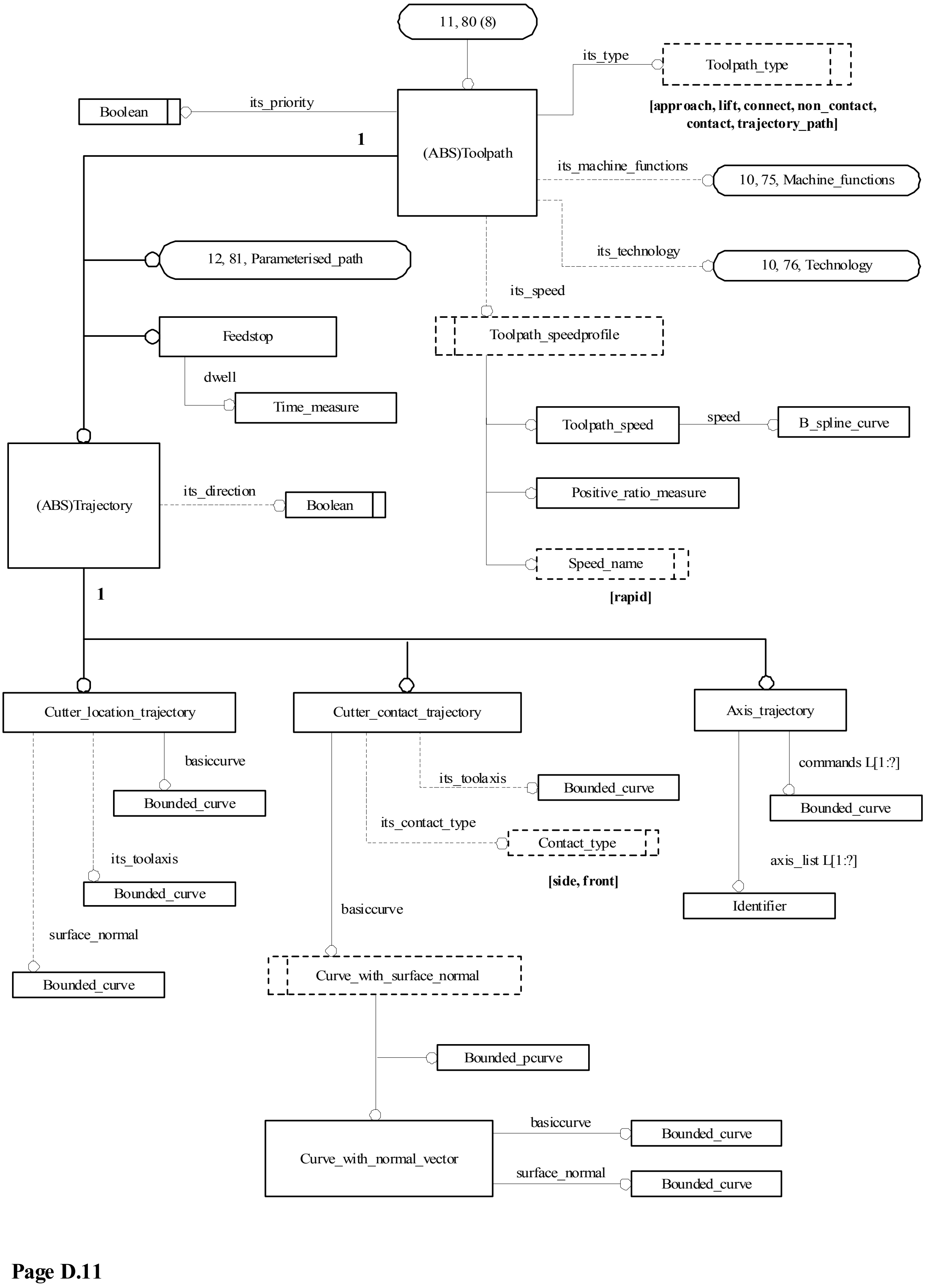
Page D.7

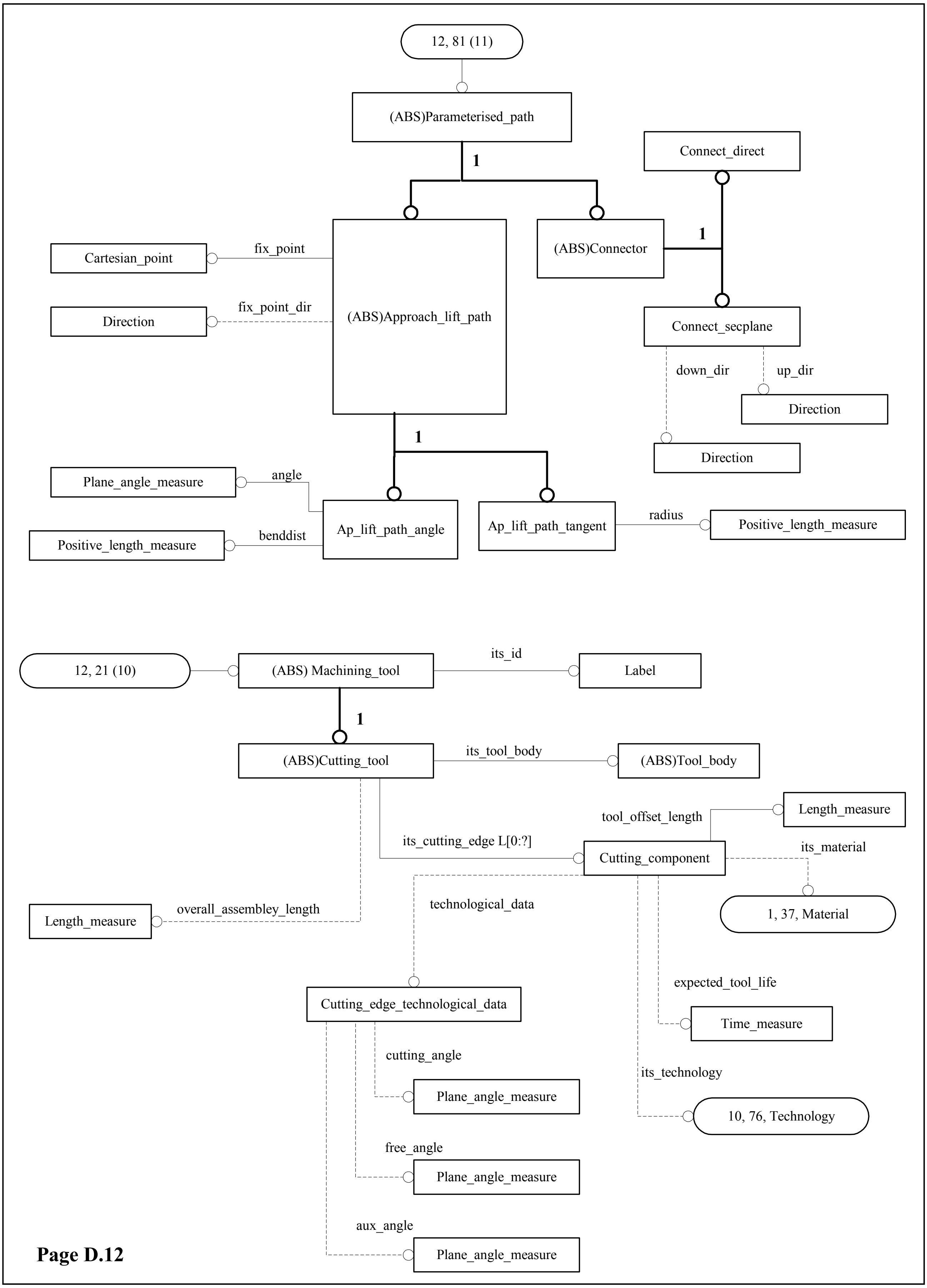

Identifier
its_id
(ABS)Executable
8, 63 (9)
1
(ABS)Workingstep
(ABS)NC_function
9, 64, Program_structure
1
its_secplane
Elementary_surface
1
Display_message
its_text
Text
10, 68, Touch_probing
Optional_stop
8, 61 (8)
Program_stop
Set_mark
Rapid_movement
Wait_for_mark
its_channel
9, 62, Channel
Return_home
its_feature
1, 46, Manufacturing_feature
Machining_workingstep
its_operation
10, 67, Machining_operation
its_effect
In_process_geometry
as_is
Advanced_brep_shape_representation
to_be
Advanced_brep_shape_representation
8, 35 (9)
removal
Advanced_brep_shape_representation
(ABS) Operation
1
10, 67, Machining_operation
its_toolpath
Toolpath_list
10, 68, Touch_probing
its_list L[1:?]
11, 80, Toolpath
8, 61, Rapid_movement
its_tool_direction
1
Two_axes
(ABS) Tool_direction
Tree_axes

