ISO TC184/SC4/3 /P9 N447

Date: 12 Septenber 1995 **Supersedes SC4/ 3 N 399 (P9)**

PRODUCT DATA REPRESENTATION AND EXCHANGE

Part: 104 **Title:** STEP Part 104: Integrated application resources: Finite element analysis

Purpose of this document as it relates to the target document is:

X Primary Content
___ Issue Discussion
___ Alternate Proposal
___ Partial Content

Current Status: ISO CD

ABSTRACT:
This document provides the integrated resources to represent a finite element model and the associated analysis control and output.

KEYWORDS:
Finite element analysis

Document Status/Dates (dd/mm/yy)

Part Documents		Other SC4 Documents	
26/6/95	Project Leader	___	Author
___	Working Group	___	PPC
29/6/95	Qualification	___	SC4
27/6/95	Integration		
28/6/95	Editorially Complete		
___	ISO Committee Draft		

Owner/Editor: Keith A. Hunten
Address: Lockheed Martin
Tactical Aircraft Systems
PO BOX 748 MZ 2824
Fort Worth TX 76101
USA
Telephone/FAX: 817 777 2147/2115
E-mail: HuntenKA@LFWC.Lockheed.com

Alternate:
Address:
Telephone/FAX:
E-mail:

Comments to Reader
This second CD version reflects the resolution of all CD1 issues, AP209 integration issues, and review issues through June 1995.

Contents Page

1 Scope . . . 1
1.1 Structural response definition schema scope . . . 3
1.2 Structural response representation schema scope . . . 3
1.3 Finite element analysis control and result schema scope . . . 3
1.4 Analysis type scope . . . 4
1.5 Scalars, vectors, and tensors schema scope . . . 4

2 Normative references . . . 4

3 Definitions, symbols, and abbreviations . . . 6
3.1 Terms defined in ISO 10303-1 . . . 6
3.2 Definitions and abbreviations . . . 6
3.2.1 3d_model . . . 6
3.2.2 2d_model . . . 6
3.2.3 FEA . . . 6
3.2.4 DOF . . . 6
3.3 Symbols . . . 7

4 Structural response definition schema . . . 8
4.1 Introduction . . . 8
4.2 Fundamental concepts and assumptions . . . 8
4.3 Structural response definition schema entity definitions: Structural response property . . . 10
4.3.1 structural_response_property . . . 10
4.4 Structural response definition schema entity definitions: Finite element analysis model definition . . . 10
4.4.1 fea_model_definition . . . 10
4.5 Structural response definition schema entity definitions: Nodes . . . 10
4.5.1 node_definition . . . 10
4.6 Structural response definition schema entity definitions: Elements . . . 11
4.6.1 element_definition . . . 11

5 Structural response representation schema . . . 12
5.1 Introduction . . . 14
5.2 Fundamental concepts and assumptions . . . 14
5.2.1 Product relationship . . . 15
5.2.2 Finite element analysis model relationships . . . 15
5.2.3 Geometric founding and analysis space . . . 15
5.2.4 Identifiers . . . 15
5.2.5 Coordinate systems . . . 15
5.2.6 Element matrix integration . . . 15
5.2.7 Units and measures . . . 15
5.2.8 Interface with the materials schema . . . 16

5.2.9 Finite element analysis model, control, and result relationships 16
5.3 Structural response representation schema type definitions 17
5.3.1 axi_or_plane 17
5.3.2 coordinate_system_type 17
5.3.3 element_order 21
5.3.4 plane_2d_element_assumption 21
5.3.5 volume_element_purpose 22
5.3.6 surface_element_purpose 22
5.3.7 curve_element_purpose 23
5.3.8 volume_3d_element_shape 24
5.3.9 element_2d_shape 24
5.3.10 matrix_property_type 24
5.3.11 surface_matrix_property_type 25
5.3.12 curve_matrix_property_type 26
5.3.13 matrix_symmetry 27
5.3.14 degree_of_freedom 28
5.3.15 curve_element_freedom 29
5.3.16 integration_rule 30
5.3.17 shape_function 30
5.3.18 additional_node 30
5.3.19 dummy_node 31
5.3.20 volume_2d_element_representation 31
5.3.21 surface_2d_element_representation 31
5.3.22 curve_2d_element_representation 31
5.3.23 volume_2d_element_descriptor 32
5.3.24 surface_2d_element_descriptor 32
5.3.25 curve_2d_element_descriptor 32
5.3.26 volume_3d_element_coordinate_system 33
5.3.27 volume_2d_element_coordinate_system 33
5.3.28 surface_3d_element_coordinate_system 33
5.3.29 surface_2d_element_coordinate_system 33
5.3.30 curve_3d_element_coordinate_system 34
5.3.31 curve_element_end_coordinate_system 34
5.3.32 element_aspect 34
5.3.33 element_volume 35
5.3.34 curve_edge 35
5.4 Structural response representation schema entity definitions: Finite element analysis model 36
5.4.1 fea_model 36
5.4.2 fea_model_3d 36
5.4.3 fea_model_2d 37
5.4.4 structural_response_property_definition_representation 37
5.4.5 fea_representation_item 38
5.5 Structural response representation schema entity definitions: Nodal geometry and geometric coordinate systems 39
5.5.1 direction_node 41

5.5.2 fea_axis2_placement_2d . . . 41
5.5.3 fea_axis2_placement_3d . . . 42
5.5.4 fea_representation_relationship . . . 42
5.5.5 cylindrical_point . . . 43
5.5.6 spherical_point . . . 43
5.6 Structural response representation schema entity definitions: Node representation 44
5.6.1 node_representation . . . 44
5.6.2 node_geometric_relationship . . . 45
5.6.3 node . . . 45
5.6.4 node_with_solution_coordinate_system . . . 46
5.6.5 node_with_vector . . . 46
5.6.6 geometric_node . . . 47
5.6.7 node_set . . . 47
5.7 Structural response representation schema entity definitions: Element representations . . . 48
5.7.1 element_representation . . . 53
5.7.2 higher_order_element_representation . . . 54
5.7.3 element_geometric_relationship . . . 54
5.7.4 volume_3d_element_representation . . . 55
5.7.5 axisymmetric_volume_2d_element_representation . . . 58
5.7.6 plane_volume_2d_element_representation . . . 60
5.7.7 surface_3d_element_representation . . . 63
5.7.8 axisymmetric_surface_2d_element_representation . . . 66
5.7.9 plane_surface_2d_element_representation . . . 68
5.7.10 curve_3d_element_representation . . . 71
5.7.11 axisymmetric_curve_2d_element_representation . . . 74
5.7.12 plane_curve_2d_element_representation . . . 75
5.7.13 element_descriptor . . . 77
5.7.14 volume_3d_element_descriptor . . . 78
5.7.15 axisymmetric_volume_2d_element_descriptor . . . 78
5.7.16 plane_volume_2d_element_descriptor . . . 79
5.7.17 volume_3d_element_basis . . . 79
5.7.18 volume_2d_element_basis . . . 80
5.7.19 surface_3d_element_descriptor . . . 80
5.7.20 axisymmetric_surface_2d_element_descriptor . . . 81
5.7.21 plane_surface_2d_element_descriptor . . . 81
5.7.22 surface_3d_element_basis . . . 82
5.7.23 surface_2d_element_basis . . . 82
5.7.24 curve_3d_element_descriptor . . . 83
5.7.25 axisymmetric_curve_2d_element_descriptor . . . 83
5.7.26 plane_curve_2d_element_descriptor . . . 84
5.7.27 curve_3d_element_basis . . . 84
5.7.28 curve_2d_element_basis . . . 85
5.7.29 point_element_representation . . . 85
5.7.30 point_element_matrix . . . 86
5.7.31 stationary_mass . . . 87

5.7.32 grounded_spring . . . 88
5.7.33 grounded_damper . . . 88
5.7.34 directionally_explicit_element_representation . . . 89
5.7.35 system_and_freedom . . . 91
5.7.36 directionally_explicit_element_coefficient . . . 91
5.7.37 explicit_element_representation . . . 91
5.7.38 explicit_element_matrix . . . 93
5.8 Structural response representation schema definitions: Element topologies . . . 93
5.9 Structural response representation schema entity definitions: Element coordinate systems . . . 118
5.9.1 aligned_axis_tolerance . . . 118
5.9.2 arbitrary_volume_3d_element_coordinate_system . . . 119
5.9.3 parametric_volume_3d_element_coordinate_system . . . 119
5.9.4 arbitrary_volume_2d_element_coordinate_system . . . 122
5.9.5 parametric_volume_2d_element_coordinate_system . . . 122
5.9.6 aligned_surface_3d_element_coordinate_system . . . 125
5.9.7 parametric_surface_3d_element_coordinate_system . . . 127
5.9.8 constant_surface_3d_element_coordinate_system . . . 129
5.9.9 aligned_surface_2d_element_coordinate_system . . . 129
5.9.10 parametric_surface_2d_element_coordinate_system . . . 131
5.9.11 aligned_curve_3d_element_coordinate_system . . . 132
5.9.12 parametric_curve_3d_element_coordinate_system . . . 132
5.9.13 parametric_curve_3d_element_coordinate_direction . . . 135
5.9.14 curve_2d_element_coordinate_system . . . 135
5.9.15 directionally_explicit_element_coordinate_system . . . 137
5.9.16 directionally_explicit_element_coordinate_system_arbitrary . . . 137
5.9.17 directionally_explicit_element_coordinate_system_aligned . . . 137
5.9.18 euler_angles . . . 138
5.10 Structural response representation schema entity definitions: Element matrix integration . . . 138
5.10.1 Volume 3D Element . . . 143
5.10.2 Volume 2D Element . . . 143
5.10.3 Curve 2D Element . . . 144
5.10.4 Surface 3D or Curve Element . . . 144
5.10.5 Surface 2D Element . . . 145
5.10.6 volume_3d_element_integrated_matrix . . . 145
5.10.7 volume_3d_element_integrated_matrix_with_definition . . . 146
5.10.8 volume_3d_element_field_integration . . . 146
5.10.9 element_integration_algebraic . . . 146
5.10.10 volume_3d_element_field_integration_rule . . . 147
5.10.11 volume_3d_element_field_integration_explicit . . . 147
5.10.12 volume_position_weight . . . 147
5.10.13 volume_2d_element_integrated_matrix . . . 148
5.10.14 volume_2d_element_integrated_matrix_with_definition . . . 148
5.10.15 volume_2d_element_field_integration . . . 149
5.10.16 volume_2d_element_field_integration_rule . . . 149

5.10.17 volume_2d_element_field_integration_explicit 149
5.10.18 surface_3d_element_integrated_matrix 150
5.10.19 surface_3d_element_integrated_matrix_with_definition 150
5.10.20 surface_3d_element_integration 150
5.10.21 surface_3d_element_field_integration 151
5.10.22 surface_section_integration 151
5.10.23 surface_3d_element_field_integration_rule 151
5.10.24 surface_3d_element_field_integration_explicit 152
5.10.25 surface_position_weight 152
5.10.26 surface_section_integration_rule 152
5.10.27 surface_section_integration_explicit 153
5.10.28 surface_section_position_weight 153
5.10.29 surface_2d_element_integrated_matrix 154
5.10.30 surface_2d_element_integrated_matrix_with_definition 154
5.10.31 surface_2d_element_integration 154
5.10.32 surface_2d_element_length_integration 155
5.10.33 surface_2d_element_length_integration_rule 155
5.10.34 surface_2d_element_length_integration_explicit 155
5.10.35 curve_3d_element_integrated_matrix 156
5.10.36 curve_3d_element_integrated_matrix_with_definition 156
5.10.37 curve_3d_element_integration 156
5.10.38 curve_3d_element_length_integration 157
5.10.39 curve_3d_element_length_integration_rule 157
5.10.40 curve_3d_element_length_integration_explicit 158
5.10.41 curve_3d_element_position_weight 158
5.10.42 curve_section_integration_explicit 158
5.10.43 curve_2d_element_integrated_matrix 159
5.10.44 curve_2d_element_integrated_matrix_with_definition 159
5.10.45 curve_2d_element_integration 159
5.11 Structural response representation schema entity definitions: Element locations 160
5.11.1 fea_parametric_point 160
5.11.2 volume_element_location 161
5.11.3 surface_volume_element_location 161
5.11.4 surface_element_location 161
5.11.5 surface_section_element_location 162
5.11.6 surface_section_element_location_absolute 162
5.11.7 surface_section_element_location_dimensionless 163
5.11.8 curve_volume_element_location 163
5.11.9 curve_element_location 163
5.11.10 curve_section_element_location 164
5.12 Structural response representation schema entity definitions: Finite element analysis material properties 164
5.12.1 element_material 166
5.12.2 fea_material_property_representation 166
5.12.3 fea_material_property_representation_item 167
5.12.4 fea_linear_elasticity 168