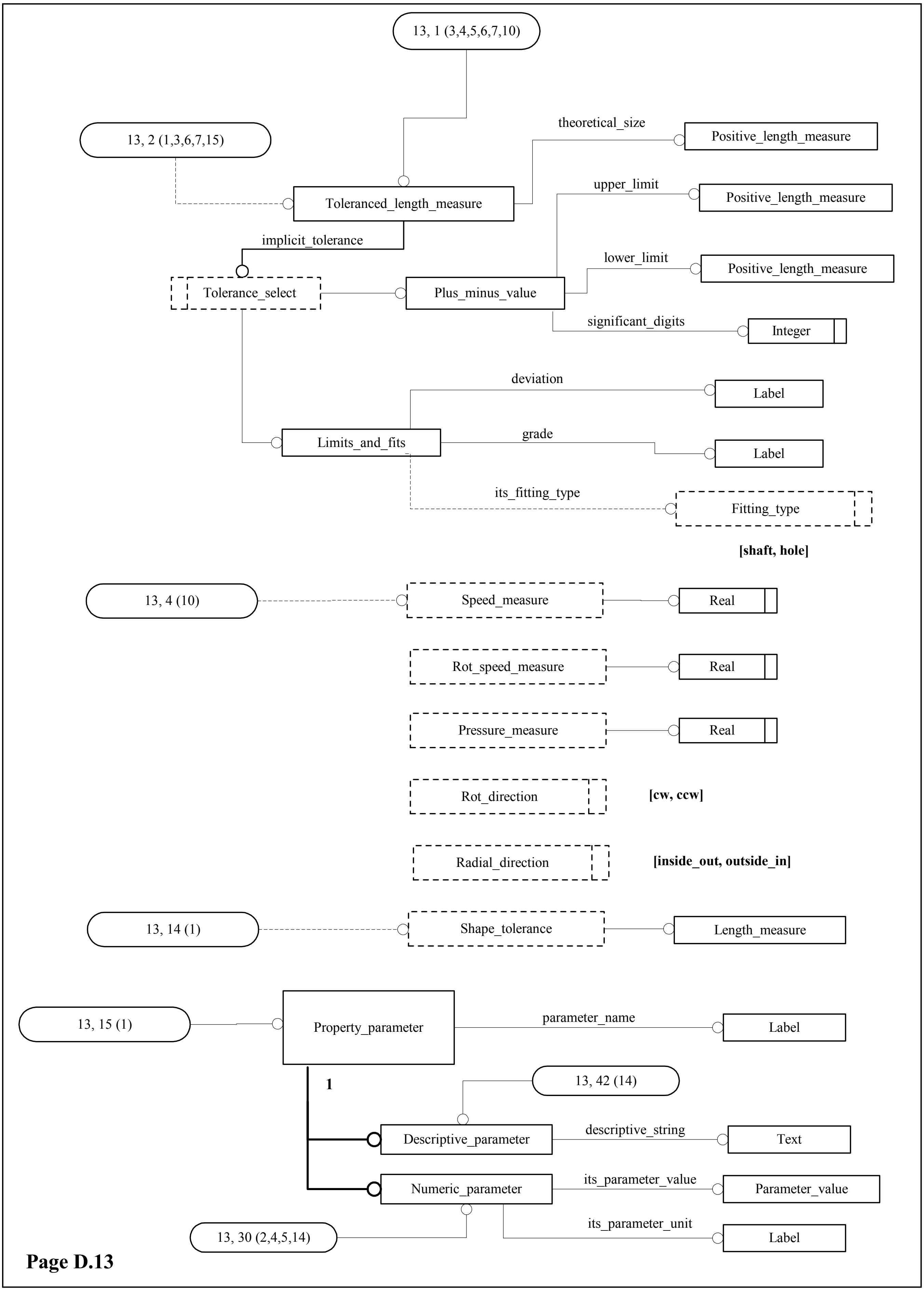
Page D.8


its_effect
8, 35, In_process_geometry
9, 64 (8)
its_elements L[1:?]
8, 63, Executable
9, 62 (8)
its_channel
Channel
its_id
Identifier
(ABS) Program_structure
its_id
Identifier
1
9, 65 (1)
its_setup
Setup
its_origin
Axis2_placement_3d
its_secplane
Elementary_surface
Workplan
its_workpiece_setup L[0:?]
Workpiece_setup
its_workpiece
1, 45, Workpiece
Parallel
branches S[2:?]
its_origin
Axis2_placement_3d
8, 63, Executable
Non_sequential
its_elements S[2:?]
its_restricted_area
Restricted_area_select
8, 63, Executable
Bounded_surface
1, 36, Bounding_geometry_select
Selective
its_elements S[2:?]
8, 63, Executable
its_offset
10, 70, Offset_vector
If_statement
its_instructions L[0:?]
Setup_instruction
condition
9, 66, Boolean_expression
description
true_branch
Text
false_branch
8, 63, Executable
8, 63, Executable
external_document
condition
Identifier
9, 66, Boolean_expression
While_statement
body
8, 63, Executable
9, 71 (9)
its_lvalue
Label
Assignment
Nc_variable
its_name
Number
initial_value
its_rvalue
Label
Rvalue
Nc_constant
its_name
Not_expression
Number
operand2
its_value
operand1
9, 71, Nc_variable
(ABS)Unary_boolean_expression
Comparison_expression
Comparison_equal
Comparison_greater
operand
Comparison_less_equal
Comparison_greater_equal
Comparison_less
Comparison_not_equal
xor_expression
1
(ABS) Boolean_expression
(ABS)Multiple_arity_boolean_expression
operand1
(ABS)Binary_boolean_expression
operands [2:?]
And_expression
Or_expression
operand2
9, 66 (9)
Page D.9