5.12.5 fea_mass_density . . . 168
5.12.6 fea_area_density . . . 169
5.12.7 fea_tangential_coefficient_of_linear_thermal_expansion . . . 169
5.12.8 fea_secant_coefficient_of_linear_thermal_expansion . . . 170
5.12.9 fea_moisture_absorption . . . 171
5.12.10 fea_shell_membrane_stiffness . . . 171
5.12.11 fea_shell_bending_stiffness . . . 172
5.12.12 fea_shell_membrane_bending_coupling_stiffness . . . 173
5.12.13 fea_shell_shear_stiffness . . . 174
5.13 Structural response representation schema entity definitions: Element properties 175
5.13.1 Surface element properties . . . 175
5.13.2 Curve element properties . . . 180
5.13.3 surface_element_property . . . 183
5.13.4 surface_section_field . . . 186
5.13.5 surface_section_field_constant . . . 186
5.13.6 surface_section_field_varying . . . 186
5.13.7 surface_section . . . 187
5.13.8 uniform_surface_section . . . 187
5.13.9 uniform_surface_section_layered . . . 188
5.13.10 curve_3d_element_property . . . 188
5.13.11 curve_element_interval . . . 189
5.13.12 curve_element_interval_constant . . . 190
5.13.13 curve_element_interval_linearly_varying . . . 190
5.13.14 curve_2d_element_property . . . 191
5.13.15 curve_element_section_definition . . . 191
5.13.16 curve_element_section_derived_definitions . . . 192
5.13.17 curve_element_end_offset . . . 193
5.13.18 curve_element_end_release . . . 194
5.13.19 curve_element_end_release_packet . . . 194
5.13.20 axisymmetric_2d_element_property . . . 195
5.13.21 plane_2d_element_property . . . 195
5.13.22 simple_plane_2d_element_property . . . 195
5.14 Structural response representation schema entity definitions: Groups . . . 196
5.14.1 fea_group . . . 196
5.14.2 element_group . . . 196
5.14.3 node_group . . . 196
5.14.4 fea_group_relation . . . 197
5.14.5 volume_3d_element_group . . . 197
5.14.6 volume_2d_element_group . . . 198
5.14.7 surface_3d_element_group . . . 198
5.14.8 surface_2d_element_group . . . 198
5.14.9 curve_3d_element_group . . . 199
5.14.10 curve_2d_element_group . . . 199
5.15 Structural response representation schema function definitions . . . 200
5.15.1 required_0d_nodes . . . 200
5.15.2 required_1d_nodes . . . 200

5.15.3 required_2d_nodes . . . 201
5.15.4 required_3d_nodes . . . 202
5.15.5 additional_0d_nodes . . . 203
5.15.6 additional_1d_nodes . . . 203
5.15.7 additional_2d_nodes . . . 204
5.15.8 additional_3d_nodes . . . 205
5.15.9 number_of_terms . . . 206
5.15.10 valid_parametric_coordinate . . . 207
5.15.11 build_direction_node . . . 208
5.15.12 map_to_common_cartesian . . . 209
5.15.13 consistent_geometric_reference . . . 209
5.15.14 consistent_element_reference . . . 212

6 Finite element analysis control and result schema . . . 214
6.1 Introduction . . . 216
6.2 Fundamental concepts and assumptions . . . 216
6.2.1 Control . . . 216
6.2.2 Process . . . 216
6.2.3 Result . . . 216
6.2.4 State . . . 217
6.2.5 Step . . . 217
6.2.6 State definition . . . 217
6.2.7 Units and measures . . . 218
6.3 Finite element analysis control and result schema type definitions . . . 219
6.3.1 model_or_control_element . . . 219
6.3.2 cylindrical_harmonic_number . . . 219
6.3.3 volume_3d_face . . . 219
6.3.4 volume_2d_face . . . 220
6.3.5 volume_3d_edge . . . 220
6.3.6 volume_2d_edge . . . 221
6.3.7 surface_3d_face . . . 221
6.3.8 surface_3d_edge . . . 221
6.3.9 surface_2d_face . . . 222
6.3.10 surface_2d_edge . . . 222
6.3.11 field_value . . . 223
6.3.12 unspecified_value . . . 223
6.3.13 measure_or_unspecified_value . . . 223
6.3.14 boundary_variable . . . 223
6.3.15 boundary_aggregated_variable . . . 224
6.3.16 volume_variable . . . 224
6.3.17 volume_aggregated_variable . . . 224
6.3.18 surface_element_variable . . . 225
6.3.19 boundary_edge_variable . . . 225
6.3.20 curve_element_variable . . . 225
6.3.21 curve_scalar_variable . . . 226
6.3.22 surface_scalar_variable . . . 227

6.3.23 volume_scalar_variable . . . 228
6.3.24 boundary_curve_scalar_variable . . . 228
6.3.25 boundary_surface_scalar_variable . . . 228
6.3.26 aggregated_scalar_variable . . . 229
6.3.27 volume_angular_variable . . . 229
6.3.28 aggregated_angular_variable . . . 230
6.3.29 user_defined_scalar_variable . . . 230
6.3.30 curve_vector_2d_variable . . . 230
6.3.31 surface_vector_2d_variable . . . 232
6.3.32 user_defined_vector_2d_variable . . . 234
6.3.33 curve_vector_3d_variable . . . 234
6.3.34 surface_vector_3d_variable . . . 234
6.3.35 volume_vector_3d_variable . . . 235
6.3.36 boundary_curve_vector_3d_variable . . . 235
6.3.37 boundary_surface_vector_3d_variable . . . 236
6.3.38 aggregated_vector_3d_variable . . . 236
6.3.39 user_defined_vector_3d_variable . . . 237
6.3.40 surface_tensor2_2d_variable . . . 237
6.3.41 user_defined_tensor2_2d_variable . . . 239
6.3.42 volume_tensor2_3d_variable . . . 239
6.3.43 aggregated_tensor2_3d_variable . . . 240
6.3.44 user_defined_tensor2_3d_variable . . . 241
6.3.45 message_level . . . 241
6.3.46 surface_3d_state_coordinate_system . . . 242
6.3.47 surface_2d_state_coordinate_system . . . 242
6.3.48 curve_3d_state_coordinate_system . . . 242
6.3.49 curve_2d_state_coordinate_system . . . 242
6.3.50 action_type . . . 243
6.3.51 volume_3d_element_output_reference . . . 243
6.3.52 volume_2d_element_output_reference . . . 243
6.3.53 surface_3d_element_output_reference . . . 244
6.3.54 surface_2d_element_output_reference . . . 244
6.3.55 curve_3d_element_output_reference . . . 244
6.3.56 curve_2d_element_output_reference . . . 245
6.3.57 node_output_reference . . . 245
6.4 Finite element analysis control and result entity definitions: Analysis control . 246
6.4.1 control . . . 246
6.4.2 analysis_step . . . 246
6.4.3 control_analysis_step . . . 247
6.4.4 symmetry_control . . . 247
6.4.5 no_symmetry_control . . . 248
6.4.6 cylindrical_symmetry_control . . . 248
6.4.7 control_linear_static_analysis_step . . . 248
6.4.8 control_linear_static_analysis_step_with_harmonic . . . 249
6.4.9 constraint_element . . . 249
6.4.10 single_point_constraint_element . . . 250

6.4.11 linear_constraint_equation_element . . . 251
6.4.12 linear_constraint_equation_nodal_term . . . 251
6.4.13 freedom_and_coefficient . . . 252
6.4.14 point_constraint . . . 252
6.4.15 curve_constraint . . . 253
6.4.16 surface_constraint . . . 254
6.4.17 solid_constraint . . . 254
6.4.18 control_process . . . 255
6.4.19 control_static_load_increment_process . . . 255
6.4.20 element_sequence . . . 256
6.4.21 node_sequence . . . 257
6.5 Finite element analysis control and result entity definitions: Analysis results . 257
6.5.1 result . . . 257
6.5.2 result_analysis_step . . . 258
6.5.3 result_linear_static_analysis_sub_step . . . 258
6.5.4 control_result_relationship . . . 259
6.6 Finite element analysis control and result entity definitions: Model state . . . 259
6.6.1 state . . . 259
6.6.2 state_with_harmonic . . . 260
6.6.3 specified_state . . . 260
6.6.4 calculated_state . . . 260
6.6.5 linearly_superimposed_state . . . 260
6.6.6 state_component . . . 261
6.6.7 output_request_state . . . 261
6.7 Finite element analysis control and result entity definitions: State definition . . 262
6.7.1 state_definition . . . 264
6.7.2 field_variable_definition . . . 264
6.7.3 field_variable_element_definition . . . 265
6.7.4 volume_3d_element_field_variable_definition . . . 265
6.7.5 volume_3d_element_location_point_variable_values . . . 266
6.7.6 volume_3d_element_value_and_location . . . 266
6.7.7 volume_3d_whole_element_variable_value . . . 267
6.7.8 volume_3d_element_constant_specified_variable_value . . . 268
6.7.9 volume_3d_element_nodal_specified_variable_values . . . 268
6.7.10 volume_3d_element_boundary_location_point_variable_values . . . 269
6.7.11 volume_3d_element_boundary_whole_face_variable_value . . . 270
6.7.12 volume_3d_element_boundary_constant_specified_variable_value . . . 271
6.7.13 volume_3d_element_boundary_nodal_specified_variable_values . . . 272
6.7.14 volume_3d_element_boundary_edge_location_point_volume_variable_values 273
6.7.15 volume_3d_element_boundary_edge_whole_edge_variable_value . . . 274
6.7.16 volume_3d_element_boundary_edge_constant_specified_volume_variable_-value . . . 274
6.7.17 volume_3d_element_boundary_edge_nodal_specified_variable_values . . . 275
6.7.18 volume_2d_element_field_variable_definition . . . 276
6.7.19 volume_2d_element_location_point_variable_values . . . 277
6.7.20 volume_2d_element_value_and_location . . . 278