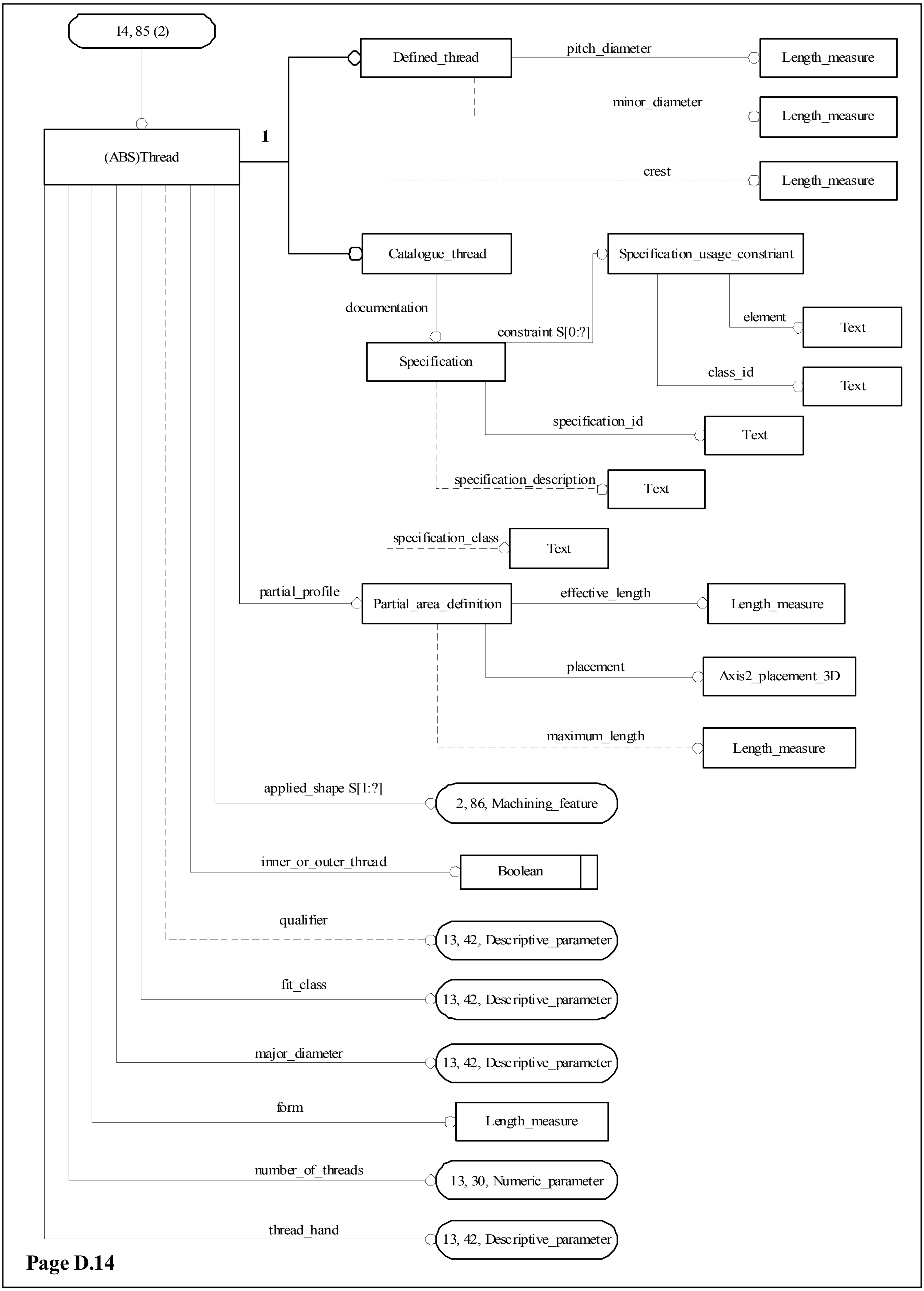

10, 67 (1,8)
(ABS) Machining_operation
its_id
Identifier
retract_plane
Lenght_measure
start_point
Catesian_point
its_tool
12, 21, Machining_tool
its_machine_functions
(ABS) Machine_functions
10, 75 (11)
its_technology
(ABS) Technology
feedrate
13, 4, Speed_measure
10, 76 (11,12)
feedrate_reference
Tool_reference_point
[tcp, ccp]
10, 68 (8)
(ABS) Touch_probing
measured_offset
9, 71, Nc_variable
1
10, 69 (10)
Workpiece_probing
its_probe
10, 69, Touch_probe
Workpiece_complete_probing
its_probe
Touch_probe
its_id
Identifier
start_position
Axis2_placement_3D
its_workpiece
1, 45, Workpiece
its_workpiece
1, 45, Workpiece
probing_distance
13, 1, Toleranced_length_measure
its_direction
Direction
expected_value
13, 1, Toleranced_length_measure
computed_offset
10, 70 (9)
12, 21, Machining_tool
its_tool
(ABS) Tool_probing
Offset_vector
1
offset
Cartesian_point
translate L[3:3]
9, 71, Nc_variable
max_wear
Positive_length_measure
rotate L[3:3]
9, 71, Nc_variable
Tool_length_probing
Tool_radius_probing
Page D.10