6.7.21 volume_2d_whole_element_variable_value 278
6.7.22 volume_2d_element_constant_specified_variable_value 279
6.7.23 volume_2d_element_nodal_specified_variable_values 279
6.7.24 volume_2d_element_boundary_location_point_variable_values 280
6.7.25 volume_2d_element_boundary_whole_face_variable_value 281
6.7.26 volume_2d_element_boundary_constant_specified_variable_value 282
6.7.27 volume_2d_element_boundary_nodal_specified_variable_values 283
6.7.28 volume_2d_element_boundary_edge_location_point_volume_variable_values 284
6.7.29 volume_2d_element_boundary_edge_whole_edge_variable_value 285
6.7.30 volume_2d_element_boundary_edge_constant_specified_volume_variable_-value 286
6.7.31 volume_2d_element_boundary_edge_nodal_specified_variable_values 286
6.7.32 surface_3d_element_field_variable_definition 287
6.7.33 surface_3d_element_location_point_volume_variable_values 288
6.7.34 surface_3d_element_value_and_location 289
6.7.35 surface_3d_element_value_and_volume_location 289
6.7.36 surface_3d_element_location_point_variable_values 290
6.7.37 surface_3d_whole_element_variable_value 291
6.7.38 surface_3d_element_constant_specified_variable_value 291
6.7.39 surface_3d_element_constant_specified_volume_variable_value 292
6.7.40 surface_3d_element_nodal_specified_variable_values 293
6.7.41 surface_3d_element_boundary_location_point_surface_variable_values 293
6.7.42 surface_3d_element_boundary_whole_face_variable_value 294
6.7.43 surface_3d_element_boundary_constant_specified_surface_variable_value . . 295
6.7.44 surface_3d_element_boundary_constant_specified_variable_value 296
6.7.45 surface_3d_element_boundary_nodal_specified_variable_values 297
6.7.46 surface_3d_element_boundary_edge_location_point_surface_variable_values . 298
6.7.47 surface_3d_element_boundary_edge_location_point_variable_values 299
6.7.48 surface_3d_element_boundary_edge_whole_edge_variable_value 300
6.7.49 surface_3d_element_boundary_edge_constant_specified_surface_variable_value 300
6.7.50 surface_3d_element_boundary_edge_constant_specified_variable_value . . . 301
6.7.51 surface_3d_element_boundary_edge_nodal_specified_variable_values 302
6.7.52 surface_2d_element_field_variable_definition 303
6.7.53 surface_2d_element_location_point_volume_variable_values 304
6.7.54 surface_2d_element_value_and_location 305
6.7.55 surface_2d_element_value_and_volume_location 305
6.7.56 surface_2d_element_location_point_variable_values 306
6.7.57 surface_2d_whole_element_variable_value 306
6.7.58 surface_2d_element_constant_specified_variable_value 307
6.7.59 surface_2d_element_constant_specified_volume_variable_value 308
6.7.60 surface_2d_element_nodal_specified_variable_values 308
6.7.61 surface_2d_element_boundary_location_point_surface_variable_values 309
6.7.62 surface_2d_element_boundary_whole_face_variable_value 310
6.7.63 surface_2d_element_boundary_constant_specified_surface_variable_value . . 311
6.7.64 surface_2d_element_boundary_constant_specified_variable_value 312
6.7.65 surface_2d_element_boundary_nodal_specified_variable_values 313

6.7.66 surface_2d_element_boundary_edge_location_point_surface_variable_values . 313
6.7.67 surface_2d_element_boundary_edge_location_point_variable_values 314
6.7.68 surface_2d_element_boundary_edge_whole_edge_variable_value 315
6.7.69 surface_2d_element_boundary_edge_constant_specified_surface_variable_value316
6.7.70 surface_2d_element_boundary_edge_constant_specified_variable_value . . . 317
6.7.71 surface_2d_element_boundary_edge_nodal_specified_variable_values 318
6.7.72 curve_3d_element_field_variable_definition 319
6.7.73 curve_3d_element_location_point_volume_variable_values 319
6.7.74 curve_3d_element_value_and_location 320
6.7.75 curve_3d_element_value_and_volume_location 320
6.7.76 curve_3d_element_location_point_variable_values 321
6.7.77 curve_3d_whole_element_variable_value 322
6.7.78 curve_3d_element_constant_specified_variable_value 322
6.7.79 curve_3d_element_constant_specified_volume_variable_value 323
6.7.80 curve_3d_element_nodal_specified_variable_values 324
6.7.81 curve_2d_element_field_variable_definition 325
6.7.82 curve_2d_element_location_point_volume_variable_values 325
6.7.83 curve_2d_element_value_and_location 326
6.7.84 curve_2d_element_value_and_volume_location 326
6.7.85 curve_2d_element_location_point_variable_value 327
6.7.86 curve_2d_whole_element_variable_value 327
6.7.87 curve_2d_element_constant_specified_variable_value 328
6.7.88 curve_2d_element_constant_specified_volume_variable_value 329
6.7.89 field_variable_element_group_value 329
6.7.90 field_variable_whole_model_value 330
6.7.91 field_variable_node_definition 331
6.7.92 volume_3d_node_field_variable_definition 332
6.7.93 volume_2d_node_field_variable_definition 332
6.7.94 surface_3d_node_field_variable_definition 333
6.7.95 surface_3d_node_field_section_variable_values 333
6.7.96 surface_3d_node_field_aggregated_variable_values 334
6.7.97 surface_2d_node_field_variable_definition 335
6.7.98 surface_2d_node_field_section_variable_values 335
6.7.99 surface_2d_node_field_aggregated_variable_values 336
6.7.100 curve_3d_node_field_variable_definition 336
6.7.101 curve_3d_node_field_section_variable_values 337
6.7.102 curve_3d_node_field_aggregated_variable_values 337
6.7.103 curve_2d_node_field_variable_definition 338
6.7.104 curve_2d_node_field_section_variable_values 338
6.7.105 curve_2d_node_field_aggregated_variable_values 339
6.7.106 nodal_freedom_and_value_definition 340
6.7.107 nodal_freedom_values 341
6.7.108 nodal_freedom_action_definition 341
6.7.109 element_nodal_freedom_actions 341
6.7.110 element_nodal_freedom_terms 342
6.7.111 curve_freedom_and_value_definition 342

6.7.112 curve_freedom_values . . . 343
6.7.113 curve_freedom_action_definition . . . 343
6.7.114 point_freedom_and_value_definition . . . 344
6.7.115 point_freedom_values . . . 344
6.7.116 point_freedom_action_definition . . . 345
6.7.117 surface_freedom_and_value_definition . . . 345
6.7.118 surface_freedom_values . . . 346
6.7.119 surface_freedom_action_definition . . . 346
6.7.120 solid_freedom_and_value_definition . . . 346
6.7.121 solid_freedom_values . . . 347
6.7.122 solid_freedom_action_definition . . . 347
6.7.123 freedom_and_value . . . 348
6.7.124 linear_constraint_equation_element_value . . . 348
6.7.125 single_point_constraint_element_values . . . 348
6.7.126 analysis_message . . . 349
6.7.127 whole_model_analysis_message . . . 349
6.7.128 element_analysis_message . . . 350
6.7.129 node_analysis_message . . . 350
6.7.130 element_group_analysis_message . . . 350
6.8 Finite element analysis control and result schema function definitions . . . 351
6.8.1 necessary_freedom_coordinate_system . . . 351
6.8.2 necessary_freedom_coordinate_system_set . . . 352
6.8.3 necessary_value_coordinate_system . . . 353
6.8.4 consistent_set_values . . . 354
6.8.5 consistent_list_values . . . 355
6.8.6 consistent_value . . . 355
6.8.7 variable_value_type . . . 356
6.8.8 appropriate_set_value_existence . . . 359
6.8.9 appropriate_list_value_existence . . . 359
6.8.10 appropriate_value_existence . . . 360

7 FEA scalar vector tensor schema . . . 362
7.1 Introduction . . . 362
7.2 Fundamental concepts and assumptions . . . 362
7.3 Scalar vector tensor schema type definitions: The FEA representation of scalars, vectors, and tensors . . . 362
7.3.1 angular_value . . . 362
7.3.2 scalar . . . 363
7.3.3 tensor1 . . . 363
7.3.4 tensor1_2d . . . 363
7.3.5 tensor1_3d . . . 363
7.3.6 symmetric_tensor2_2d . . . 364
7.3.7 anisotropic_symmetric_tensor2_2d . . . 364
7.3.8 symmetric_tensor2_3d . . . 365
7.3.9 isotropic_symmetric_tensor2_3d . . . 365
7.3.10 orthotropic_symmetric_tensor2_3d . . . 365

7.3.11 anisotropic_symmetric_tensor2_3d . . . 366
7.3.12 symmetric_tensor4_2d . . . 366
7.3.13 anisotropic_symmetric_tensor4_2d . . . 367
7.4 Scalar vector tensor schema type definitions: The FEA representation of a fourth order material response tensor . . . 367
7.4.1 symmetric_tensor4_3d . . . 368
7.4.2 anisotropic_symmetric_tensor4_3d . . . 369
7.4.3 fea_isotropic_symmetric_tensor4_3d . . . 370
7.4.4 fea_iso_orthotropic_symmetric_tensor4_3d . . . 371
7.4.5 fea_transverse_isotropic_symmetric_tensor4_3d . . . 372
7.4.6 fea_column_normalised_orthotropic_symmetric_tensor4_3d . . . 373
7.4.7 fea_column_normalised_monoclinic_symmetric_tensor4_3d . . . 375

Annexes

A Short names of entities . . . 378
B EXPRESS listing . . . 379
C Finite element multi-disciplinary and nonlinear analysis extensions . . . 380
D EXPRESS-G Diagrams . . . 383

Index . . . 469

Figures

1 FEA schema level model in EXPRESS-G . . . 2
2 FEA information model high level relationships EXPRESS-G partial model . . . 9
3 Cartesian coordinate system definition . . . 18
4 Cylindrical coordinate system definition . . . 19
5 Spherical coordinate system definition . . . 20
6 EXPRESS-G partial model illustrating finite element analysis relationships with geometry and representation . . . 40
7 Element subtyping . . . 50
8 Volume element properties EXPRESS-G partial model . . . 51
9 Point, directionally explicit and explicit elements EXPRESS-G partial model . . . 52
10 Curve 3D or surface 2D elements - linear . . . 95
11 Curve 3D or surface 2D elements - quadratic . . . 95
12 Curve 3D or surface 2D elements - cubic . . . 96
13 Surface 2D elements - faces and edges . . . 97
14 Surface 3D or volume 2D elements - triangle - linear . . . 98
15 Surface 3D or volume 2D elements - triangle - quadratic . . . 98
16 Surface 3D or volume 2D elements - triangle - cubic . . . 99

17 Surface 3D elements - triangle faces and edges 100
18 Volume 2D elements - triangle faces 101
19 Surface 3D or volume 2D elements - quadrilateral - linear 102
20 Surface 3D or volume 2D elements - quadrilateral - quadratic 103
21 Surface 3D or volume 2D elements - quadrilateral - cubic 103
22 Surface 3D elements - quadrilateral faces and edges 104
23 Volume 2D elements - quadrilateral faces 105
24 Volume 3D elements - hexahedron - linear 106
25 Volume 3D elements - hexahedron - quadratic 106
26 Volume 3D elements - hexahedron - cubic 107
27 Volume 3D elements - hexahedron faces 108
28 Volume 3D elements - wedge - linear 109
29 Volume 3D elements - wedge - quadratic 109
30 Volume 3D elements - wedge - cubic 110
31 Volume 3D elements - wedge faces 111
32 Volume 3D elements - tetrahedron - linear 112
33 Volume 3D elements - tetrahedron - quadratic 112
34 Volume 3D elements - tetrahedron - cubic 113
35 Volume 3D elements - tetrahedron faces 114
36 Volume 3D elements - pyramid - linear 115
37 Volume 3D elements - pyramid - quadratic 115
38 Volume 3D elements - pyramid - cubic 116
39 Volume 3D elements - pyramid faces 117
40 Orthogonal volume 3D element coordinate system derivation for axis_1 = 1 and axis_2 = 2 121
41 Arbitrary volume 2D element coordinate system orientation 123
42 Parametric volume 2D element coordinate system derivation for axis = 1 124
43 Aligned surface 3D element coordinate system derivation 126
44 Parametric surface 3D element coordinate system derivation for axis = 1 128
45 Aligned surface 2D element coordinate system derivation 130
46 Parametric surface 2D element coordinate system derivation 131