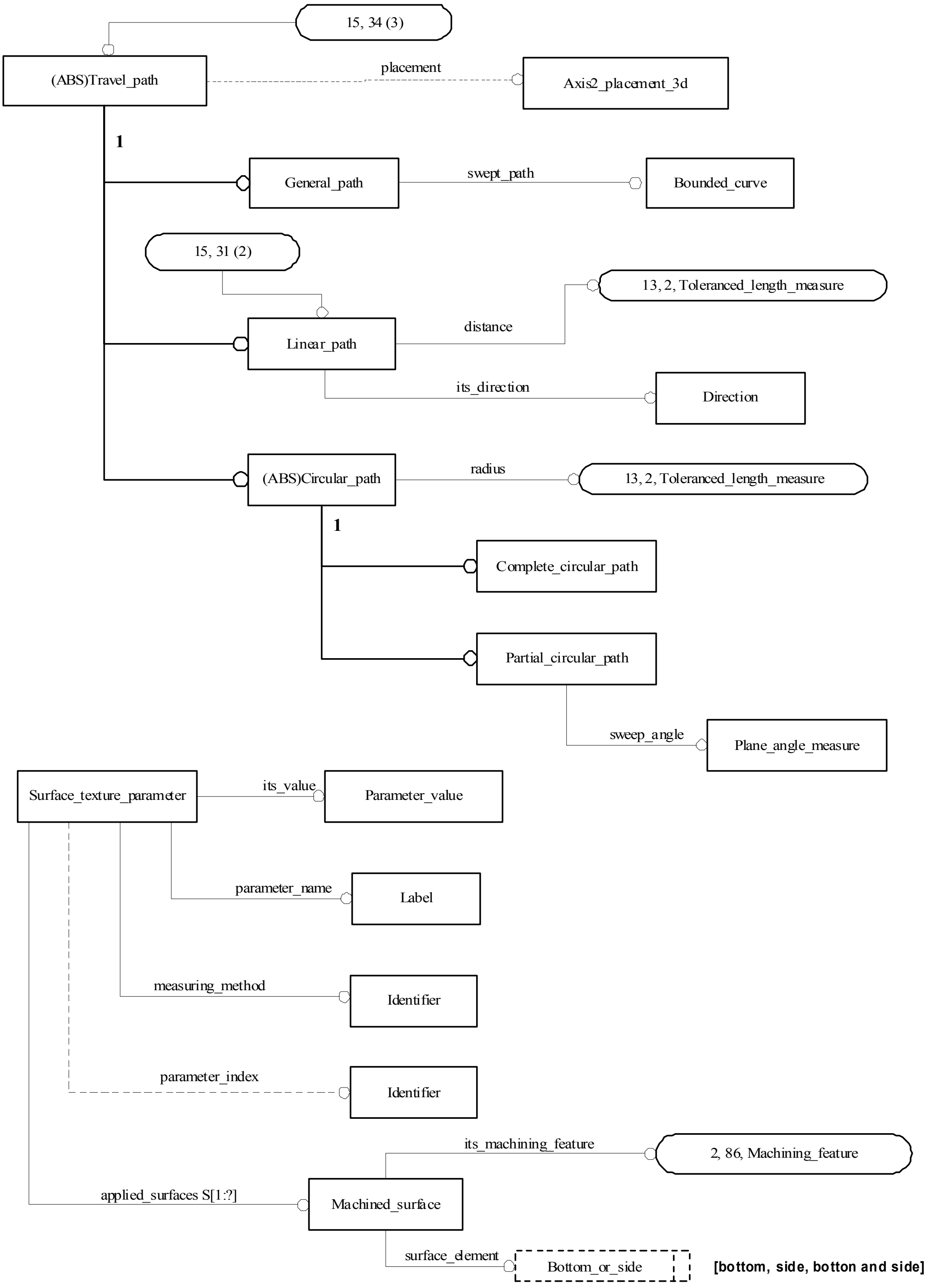

11, 80 (8)
its_type
Toolpath_type
[approach, lift, connect, non_contact, contact, trajectory_path]
Boolean
its_priority
1
(ABS)Toolpath
its_machine_functions
10, 75, Machine_functions
its_technology
10, 76, Technology
12, 81, Parameterised_path
its_speed
Toolpath_speedprofile
Feedstop
dwell
Time_measure
Toolpath_speed
speed
B_spline_curve
(ABS)Trajectory
its_direction
Boolean
Positive_ratio_measure
Speed_name
[rapid]
1
Cutter_location_trajectory
Cutter_contact_trajectory
Axis_trajectory
basiccurve
Bounded_curve
its_toolaxis
Bounded_curve
commands L[1:?]
Bounded_curve
its_contact_type
Contact_type
[side, front]
axis_list L[1:?]
Identifier
its_toolaxis
Bounded_curve
surface_normal
Bounded_curve
basiccurve
Curve_with_surface_normal
Bounded_pcurve
Curve_with_normal_vector
basiccurve
Bounded_curve
surface_normal
Bounded_curve
Page D.11