47 Aligned curve 3D element coordinate system derivation 133
48 Parametric curve 3D element coordinate system derivation 134
49 Curve 2D element coordinate system derivation 136
50 Volume element integration points 139
51 Surface element integration points 140
52 Volume element integration EXPRESS-G partial model 141
53 Surface and curve element integration EXPRESS-G partial model 142
54 Relationship between FEA and ISO 10303-45 165
55 Surface element properties EXPRESS-G partial model 179
56 Curve element end offsets 180
57 Curve element property coordinate system origin orientation 181
58 Curve element interval specification conventions 182
59 Curve element end release and end offset 184
60 Curve element properties EXPRESS-G partial model 185
61 Finite element analysis control EXPRESS-G partial model 218
62 State definition EXPRESS-G partial model 263
C.1 Multi-disciplinary analysis environment 380
D.1 Structural response definition schema EXPRESS-G diagram, 1 of 85. 384
D.2 Structural response representation schema EXPRESS-G diagram, 2 of 85. 385
D.3 Structural response representation schema EXPRESS-G diagram, 3 of 85. 386
D.4 Structural response representation schema EXPRESS-G diagram, 4 of 85. 387
D.5 Structural response representation schema EXPRESS-G diagram, 5 of 85. 388
D.6 Structural response representation schema EXPRESS-G diagram, 6 of 85. 389
D.7 Structural response representation schema EXPRESS-G diagram, 7 of 85. 390
D.8 Structural response representation schema EXPRESS-G diagram, 8 of 85. 391
D.9 Structural response representation schema EXPRESS-G diagram, 9 of 85. 392
D.10 Structural response representation schema EXPRESS-G diagram, 10 of 85. 393
D.11 Structural response representation schema EXPRESS-G diagram, 11 of 85. 394
D.12 Structural response representation schema EXPRESS-G diagram, 12 of 85. 395
D.13 Structural response representation schema EXPRESS-G diagram, 13 of 85. 396
D.14 Structural response representation schema EXPRESS-G diagram, 14 of 85. 397

D.15 Structural response representation schema EXPRESS-G diagram, 15 of 85. 398
D.16 Structural response representation schema EXPRESS-G diagram, 16 of 85. 399
D.17 Structural response representation schema EXPRESS-G diagram, 17 of 85. 400
D.18 Structural response representation schema EXPRESS-G diagram, 18 of 85. 401
D.19 Structural response representation schema EXPRESS-G diagram, 19 of 85. 402
D.20 Structural response representation schema EXPRESS-G diagram, 20 of 85. 403
D.21 Structural response representation schema EXPRESS-G diagram, 21 of 85. 404
D.22 Structural response representation schema EXPRESS-G diagram, 22 of 85. 405
D.23 Structural response representation schema EXPRESS-G diagram, 23 of 85. 406
D.24 Structural response representation schema EXPRESS-G diagram, 24 of 85. 407
D.25 Structural response representation schema EXPRESS-G diagram, 25 of 85. 408
D.26 Finite element analysis and control schema EXPRESS-G diagram, 26 of 85. 409
D.27 Finite element analysis and control schema EXPRESS-G diagram, 27 of 85. 410
D.28 Finite element analysis and control schema EXPRESS-G diagram, 28 of 85. 411
D.29 Finite element analysis and control schema EXPRESS-G diagram, 29 of 85. 412
D.30 Finite element analysis and control schema EXPRESS-G diagram, 30 of 85. 413
D.31 Finite element analysis and control schema EXPRESS-G diagram, 31 of 85. 414
D.32 Finite element analysis and control schema EXPRESS-G diagram, 32 of 85. 415
D.33 Finite element analysis and control schema EXPRESS-G diagram, 33 of 85. 416
D.34 Finite element analysis and control schema EXPRESS-G diagram, 34 of 85. 417
D.35 Finite element analysis and control schema EXPRESS-G diagram, 35 of 85. 418
D.36 Finite element analysis and control schema EXPRESS-G diagram, 36 of 85. 419
D.37 Finite element analysis and control schema EXPRESS-G diagram, 37 of 85. 420
D.38 Finite element analysis and control schema EXPRESS-G diagram, 38 of 85. 421
D.39 Finite element analysis and control schema EXPRESS-G diagram, 39 of 85. 422
D.40 Finite element analysis and control schema EXPRESS-G diagram, 40 of 85. 423
D.41 Finite element analysis and control schema EXPRESS-G diagram, 41 of 85. 424
D.42 Finite element analysis and control schema EXPRESS-G diagram, 42 of 85. 425
D.43 Finite element analysis and control schema EXPRESS-G diagram, 43 of 85. 426
D.44 Finite element analysis and control schema EXPRESS-G diagram, 44 of 85. 427
D.45 Finite element analysis and control schema EXPRESS-G diagram, 45 of 85. 428

D.46 Finite element analysis and control schema EXPRESS-G diagram, 46 of 85. 429
D.47 Finite element analysis and control schema EXPRESS-G diagram, 47 of 85. 430
D.48 Finite element analysis and control schema EXPRESS-G diagram, 48 of 85. 431
D.49 Finite element analysis and control schema EXPRESS-G diagram, 49 of 85. 432
D.50 Finite element analysis and control schema EXPRESS-G diagram, 50 of 85. 433
D.51 Finite element analysis and control schema EXPRESS-G diagram, 51 of 85. 434
D.52 Finite element analysis and control schema EXPRESS-G diagram, 52 of 85. 435
D.53 Finite element analysis and control schema EXPRESS-G diagram, 53 of 85. 436
D.54 Finite element analysis and control schema EXPRESS-G diagram, 54 of 85. 437
D.55 Finite element analysis and control schema EXPRESS-G diagram, 55 of 85. 438
D.56 Finite element analysis and control schema EXPRESS-G diagram, 56 of 85. 439
D.57 Finite element analysis and control schema EXPRESS-G diagram, 57 of 85. 440
D.58 Finite element analysis and control schema EXPRESS-G diagram, 58 of 85. 441
D.59 Finite element analysis and control schema EXPRESS-G diagram, 59 of 85. 442
D.60 Finite element analysis and control schema EXPRESS-G diagram, 60 of 85. 443
D.61 Finite element analysis and control schema EXPRESS-G diagram, 61 of 85. 444
D.62 Finite element analysis and control schema EXPRESS-G diagram, 62 of 85. 445
D.63 Finite element analysis and control schema EXPRESS-G diagram, 63 of 85. 446
D.64 Finite element analysis and control schema EXPRESS-G diagram, 64 of 85. 447
D.65 Finite element analysis and control schema EXPRESS-G diagram, 65 of 85. 448
D.66 Finite element analysis and control schema EXPRESS-G diagram, 66 of 85. 449
D.67 Finite element analysis and control schema EXPRESS-G diagram, 67 of 85. 450
D.68 Finite element analysis and control schema EXPRESS-G diagram, 68 of 85. 451
D.69 Finite element analysis and control schema EXPRESS-G diagram, 69 of 85. 452
D.70 Structural response representation schema EXPRESS-G diagram, 70 of 85. 453
D.71 Structural response representation schema EXPRESS-G diagram, 71 of 85. 454
D.72 Structural response representation schema EXPRESS-G diagram, 72 of 85. 455
D.73 Structural response representation schema EXPRESS-G diagram, 73 of 85. 456
D.74 Structural response representation schema EXPRESS-G diagram, 74 of 85. 457
D.75 Structural response representation schema EXPRESS-G diagram, 75 of 85. 458
D.76 Structural response representation schema EXPRESS-G diagram, 76 of 85. 459

D.77 Finite element analysis and control schema EXPRESS-G diagram, 77 of 85. 460
D.78 Finite element analysis and control schema EXPRESS-G diagram, 78 of 85. 461
D.79 Finite element analysis and control schema EXPRESS-G diagram, 79 of 85. 462
D.80 Finite element analysis and control schema EXPRESS-G diagram, 80 of 85. 463
D.81 Finite element analysis and control schema EXPRESS-G diagram, 81 of 85. 464
D.82 Finite element analysis and control schema EXPRESS-G diagram, 82 of 85. 465
D.83 Finite element analysis and control schema EXPRESS-G diagram, 83 of 85. 466
D.84 Finite element analysis and control schema EXPRESS-G diagram, 84 of 85. 467
D.85 Scalar vector tensor schema EXPRESS-G diagram, 85 of 85. 468

Tables

1 Element coordinate system definition symbols 7
2 Element coordinate axis symbols 7

Foreword

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

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

International Standard ISO 10303–104 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Subcommittee SC4, *Industrial data.* ISO 10303 consists of the following parts under the general title *Industrial automation systems and integration – Product data representation and exchange:*

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

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

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

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

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

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

Introduction

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

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

This part of ISO 10303 is concerned with information exchange needs of finite element analysis. Many types of analyses can be conducted to ensure the performance and integrity of a product. Different aspects of a product may be idealized and then analyzed as a continuum. Exact mathematical models for any but the simplest continuum shapes are intractable. Therefore analytical methods that represent the continuum as discrete tractable shapes are used. There are many discrete analytical methodologies, some of which are finite element, finite difference, and boundary element. This part of ISO 10303 addresses only finite element analysis.

In performing an analysis with finite element analysis methods the continuum of a product is discretized into a finite element model which consists of a mesh of points (nodes) in the continuum that are connected with elements. The elements represent finite portions of the continuum that when connected with shared nodes collectively respond as would the entire continuum. The elements have associated physical and material properties. There are also coordinate systems, groups, and administrative information associated with the finite element model. Load, constraint, and analysis output control information, along with analysis selection information, are combined with the finite element model to form a complete input to an analysis. Once an analysis is performed, analysis results information may be output at the nodes and at one or more positions within an element. There may be other output information not associated with a position within the finite element model such as total strain energy.

This part of ISO 10303 specifies the resources for the exchange of the information associated with the discretized (finite element) model, and the analysis controls, boundary conditions, and analysis results information that are associated with it. It is expected that the reader of this part of ISO 10303 is familiar with finite element analysis techniques.

Industrial automation systems and integration — Product data representation and exchange — Part 104 : Integrated application resources : Finite element analysis

1 Scope

This part of ISO 10303 specifies the resources for finite element analysis input and output information. This includes finite element models, the analysis control information, and the analysis results. The finite element analysis information model is partitioned into four schemas.

NOTE – The EXPRESS-G schema level model in figure 1 shows the relationships of the four schemas and their relationships to the ISO 10303 Integrated Resources.

Figure 1 – FEA schema level model in EXPRESS-G

The following rules apply to the entire FEA information model:

- a model shall be combined with a control to produce a complete FEA input;
- a result shall be combined with a model and a control to be valid.

  NOTE – The FEA information model includes the consideration of a multi-disciplinary analysis environment, which is presented in detail in Annex C.

1.1 Structural response definition schema scope

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

- the definitional aspects of a finite element analysis model;
- the definitional aspects of a finite element;
- the definitional aspects of a node.

1.2 Structural response representation schema scope

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

- node, elements and the associated element material and geometric property representations which combine to form a discretised mesh;
- material representations;
- coordinate space representations;
- administrative information.

1.3 Finite element analysis control and result schema scope

The following are within the scope of the **finite_element_analysis_control_and_result_-schema** of this part of ISO 10303 :

- analysis selection and related information;
- environmental information such as loads and constraints;
- output control information;
- nodal output information;
- element output information;

- output information applying to a whole model ;
- administrative information.

1.4 Analysis type scope

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

Linear static structural analysis of 2D and 3D continua and embedded elements such as thick shells, thin shells, beams and bars

- general 3D stress;
- plane stress;
- axisymmetric and simple plane strain.

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

Linear static structural analysis

- generalized plane strain

1.5 Scalars, vectors, and tensors schema scope

The following are the scalars, vectors and tensors necessary to represent the input and output of finite element analyses that are within the scope of the **fea_scalar_vector_tensor_schema** of this part of ISO 10303 :

- scalars;
- 2D and 3D vectors;
- 2D and 3D second order tensors;
- 2D and 3D symmetric fourth order tensors.

2 Normative references

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

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

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

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

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

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

ISO/DIS 10303-45[1)], *Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resources: Materials.*

[1)] To be published.

3 Definitions, symbols, and abbreviations

3.1 Terms defined in ISO 10303-1

This part of ISO 10303 makes use of the following terms defined in ISO 10303-1:

- application;
- application protocol;
- application resource;
- context;
- data;
- generic resource;
- information;
- information model;
- integrated resource;
- model;
- product;
- product data;
- structure.

3.2 Definitions and abbreviations

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

3.2.1 3d_model: A finite element model that has geometry in three dimensions.

3.2.2 2d_model: A finite element model that has geometry restricted to a plane that is either swept around an axis of symmetry (axisymmetric model) or projected perpendicular to a plane (plane model) to create the third dimension of the volume.

3.2.3 FEA: Finite Element Analysis

3.2.4 DOF: Degree of Freedom

3.3 Symbols

For the purposes of this part of ISO 10303, the following symbols apply. The notation in table 1 is used in the definition of element coordinate systems:

Table 1 – Element coordinate system definition symbols

Symbol	Definition
$\mathbf{X}$	Vector quantity
$\langle\rangle$	Vector normalisation
$\times$	Vector (cross) product
$\cdot$	Scalar product
$\|\|$	Vector magnitude

The notation in table 2 is used to denote the axes of a coordinate system:

Table 2 – Element coordinate axis symbols

Normalized Axes	Definition of the Coordinate System
(ξ,η,ζ)	an element parametric system
$(\mathbf{x},\mathbf{y},\mathbf{z})$	an element orthogonal system
$(\mathbf{x}',\mathbf{y}',\mathbf{z}')$	an intermediate element orthogonal system (used in the derivation of an element orthogonal system)
$(\mathbf{X},\mathbf{Y},\mathbf{Z})$	a specified arbitrary system
$(\mathbf{i},\mathbf{j},\mathbf{k})$	the 2D analysis plane definition system

4 Structural response definition schema

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

EXPRESS specification:

```
*)
SCHEMA structural_response_definition_schema;

  REFERENCE FROM product_property_definition_schema
      (shape_aspect,
       property_definition);

(*
```

NOTE – The schema referenced above can be found in the following Part of ISO 10303:

product_property_definition_schema	ISO 10303-41

4.1 Introduction

The subject of the **structural_response_definition_schema** is the definitional aspects of a finite element analysis model that describes the structural response of a product.

4.2 Fundamental concepts and assumptions

The definitional aspects of a finite element model are related to the representational aspects that they describe by a **property_definition_representation** association to a **structural_response_property**. This relationship is enforced by the rules in the **structural_response_property_definition_representation** entity in the **structural_response_representation_schema** .

NOTE – The relationships of the definitional and representational aspects of the FEA information model are shown in figure 2.

Figure 2 – FEA information model high level relationships EXPRESS-G partial model

4.3 Structural response definition schema entity definitions: Structural response property

A **structural_response_property** identifies that a finite element model representation is related to the properties characterizing a product.

4.3.1 structural_response_property

A **structural_response_property** is the relationship of the response of a finite element model representation of a product to the properties characterizing the product.

EXPRESS specification:

```
*)
ENTITY structural_response_property
  SUBTYPE OF (property_definition);
END_ENTITY;
(*
```

4.4 Structural response definition schema entity definitions: Finite element analysis model definition

A finite element analysis model is a collection of information associated with the definition of a finite element analysis of a product.

4.4.1 fea_model_definition

An **fea_model_definition** describes a specific finite element analysis model.

EXPRESS specification:

```
*)
ENTITY fea_model_definition
  SUBTYPE OF (shape_aspect);
END_ENTITY;
(*
```

4.5 Structural response definition schema entity definitions: Nodes

Nodes are the discretized points in the continuum being modelled that are connected with elements.

4.5.1 node_definition

A **node_definition** is a description of the shape aspect of a product that a node represents.

EXPRESS specification:

```
*)
ENTITY node_definition
  SUBTYPE OF (shape_aspect);
END_ENTITY;
(*
```

4.6 Structural response definition schema entity definitions: Elements

A finite element is the basic building block of a finite element model.

4.6.1 element_definition

An **element_definition** is a description of the shape aspect of a product that an element represents.

EXPRESS specification:

```
*)
ENTITY element_definition
  SUBTYPE OF (shape_aspect);
END_ENTITY;
(*
```

EXPRESS specification:

```
*)
END_SCHEMA; -- structural_response_definition_schema
(*
```

5 Structural response representation schema

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

EXPRESS specification:

```
*)
SCHEMA structural_response_representation_schema;

  REFERENCE FROM support_resource_schema
      (text,
       identifier);

  REFERENCE FROM measure_schema
      (context_dependent_measure,
       length_measure,
       plane_angle_measure,
       parameter_value,
       thermodynamic_temperature_measure);

  REFERENCE FROM group_schema
      (group,
       group_relationship);

  REFERENCE FROM product_property_representation_schema
      (property_definition_representation);

  REFERENCE FROM product_property_definition_schema
      (characterized_definition,
       property_definition);

  REFERENCE FROM representation_schema
      (definitional_representation,
       representation,
       representation_item,
       representation_relationship,
       using_representations);

  REFERENCE FROM structural_response_definition_schema
      (structural_response_property,
       fea_model_definition,
       node_definition,
       element_definition);

  REFERENCE FROM finite_element_analysis_control_and_result_schema
      (volume_3d_face,
       volume_2d_face,
       volume_3d_edge,
       volume_2d_edge,
       surface_3d_face,
       surface_2d_face,
```

```
      surface_3d_edge,
      surface_2d_edge,
      volume_scalar_variable,
      surface_scalar_variable,
      curve_scalar_variable,
      unspecified_value,
      measure_or_unspecified_value,
      volume_variable,
      surface_element_variable,
      curve_element_variable);

  REFERENCE FROM fea_scalar_vector_tensor_schema
     (scalar,
      tensor1,
      symmetric_tensor2_2d,
      symmetric_tensor2_3d,
      symmetric_tensor4_2d,
      symmetric_tensor4_3d);

  REFERENCE FROM geometry_schema
     (geometric_representation_item,
      geometric_representation_context,
      point,
      cartesian_point,
      direction,
      point_on_curve,
      point_on_surface,
      axis2_placement_2d,
      axis2_placement_3d,
      cross_product,
      normalise,
      curve,
      surface);

  REFERENCE FROM topology_schema
     (poly_loop);

  REFERENCE FROM geometric_model_schema
     (solid_model);


  REFERENCE FROM material_property_definition_schema
     (material_property);

  REFERENCE FROM material_property_representation_schema
     (material_property_representation);
(*
```

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

support_resource_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41

measure_schema	ISO 10303-41
representation_schema	ISO 10303-43
structural_response_definition_schema	Clause 4 of this part of ISO 10303
finite_element_analysis_control_and_result_schema	Clause 6 of this part of ISO 10303
fea_scalar_vector_tensor_schema	Clause 7 of this part of ISO 10303
geometry_schema	ISO 10303-42
material_property_definition_schema	ISO 10303-45
material_property_representation_schema	ISO 10303-45

5.1 Introduction

The subjects of the **structural_response_representation_schema** are the generalized nodes, elements, materials, and element properties which are combined to create an FEA model of a product.

NOTES

1 – The overall structure of the FEA information model and the external relationships with the Integrated Resources and Materials are shown in figure 1.

2 – The element entities are subtyped for extensibility with a minimum of disturbance to the existing information model.

3 – Many model generation and representation features contained in input files for some FEA applications are not reproduced exactly as an application would represent them within this information model, but the requisite information can be derived from the more generic form in this part of ISO 10303. Hence information may be expanded during conversion from the input file for an FEA application into the structures in ISO 10303-104. If the information is then converted back, an equivalent but expanded input file may be obtained.

5.2 Fundamental concepts and assumptions

The fundamental concepts and assumptions include the uniqueness of a finite element analysis model, the relationships of a finite element model to a product, the geometric founding and the definition of two and three dimensional finite element models, identifiers, coordinate systems, element integration, units and measures, the interface with the Materials schema, the relationships between the model definition and representation, control, and result portions of a finite element analysis, and subtyping philosophy and extensibility.

NOTE – The concept of subtyping to allow extensibility with a minimum of impact on the existing information model was a key design philosophy. Several single subtypes occur in the Finite Element Analysis Model information model for reasons of extensibility.

The surface property entities are an example of a supertype with only one subtype. The single subtype supports an extension of a functional description of surface thickness without structural change to the information model. Similarly, for the curve element properties, support is provided for the addition of standard cross-section information and cross-section libraries.

5.2.1 Product relationship

The relationship of a finite element analysis model and the product for which it is a representation is specified using a **structural_response_property** entity.

5.2.2 Finite element analysis model relationships

Nodes, elements, surface element properties, curve element properties, FEA materials, groups, and controls are associated with a unique finite element model.

5.2.3 Geometric founding and analysis space

The **fea_model** is a **representation** and is therefore geometrically founded in a coordinate space. This reference defines the basic coordinate system of the **fea_model** (see ISO 10303-43 for a definition of geometric founding). The number of dimensions (two or three) of the basic coordinate system establishes the dimensionality of the analysis.

5.2.4 Identifiers

An identifier within the FEA information model commonly is an integer. In this part of ISO 10303 integer identifiers are represented by string identifiers that may contain only digits. An identifier can be any string of characters, both letters and digits, in any combination.

5.2.5 Coordinate systems

The placement coordinate systems and geometric founding specified within parts ISO 10303-42 and 10303-43 are generalized in this part of ISO 10303 to include spherical and cylindrical coordinate systems. All coordinate systems shall be right handed.

5.2.6 Element matrix integration

Many finite elements require numerical integration of the matrices that are generated from the element specification. Element matrix has been included in this part of ISO 10303 to provide flexibility in defining element matrix integration options. This information is optional. To provide a simple integration specification option a textual description may be associated with an element.

NOTE – It is intended that the element integration and basis information provide the translator or interface writer with information for a best guess match for the more unusual types of elements.

5.2.7 Units and measures

Units are assigned in a context, and the value of the measure is assigned directly.

5.2.8 Interface with the materials schema

The FEA idealisation of material response behavior matrices is included in this part of ISO 10303. Information associated with a material (such as supplier and information generation environment) is given in ISO 10303-45.

NOTE – FEA element integration points and composite materials in material layers are correlated to plies by geometric position. The plies are defined by an Application Protocol using this part of ISO 10303 as a resource. This makes possible the use of a completely general integration and output point location definition capability within all elements.

5.2.9 Finite element analysis model, control, and result relationships

The **result** entity refers to a **control** entity which in turn refers to a **fea_model** and are thereby associated with a unique **fea_model**.

5.3 Structural response representation schema type definitions

The following TYPES are a resource for the entities in the **finite_element_analysis_model_-schema**.

5.3.1 axi_or_plane

An **axi_or_plane** specifies whether an **fea_model_2d** is axisymmetric, or plane.

EXPRESS specification:

```
*)
TYPE axi_or_plane = ENUMERATION OF
  (axisymmetric,
   planar);
END_TYPE;
(*
```

Enumerated item definitions:

axisymmetric: the **fea_model** is an axisymmetric analysis model where 2D element geometry is assumed to be swept about the j axis of the founding coordinate system to create a volume.

planar: the **fea_model** is a 2D analysis model where 2D elements are assumed to be extruded perpendicular to the analysis plane to create a volume.

5.3.2 coordinate_system_type

A **coordinate_system_type** is an enumeration of the allowable types of coordinate systems.

Figure 3 defines the convention for a Cartesian coordinate system:

Figure 3 – Cartesian coordinate system definition

Figure 4 defines the convention for a cylindrical coordinate system:

Figure 4 – Cylindrical coordinate system definition

Figure 5 defines the convention for a Spherical coordinate system:

Figure 5 – Spherical coordinate system definition

EXPRESS specification:

```
*)
TYPE coordinate_system_type = ENUMERATION OF
     (cartesian,
      cylindrical,
      spherical);
END_TYPE;
(*
```

Enumerated item definitions:

cartesian: the coordinate system is of type cartesian.

cylindrical: the coordinate system is of type cylindrical.

spherical: the coordinate system is of type spherical.

5.3.3 element_order

An **element_order** is an enumeration of the allowable geometric order of the element interpolation functions of a finite element.

EXPRESS specification:

```
*)
TYPE element_order = ENUMERATION OF
     (linear,
      quadratic,
      cubic);
END_TYPE;
(*
```

Enumerated item definitions:

linear: the element basic interpolation order is linear.

quadratic: the element basic interpolation order is quadratic.

cubic: the element basic interpolation order is cubic.