12, 81 (11)
(ABS)Parameterised_path
1
Connect_direct
(ABS)Connector
1
Cartesian_point
fix_point
Direction
fix_point_dir
(ABS)Approach_lift_path
Connect_secplane
down_dir
up_dir
Direction
Direction
1
Plane_angle_measure
angle
Ap_lift_path_angle
Ap_lift_path_tangent
radius
Positive_length_measure
Positive_length_measure
benddist
12, 21 (10)
(ABS) Machining_tool
its_id
Label
1
(ABS)Cutting_tool
its_tool_body
(ABS)Tool_body
tool_offset_length
Length_measure
its_cutting_edge L[0:?]
Cutting_component
its_material
1, 37, Material
Length_measure
overall_assembley_length
technological_data
Cutting_edge_technological_data
expected_tool_life
Time_measure
cutting_angle
Plane_angle_measure
its_technology
10, 76, Technology
free_angle
Plane_angle_measure
aux_angle
Plane_angle_measure
Page D.12


13, 1 (3,4,5,6,7,10)
13, 2 (1,3,6,7,15)
Toleranced_length_measure
theoretical_size
Positive_length_measure
upper_limit
Positive_length_measure
lower_limit
Positive_length_measure
implicit_tolerance
Tolerance_select
Plus_minus_value
significant_digits
Integer
deviation
Label
Limits_and_fits
grade
Label
its_fitting_type
Fitting_type
[shaft, hole]
13, 4 (10)
Speed_measure
Real
Rot_speed_measure
Real
Pressure_measure
Real
Rot_direction
[cw, ccw]
Radial_direction
[inside_out, outside_in]
13, 14 (1)
Shape_tolerance
Length_measure
13, 15 (1)
Property_parameter
parameter_name
Label
1
13, 42 (14)
Descriptive_parameter
descriptive_string
Text
Numeric_parameter
its_parameter_value
Parameter_value
its_parameter_unit
Label
13, 30 (2,4,5,14)

Page D.13


14, 85 (2)
(ABS)Thread
1
Defined_thread
pitch_diameter
Length_measure
minor_diameter
Length_measure
crest
Length_measure
Catalogue_thread
Specification_usage_constriant
documentation
element
Text
constraint S[0:?]
Specification
class_id
Text
specification_id
Text
specification_description
Text
specification_class
Text
partial_profile
Partial_area_definition
effective_length
Length_measure
placement
Axis2_placement_3D
maximum_length
Length_measure
applied_shape S[1:?]
2, 86, Machining_feature
inner_or_outer_thread
Boolean
qualifier
13, 42, Descriptive_parameter
fit_class
13, 42, Descriptive_parameter
major_diameter
13, 42, Descriptive_parameter
form
Length_measure
number_of_threads
13, 30, Numeric_parameter
thread_hand
13, 42, Descriptive_parameter
Page D.14


15, 34 (3)
(ABS)Travel_path
placement
Axis2_placement_3d
1
General_path
swept_path
Bounded_curve
15, 31 (2)
13, 2, Toleranced_length_measure
distance
Linear_path
its_direction
Direction
(ABS)Circular_path
radius
13, 2, Toleranced_length_measure
1
Complete_circular_path
Partial_circular_path
sweep_angle
Plane_angle_measure
Surface_texture_parameter
its_value
Parameter_value
parameter_name
Label
measuring_method
Identifier
parameter_index
Identifier
its_machining_feature
2, 86, Machining_feature
applied_surfaces S[1:?]
Machined_surface
surface_element
Bottom_or_side
[bottom, side, botton and side]

Page D.15

Annex E
(informative)

Computer-interpretable listings

This annex provides a listing of the short names with their corresponding entity names and a listing of the EXPRESS specified in this Part of ISO 14649. No text or annotation is included. This annex is provided only in computer-interpretable form.

INDEX