5.3.4 plane_2d_element_assumption

A **plane_2d_element_assumption** is an enumeration of the allowable assumptions of response through the thickness of a finite element.

EXPRESS specification:

```
*)
TYPE plane_2d_element_assumption = ENUMERATION OF
     (plane_stress,
      plane_strain);
END_TYPE;
(*
```

Enumerated item definitions:

plane_stress: a specialised case of 2D plane elements with membrane only properties where the stress normal to the element is zero.

plane_strain: a specialised case of 2D plane elements with membrane only properties where the strain normal to the element is zero, but the stress is not.

5.3.5 volume_element_purpose

A **volume_element_purpose** is an enumeration of the allowable types of strain-displacement relationship assumed for a volume element.

EXPRESS specification:

```
*)
TYPE volume_element_purpose = ENUMERATION OF
     (stress_displacement);
END_TYPE;
(*
```

Enumerated item definitions:

stress_displacement: the element considers only specialised behaviour inside the stress-displacement analysis domain.

5.3.6 surface_element_purpose

A **surface_element_purpose** is an enumeration of the allowable types of strain-displacement relationship assumed for a surface element.

NOTE – The **surface_element_purpose** is referenced as a SET of SETs to allow the appropriate combinations of coupled and uncoupled responses, such as ((.MEMBRANE_DIRECT.)(.MEMBRANE_DIRECT.,.BENDING_DIRECT.)) to specify that an element has membrane direct and coupled membrane - bending direct responses.

EXPRESS specification:

```
*)
TYPE surface_element_purpose = ENUMERATION OF
     (membrane_direct,
      membrane_shear,
      bending_direct,
      bending_torsion,
      normal_to_plane_shear);
END_TYPE;
(*
```

Enumerated item definitions:

membrane_direct: in-plane deformations of the surface element contribute to the element's strain energy.

membrane_shear: in-plane shear deformations contribute to the element's strain energy.

bending_direct: changes in curvature of the surface element contribute to the element's strain energy.

bending_torsion: torsional deformations of the surface element contributes to the element's strain energy.

normal_to_plane_shear: normal-to-plane shear deformations contribute to the element's strain energy.

5.3.7 curve_element_purpose

A **curve_element_purpose** is an enumeration of the allowable types of strain-displacement relationship assumed for a curve element.

NOTE – The **curve_element_purpose** is referenced as a SET of SETs to allow the appropriate combinations of coupled and uncoupled responses, such as ((.AXIAL.)(.TORSION.,.WARPING.)) to specify that an element has axial and coupled torsion - warping responses.

EXPRESS specification:

```
*)
TYPE curve_element_purpose = ENUMERATION OF
     (axial,
      y_y_bending,
      z_z_bending,
      torsion,
      x_y_shear,
      x_z_shear,
      warping);
END_TYPE;
(*
```

Enumerated item definitions:

axial: axial deformation along the curve of the element contributes to the element's strain energy.

y_y_bending: changes in curvature about the curve element's local **y** axis contribute to the strain energy.

z_z_bending: changes in curvature about the curve element's local **z** axis contribute to the strain energy.

torsion: torsion about the curve of the element contribute to the element's strain energy.

x_y_shear: shear deformation in the xy plane of the curve contributes to the element's strain energy.

x_z_shear: shear deformation in the xz plane of the curve contributes to the element's strain energy.

warping: warping deformation of the cross-section of the curve contributes to the element's strain energy.

5.3.8 volume_3d_element_shape

A **volume_3d_element_shape** is an enumeration of the allowable geometric shapes of a 3D finite element.

EXPRESS specification:

```
*)
TYPE volume_3d_element_shape = ENUMERATION OF
     (hexahedron,
      wedge,
      tetrahedron,
      pyramid);
END_TYPE;
(*
```

Enumerated item definitions:

hexahedron: a six-sided polyhedron where all sides are quadrilaterals.

wedge: a five-sided polyhedron where three sides are quadrilaterals, and two are triangles.

tetrahedron: a four-sided polyhedron where all sides are triangles.

pyramid: a five-sided polyhedron where four sides are triangles and one a quadrilateral.

5.3.9 element_2d_shape

An **element__shape** is an enumeration of the allowable geometric shapes of a 2D (e.g. axisymmetric) finite element.

EXPRESS specification:

```
*)
TYPE element_2d_shape = ENUMERATION OF
     (quadrilateral,
      triangle);
END_TYPE;
(*
```

Enumerated item definitions:

quadrilateral: a four-edged polygon.

triangle: a three-edged polygon.

5.3.10 matrix_property_type

A **matrix_property_type** is an enumeration of the allowable property types of an explicit or volume element matrix. The type is used to define the matrix integrated within an explicit or

volume element thus allowing different interpolation rules and integration methods to be used for different types of response relationships.

EXPRESS specification:

```
*)
TYPE matrix_property_type = ENUMERATION OF
     (stiffness,
      mass,
      damping);
END_TYPE;
(*
```

Enumerated item definitions:

stiffness: the matrix contributes to the explicit or volume element small displacement linear elastic stiffness matrix.

mass: the matrix contributes to the explicit or volume element mass matrix.

damping: the matrix contributes to the explicit or volume element damping matrix.

5.3.11 surface_matrix_property_type

A **surface_matrix_property_type** is an enumeration of the allowable property types of a surface element matrix. The type is used to define the matrix integrated within a surface element thus allowing different interpolation rules and integration methods to be used for different types of response relationships.

EXPRESS specification:

```
*)
TYPE surface_matrix_property_type = ENUMERATION OF
     (membrane_direct,
      membrane_shear,
      bending_direct,
      bending_torsion,
      normal_to_plane_shear,
      membrane_direct_mass,
      membrane_shear_mass,
      bending_direct_mass,
      bending_torsion_mass,
      normal_to_plane_shear_mass,
      mass);
END_TYPE;
(*
```

Enumerated item definitions:

membrane_direct: the matrix contributes direct membrane stiffness to the surface element small displacement linear elastic stiffness matrix.

membrane_shear: the matrix contributes shear membrane stiffness to the surface element small displacement linear elastic stiffness matrix.

bending_direct: the matrix contributes direct bending stiffness to the surface element small displacement linear elastic stiffness matrix.

bending_torsion: the matrix contributes bending torsional stiffness to the surface element small displacement linear elastic stiffness matrix.

normal_to_plane_shear: the matrix contributes normal to plane shear stiffness to the surface element small displacement linear elastic stiffness matrix.

membrane_direct_mass: the matrix contributes direct membrane mass to the surface element mass matrix.

membrane_shear_mass: the matrix contributes shear membrane mass to the surface element mass matrix.

bending_direct_mass: the matrix contributes direct bending mass to the surface element mass matrix.

bending_torsion_mass: the matrix contributes bending torsional mass to the surface element mass matrix.

normal_to_plane_shear_mass: the matrix contributes normal to plane shear mass to the surface element mass matrix.

mass: the matrix contributes mass to the surface element mass matrix.

5.3.12 curve_matrix_property_type

A **curve_matrix_property_type** is an enumeration of the allowable property types of curve element matrix. The type is used to define the matrix integrated within a curve element thus allowing different interpolation rules and integration methods to be used for different types of response relationships.

EXPRESS specification:

```
*)
TYPE curve_matrix_property_type = ENUMERATION OF
     (axial,
      y_y_bending,
      z_z_bending,
      torsion,
      x_y_shear,
      x_z_shear,
      warping,
      axial_mass,
      y_y_bending_mass,
      z_z_bending_mass,
      torsion_mass,
      x_y_shear_mass,
```

```
      x_z_shear_mass,
      warping_mass,
      mass);
END_TYPE;
(*
```

Enumerated item definitions:

axial: the matrix contributes axial stiffness to the curve element small displacement linear elastic stiffness matrix.

y_y_bending: the matrix contributes y bending stiffness to the curve element small displacement linear elastic stiffness matrix.

z_z_bending: the matrix contributes z bending stiffness to the curve element small displacement linear elastic stiffness matrix.

torsion: the matrix contributes torsional stiffness to the curve element small displacement linear elastic stiffness matrix.

x_y_shear: the matrix contributes y shear stiffness to the curve element small displacement linear elastic stiffness matrix.

x_z_shear: the matrix contributes z shear stiffness to the curve element small displacement linear elastic stiffness matrix.

warping: the matrix contributes warping stiffness to the curve element small displacement linear elastic stiffness matrix.

axial_mass: the matrix contributes axial mass to the curve element mass matrix.

y_y_bending_mass: the matrix contributes y bending mass to the curve element mass matrix.

z_z_bending_mass: the matrix contributes z bending mass to the curve element mass matrix.

torsion_mass: the matrix contributes torsional mass to the curve element mass matrix.

x_y_shear_mass: the matrix contributes y shear mass to the curve element mass matrix.

x_z_shear_mass: the matrix contributes z shear mass to the curve element mass matrix.

warping_mass: the matrix contributes warping mass to the curve element mass matrix.

mass: the matrix contributes mass to the curve element mass matrix.

5.3.13 matrix_symmetry

A **matrix_symmetry** is an enumeration of the allowable types of symmetry of a square matrix. The type of symmetry determines the position in an ARRAY attribute for each term of a matrix.

Except for asymmetric matrices, only the lower half of the matrix is populated.

For a **symmetric** matrix of size n (where n denotes the number of freedoms defined by the degree_of_freedom_list) the position of the terms shall be:

$$\begin{pmatrix} 1 & & & & & & \text{(symmetric)} \\ 2 & n+1 & & & & & \\ 3 & n+2 & 2n & & & & \\ \cdot & \cdot & \cdot & 3n-2 & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\ n & 2n-1 & 3n-3 & \cdot & \cdot & \cdot & n(n+1)/2 \end{pmatrix}$$

For a **diagonal** matrix of size n (where n denotes the number of freedoms defined by the degree_of_freedom_list) the position of the terms shall be:

$$\begin{pmatrix} 1 & & & & \\ & 2 & & 0 & \\ & & 3 & & \\ & 0 & & \ddots & \\ & & & & n \end{pmatrix}$$

EXPRESS specification:

```
*)
TYPE matrix_symmetry = ENUMERATION OF
     (symmetric,
      diagonal);
END_TYPE;
(*
```

Enumerated item definitions:

symmetric: the matrix is symmetrical about the diagonal, e.g. the (1, 2) term matches the (2, 1) term.

diagonal: the matrix has terms on the diagonal only.

5.3.14 degree_of_freedom

A **degree_of_freedom** is an enumeration of the allowable degrees of freedom to be considered for the intended analysis.

EXPRESS specification:

```
*)
TYPE degree_of_freedom = ENUMERATION OF
     (x_translation, y_translation, z_translation,
      x_rotation, y_rotation, z_rotation,
      warp);
END_TYPE;
(*
```

Enumerated item definitions:

x_translation: a translation degree of freedom along the x-axis.

y_translation: a translation degree of freedom along the y-axis.

z_translation: a translation degree of freedom along the z-axis.

x_rotation: a rotation degree of freedom about the x-axis.

y_rotation: a rotation degree of freedom about the y-axis.

z_rotation: a rotation degree of freedom about the z-axis.

warp: a warping degree of freedom of a curve element cross-section.

5.3.15 curve_element_freedom

A **curve_element_freedom** is an enumeration of the allowable degrees of freedom to be considered for a curve element.

EXPRESS specification:

```
*)
TYPE curve_element_freedom = ENUMERATION OF
     (x_translation, y_translation, z_translation,
      x_rotation, y_rotation, z_rotation,
      warp,
      none);
END_TYPE;
(*
```

Enumerated item definitions:

x_translation: a translation degree of freedom along the x-axis.

y_translation: a translation degree of freedom along the y-axis.

z_translation: a translation degree of freedom along the z-axis.

x_rotation: a rotation degree of freedom about the x-axis.

y_rotation: a rotation degree of freedom about the y-axis.

z_rotation: a rotation degree of freedom about the z-axis.

warp: a warping degree of freedom of a curve element cross-section.

none: no degree of freedom of a curve element is specified.

5.3.16 integration_rule

An **integration_rule** is an enumeration of the allowable types of numerical integration rules to be considered by subtypes of the volume, surface and curve integration entities.

EXPRESS specification:

```
*)
TYPE integration_rule = ENUMERATION OF
     (gaussian,
      simpson);
END_TYPE;
(*
```

Enumerated item definitions:

gaussian: gaussian integration.

simpson: simpson's rule integration.

5.3.17 shape_function

A **shape_function** is an enumeration of the allowable types of interpolation function.

EXPRESS specification:

```
*)
TYPE shape_function = ENUMERATION OF
     (lagrangian,
      serendipity,
      hermitian,
      unspecified);
END_TYPE;
(*
```

Enumerated item definitions:

lagrangian: the shape function is based upon lagrangian polynomials.

serendipity: the shape function is based upon the serendipity modification of the lagrangian polynomials.

hermitian: the shape function is based upon hermitian polynomials.

unspecified: the shape function is not specified.

5.3.18 additional_node

An **additional_node** is an optional additional mid-edge, mid-face or mid-volume node for an **element**. This type is only to be used in the **additional_node_list** of the **element_representation** entity.

EXPRESS specification:

```
*)
TYPE additional_node = SELECT
     (node_representation,
      dummy_node);
END_TYPE;
(*
```

5.3.19 dummy_node

A **dummy_node** is a placeholder for a missing mid-edge, mid-face or mid-volume node for an **element**.

EXPRESS specification:

```
*)
TYPE dummy_node = ENUMERATION OF
     (dummy);
END_TYPE;
(*
```

5.3.20 volume_2d_element_representation

A **volume_2d_element_representation** is either an axisymmetric or plane volume 2d element.

EXPRESS specification:

```
*)
TYPE volume_2d_element_representation = SELECT
  (axisymmetric_volume_2d_element_representation,
   plane_volume_2d_element_representation);
END_TYPE;
(*
```

5.3.21 surface_2d_element_representation

A **surface_2d_element_representation** is either an axisymmetric or plane surface 2d element.

EXPRESS specification:

```
*)
TYPE surface_2d_element_representation = SELECT
  (axisymmetric_surface_2d_element_representation,
   plane_surface_2d_element_representation);
END_TYPE;
(*
```

5.3.22 curve_2d_element_representation

A **curve_2d_element_representation** is either an axisymmetric or plane curve 2d element.

EXPRESS specification:

```
*)
TYPE curve_2d_element_representation = SELECT
  (axisymmetric_curve_2d_element_representation,
   plane_curve_2d_element_representation);
END_TYPE;
(*
```

5.3.23 volume_2d_element_descriptor

A **volume_2d_element_descriptor** is a collection of information that specifies either an axisymmetric or plane volume element.

EXPRESS specification:

```
*)
TYPE volume_2d_element_descriptor = SELECT
     (axisymmetric_volume_2d_element_descriptor,
      plane_volume_2d_element_descriptor);
END_TYPE;
(*
```

5.3.24 surface_2d_element_descriptor

A **surface_2d_element_descriptor** is a collection of information that specifies either an axisymmetric or plane surface element.

EXPRESS specification:

```
*)
TYPE surface_2d_element_descriptor = SELECT
     (axisymmetric_surface_2d_element_descriptor,
      plane_surface_2d_element_descriptor);
END_TYPE;
(*
```

5.3.25 curve_2d_element_descriptor

A **curve_2d_element_descriptor** is a collection of information that specifies either an axisymmetric or plane curve element.

EXPRESS specification:

```
*)
TYPE curve_2d_element_descriptor = SELECT
     (axisymmetric_curve_2d_element_descriptor,
      plane_curve_2d_element_descriptor);
END_TYPE;
(*
```

5.3.26 volume_3d_element_coordinate_system

A **volume_3d_element_coordinate_system** is the orthogonal coordinate system for a volume 3D element that shall be either an arbitrary or parametric coordinate system.

EXPRESS specification:

```
*)
TYPE volume_3d_element_coordinate_system = SELECT
  (arbitrary_volume_3d_element_coordinate_system,
   parametric_volume_3d_element_coordinate_system);
END_TYPE;
(*
```

5.3.27 volume_2d_element_coordinate_system

A **volume_2d_element_coordinate_system** is the orthogonal coordinate system for a volume 2D element that shall be either an arbitrary or a parametric coordinate system.

EXPRESS specification:

```
*)
TYPE volume_2d_element_coordinate_system = SELECT
  (arbitrary_volume_2d_element_coordinate_system,
   parametric_volume_2d_element_coordinate_system);
END_TYPE;
(*
```

5.3.28 surface_3d_element_coordinate_system

A **surface_3d_element_coordinate_system** is an orthogonal coordinate system for a surface 3D element that shall be either an aligned, constant, or a parametric coordinate system.

EXPRESS specification:

```
*)
TYPE surface_3d_element_coordinate_system = SELECT
  (aligned_surface_3d_element_coordinate_system,
   parametric_surface_3d_element_coordinate_system,
   constant_surface_3d_element_coordinate_system);
END_TYPE;
(*
```

5.3.29 surface_2d_element_coordinate_system

A **surface_2d_element_coordinate_system** is a coordinate system for a surface 2D element that shall be either an aligned or a parametric coordinate system.

EXPRESS specification:

```
*)
TYPE surface_2d_element_coordinate_system = SELECT
  (aligned_surface_2d_element_coordinate_system,
```

```
   parametric_surface_2d_element_coordinate_system);
END_TYPE;
(*
```

5.3.30 curve_3d_element_coordinate_system

A **curve_3d_element_coordinate_system** is an orthogonal coordinate system for a curve 3D element that shall be either aligned or parametric.

EXPRESS specification:

```
*)
TYPE curve_3d_element_coordinate_system = SELECT
  (aligned_curve_3d_element_coordinate_system,
   parametric_curve_3d_element_coordinate_system);
END_TYPE;
(*
```

5.3.31 curve_element_end_coordinate_system

A **curve_element_end_coordinate_system** is a coordinate system for the end offset and end release of curve elements.

EXPRESS specification:

```
*)
TYPE curve_element_end_coordinate_system = SELECT
     (fea_axis2_placement_3d,
      curve_3d_element_coordinate_system);
END_TYPE;
(*
```

5.3.32 element_aspect

An **element_aspect** is the type of element aspect to be associated with a geometric shape representation.

EXPRESS specification:

```
*)
TYPE element_aspect = SELECT
     (element_volume,
      volume_3d_face,
      volume_2d_face,
      volume_3d_edge,
      volume_2d_edge,
      surface_3d_face,
      surface_2d_face,
      surface_3d_edge,
      surface_2d_edge,
      curve_edge);
END_TYPE;
(*
```

5.3.33 element_volume

An **element_volume** is the entire volume of an **element_representation**.

EXPRESS specification:

```
*)
TYPE element_volume = ENUMERATION OF
     (volume);
END_TYPE;
(*
```

Enumerated item definitions:

volume: the volume of an element.

5.3.34 curve_edge

A **curve_edge** is the curve comprising the element as graphically established in figures 10 through 12.

EXPRESS specification:

```
*)
TYPE curve_edge = ENUMERATION OF
     (element_edge);
END_TYPE;
(*
```

Enumerated item definitions:

element_edge: the edge of a curve element.

5.4 Structural response representation schema entity definitions: Finite element analysis model

A finite element analysis model is a collection of information that represents the finite element analysis of a product. This information includes generalised nodes, elements, materials, properties, and groups which are combined to form a discrete mesh model of the product.

5.4.1 fea_model

An **fea_model** represents a specific finite element analysis model.

EXPRESS specification:

```
*)
ENTITY fea_model
  SUPERTYPE OF (ONEOF(fea_model_2d,
                      fea_model_3d))
  SUBTYPE OF (representation);
  creating_software      : text;
  intended_analysis_code : text;
  description            : text;
  analysis_type          : text;
UNIQUE
  UR1:  SELF\representation.name;
END_ENTITY;
(*
```

Attribute definitions:

creating_software: the name of the software used to create an **fea_model**. The version of the software should be specified.

intended_analysis_code: the name of the intended analysis code that an **fea_model** was created for.

description: a description of the **fea_model** providing supporting information supplied by the author(s) of the model.

analysis_type: a description of what type of analysis is to be performed with this **fea_model**.

Formal propositions:

UR1: the **id** shall be unique to each finite element analysis model.

NOTE – The model id, a unique application defined identifier of an **fea_model** , is specified by the **name** attribute of the **representation** supertype.

5.4.2 fea_model_3d

An **fea_model_3d** is an **fea_model** that has geometry in three dimensions for three-dimensional (3D) analyses.

EXPRESS specification:

```
*)
ENTITY fea_model_3d
  SUBTYPE OF (fea_model);
WHERE
  WR1: SELF\representation.context_of_items\
       geometric_representation_context.coordinate_space_dimension = 3;
END_ENTITY;
(*
```

Formal propositions:

WR1: the dimension count of the founding coordinate system shall be 3.

5.4.3 fea_model_2d

An **fea_model_2d** is an **fea_model** whose geometry is restricted to a plane for plane or axisymmetric analyses.

EXPRESS specification:

```
*)
ENTITY fea_model_2d
  SUBTYPE OF (fea_model);
  type_of_2d_analysis    : axi_or_plane;
WHERE
  WR1: SELF\representation.context_of_items\
       geometric_representation_context.coordinate_space_dimension = 2;
END_ENTITY;
(*
```

Attribute definitions:

type_of_2d_analysis: indicates whether the **fea_model** is axisymmetric or planar.

Formal propositions:

WR1: the dimension count of the founding coordinate system shall be 2.

5.4.4 structural_response_property_definition_representation

A **structural_response_property_definition_representation** is an association between the property of a finite element model, node or element and its representation.

EXPRESS specification:

```
*)
ENTITY structural_response_property_definition_representation
  SUBTYPE OF (property_definition_representation);
WHERE
  WR1: 'STRUCTURAL_REPSPONSE_REPRESNTATION_SCHEMA.STRUCTURAL_RESPONSE_PROPERTY'
       IN TYPEOF (SELF\property_definition_representation.definition);
  WR2: ((('STRUCTURAL_REPSPONSE_REPRESNTATION_SCHEMA.FEA_MODEL'
        IN TYPEOF
        (SELF\property_definition_representation.used_representation))
        AND
```

```
        ('STRUCTURAL_RESPONSE_DEFINITION_SCHEMA.FEA_MODEL_DEFINITION'
        IN TYPEOF
        (SELF\property_definition_representation.definition.definition)))
       OR
      (('STRUCTURAL_REPSPONSE_REPRESNTATION_SCHEMA.ELEMENT_SHAPE_REPRESENTATION'
        IN TYPEOF
        (SELF\property_definition_representation.used_representation))
        AND
        ('STRUCTURAL_RESPONSE_DEFINITION_SCHEMA.ELEMENT_DEFINITION'
        IN TYPEOF
        (SELF\property_definition_representation.definition.definition)))
       OR
      (('STRUCTURAL_REPSPONSE_REPRESNTATION_SCHEMA.NODE_SHAPE_REPRESENTATION'
        IN TYPEOF
        (SELF\property_definition_representation.used_representation))
        AND
        ('STRUCTURAL_RESPONSE_DEFINITION_SCHEMA.NODE_DEFINITION'
        IN TYPEOF
        (SELF\property_definition_representation.definition.definition))));
END_ENTITY;
(*
```

Formal propositions:

WR1: the property being represented shall be a **structural_response_property**.

WR2: the **fea_model** shall be associated with the shape definition of a model, or the **node_representation** shall be associated with the shape definition of a node, or the **element_representation** shall be associated with the shape definition of an element

.

5.4.5 fea_representation_item

An **fea_representation_item** is the constituents of a finite element.

EXPRESS specification:

```
*)
ENTITY fea_representation_item
  SUPERTYPE OF (ONEOF
                (arbitrary_volume_3d_element_coordinate_system,
                 parametric_volume_3d_element_coordinate_system,
                 arbitrary_volume_2d_element_coordinate_system,
                 parametric_volume_2d_element_coordinate_system,
                 aligned_surface_3d_element_coordinate_system,
                 parametric_surface_3d_element_coordinate_system,
                 constant_surface_3d_element_coordinate_system,
                 aligned_surface_2d_element_coordinate_system,
                 parametric_surface_2d_element_coordinate_system,
                 aligned_curve_3d_element_coordinate_system,
                 parametric_curve_3d_element_coordinate_system,
                 parametric_curve_3d_element_coordinate_direction,
```

```
                 curve_2d_element_coordinate_system,
                 directionally_explicit_element_coordinate_system_arbitrary,
                 directionally_explicit_element_coordinate_system_aligned))
  SUBTYPE OF (representation_item);
END_ENTITY;
(*
```

5.5 Structural response representation schema entity definitions: Nodal geometry and geometric coordinate systems

Nodes are the discretized points in the continuum being modeled that are connected with elements. There are three types of nodes: **node**, **geometric_node**, and **dummy_node**.

In a finite element analysis model a **node** shall have the following purposes:

- a **node** shall be a discretisation point for the field variables of the finite element analysis model. A finite element analysis model field has independent variables only at the **node**s.

- a **node** shall be connected to an element.

A model may contain **geometric_node**s which are not discretisation points for field variables and are not connected to an element. A **geometric_node** shall define the geometry of an element. For example the point may provide an off-axis position for a curve element to define the orientation of the section properties.

Alternatively, a model may contain some **dummy_nodes** which are not discretisation points for field variables. These **dummy_node**s are used as place holders in element **additional_node** arrays when optional (called additional in this information model) mid-edge, mid-side, and mid-volume nodes are not present.

Geometric coordinate systems provide a frame of reference for finite element analysis model geometric constructs. Nodal coordinates and material axis orientations are examples of some of the types of geometric information within a finite element analysis model. All FEA coordinate systems shall be defined with respect to the master or basic coordinate system defined by the **representation** entity SUBTYPE reference in the **fea_model** entity. The master coordinate system shall be Cartesian. The FEA information model has added to the following ISO 10303-42 and ISO 10303-43 entities to generalize coordinate systems to include cylindrical and spherical types:

- **representation_relationship**: specialised with a subtype to add a rule to ensure proper combinations of coordinate systems;

- **direction**: specialised with a subtype to allow the definition of direction ratios with nodes;

- **axis2_placement_3d**: specialised with a subtype to allow the selection of type of coordinate system.

An element coordinate system is derived from element nodal geometry. All element nodal geometry is founded by first aggregating **node_representation** entities in a **node_set**, and then associating the **node_set** with a **placement** by a combination of **mapped_item** and

representation_relationship entities in a manner consistent with the shape representation specifications of a referencing application protocol. The **fea_representation_relationship** entity is instantiated with a complex supertype with **representation_relationship** to enforce that cartesian, cylindrical, and spherical points are founded with respect to the proper type of **fea_axis2_placement_3d** entity (see ISO 10303-43 for a more complete discussion on geometric founding). All coordinate systems shall be right handed.

NOTE – Figure 6 shows the relationships of FEA with Geometry and Representation.

Figure 6 – EXPRESS-G partial model illustrating finite element analysis relationships with geometry and representation

5.5.1 direction_node

A **direction_node** is the components of the direction vector defined by a line from **node_1** to **node_2**. The actual magnitudes of the components have no effect upon the direction being defined; only the ratios x:y:z or x:y are significant.

NOTE – It is assumed that this vector will be normalised when used.

EXPRESS specification:

```
*)
ENTITY direction_node
  SUBTYPE OF (direction);
  node_1                          : node_representation;
  node_2                          : node_representation;
DERIVE
  SELF\direction.direction_ratios  : LIST [2:3] OF REAL := build_direction_node
                                     (node_1, node_2);
WHERE
  WR1: NOT ((direction_ratios[1] = 0.0) AND
            (direction_ratios[2] = 0.0) AND
            (direction_ratios[3] = 0.0));
END_ENTITY;
(*
```

Attribute definitions:

node_1: the first node used to define the direction vector.

node_2: the second node used to define the direction vector.

direction_ratios [1]: the component of the direction in the X axis.

direction_ratios [2]: the component of the direction in the Y axis.

direction_ratios [3]: the component of the direction in the Z axis; this will not be present in the case of a two dimensional space.

Formal propositions:

WR1: the magnitude of the direction vector shall be greater than zero.

5.5.2 fea_axis2_placement_2d

An **fea_axis2_placement_2d** is a location and orientation in two dimensional space, and declares that the Placement Coordinate System may be either Cartesian, Cylindrical, or Spherical. All coordinate systems in a finite element analysis model shall be right handed.

EXPRESS specification:

```
*)
ENTITY fea_axis2_placement_2d
  SUBTYPE OF (axis2_placement_2d);
  system_type                     : coordinate_system_type;
```

```
  description                       : text;
END_ENTITY;
(*
```

Attribute definitions:

system_type: the type of placement coordinate system, which may be a right handed Cartesian, Cylindrical, or Spherical coordinate system.

description: additional information about the formulation or purpose of the coordinate system.

Informal propositions:

IP1: the combination of an **fea_model** and **coordinate_system_id** shall be unique within a finite element analysis model.

NOTE – The system id, a unique application defined identifier of an **fea_axis2_placement_2d** , is specified by the **name** attribute of the **representation_item** supertype.

5.5.3 fea_axis2_placement_3d

An **fea_axis2_placement_3d** is a location and orientation in three dimensional space, and declares that the Placement Coordinate System may be either Cartesian, Cylindrical, or Spherical. All coordinate systems in a finite element analysis model shall be right handed.

EXPRESS specification:

```
*)
ENTITY fea_axis2_placement_3d
  SUBTYPE OF (axis2_placement_3d);
  system_type                       : coordinate_system_type;
  description                       : text;
END_ENTITY;
(*
```

Attribute definitions:

system_type: the type of placement coordinate system, which may be a right handed Cartesian, Cylindrical, or Spherical coordinate system.

description: additional information about the formulation or purpose of the coordinate system.

Informal propositions:

IP1: the combination of an **fea_model** and **coordinate_system_id** shall be unique within a finite element analysis model.

NOTE – The system id, a unique application defined identifier of an **fea_axis2_placement_3d** , is specified by the **name** attribute of the **representation_item** supertype.

5.5.4 fea_representation_relationship

An **fea_representation_relationship** specifies that the two representations being associated are founded in the same type of coordinate system (for example a **cylindrical_point** is founded by an **fea_axis2_placement_3d** whose **coordinate_system_type** attribute is cylindrical).

EXPRESS specification:

```
*)
ENTITY fea_representation_relationship
  SUBTYPE OF (representation_relationship);
END_ENTITY;
(*
```

Informal propositions:

IP1: the coordinate system types shall correspond between **point**s and **placement**s.

5.5.5 cylindrical_point

A **cylindrical_point** is a point in a three dimensional cylindrical coordinate system.

EXPRESS specification:

```
*)
ENTITY cylindrical_point
  SUBTYPE OF (point);
  r_coordinate                    : length_measure;
  theta_coordinate                : plane_angle_measure;
  z_coordinate                    : length_measure;
WHERE
  WR1:  (NOT( 'GEOMETRY_SCHEMA.CARTESIAN_POINT'   IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_ON_CURVE'    IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_ON_SURFACE'  IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.DEGENERATE_PCURVE' IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_REPLICA'     IN (TYPEOF(SELF)))) AND
        (NOT( 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.FEA_PARAMETRIC_POINT'
                                                   IN (TYPEOF(SELF)))) AND
        (NOT( 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.SPHERICAL_POINT'
                                                  IN (TYPEOF(SELF))));
END_ENTITY;
(*
```

Attribute definitions:

r_coordinate: the r coordinate of the point location.

theta_coordinate: the theta coordinate of the point location.

z_coordinate: the z coordinate of the point location. This cooridinate is zero if the **cylindrical_point** is founded in a two dimensional coordinate space.

Formal propositions:

WR1: only the SUBTYPE of **cylindrical_point** shall exist in this instance.

5.5.6 spherical_point

A **spherical_point** is a point in a three dimensional spherical coordinate system.

EXPRESS specification:

```
*)
ENTITY spherical_point
  SUBTYPE OF (point);
  r_coordinate                    : length_measure;
  phi_coordinate                  : plane_angle_measure;
  theta_coordinate                : plane_angle_measure;
WHERE
  WR1:  (NOT( 'GEOMETRY_SCHEMA.CARTESIAN_POINT'   IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_ON_CURVE'    IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_ON_SURFACE'  IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.DEGENERATE_PCURVE' IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_REPLICA'     IN (TYPEOF(SELF)))) AND
        (NOT( 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.FEA_PARAMETRIC_POINT'
                                                  IN (TYPEOF(SELF)))) AND
        (NOT( 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.CYLINDRICAL_POINT'
              IN (TYPEOF(SELF))));
END_ENTITY;
(*
```

Attribute definitions:

r_coordinate: the r coordinate of the point location.

phi_coordinate: the phi coordinate of the point location.

theta_coordinate: the theta coordinate of the point location. This cooridinate is zero if the **spherical_point** is founded in a two dimensional coordinate space.

Formal propositions:

WR1: only the SUBTYPE of **spherical_point** shall exist in this instance.

5.6 Structural response representation schema entity definitions: Node representation

Node representations are in the most general sense locations within a finite element model. These locations may be associated with geometric shape representations.

5.6.1 node_representation

A **node_representation** is location within a finite element model that may have associated degrees of freedom.

EXPRESS specification:

```
*)
ENTITY node_representation
  SUPERTYPE OF (ONEOF
                     (node,
                      geometric_node))
  SUBTYPE OF (representation);
  model_ref                       : fea_model;
```

```
UNIQUE
  UR1: model_ref, SELF\representation.name;
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the node.

NOTE – The node id, a unique application defined identifier of a **node_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and node id shall be unique within a model.

5.6.2 node_geometric_relationship

A **node_geometric_relationship** is the association of a **node_representation** with any type of geometric shape representation.

EXPRESS specification:

```
*)
ENTITY node_geometric_relationship
 SUBTYPE OF (representation_relationship);
WHERE
  WR1: 'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.SHAPE_REPRESENTATION'
       IN TYPEOF (SELF\representation_relationship.rep_1);
  WR2: 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.NODE_REPRESENTATION'
       IN TYPEOF (SELF\representation_relationship.rep_2);
END_ENTITY;
(*
```

Formal propositions:

WR1: the **rep_1** attribute of the **representation_relationship** supertype shall reference a **shape_representation**.

WR2: the **rep_2** attribute of the **representation_relationship** supertype shall reference a **node_representation**.

5.6.3 node

A **node** is a discretisation point for the field variables of the finite element analysis model.

EXPRESS specification:

```
*)
ENTITY node
  SUPERTYPE OF (node_with_vector ANDOR
                node_with_solution_coordinate_system)
  SUBTYPE OF (node_representation);
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
```

```
              'GEOMETRY_SCHEMA.POINT'
              IN TYPEOF (item))) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be exactly one position associated with a **node**. The position (POINT) is the coordinates of the node with respect to the founding placement coordinate system in the using **node_set**.

5.6.4 node_with_solution_coordinate_system

A **node_with_solution_coordinate_system** is a discretisation point with a solution coordinate system for the field variables of elements in a finite element analysis model.

EXPRESS specification:

```
*)
ENTITY node_with_solution_coordinate_system
  SUBTYPE OF (node);
WHERE
  WR1: ( (SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'FEA_AXIS2_PLACEMENT_3D'
              IN TYPEOF (item))) = 1)
          AND
          (SELF\representation.context_of_items\
       geometric_representation_context.coordinate_space_dimension = 3) )
     OR
       ( (SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'FEA_AXIS2_PLACEMENT_2D'
              IN TYPEOF (item))) = 1)
          AND
          (SELF\representation.context_of_items\
       geometric_representation_context.coordinate_space_dimension = 2) );
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be exactly one solution coordinate system associated with a **node**, and it shall be of the same coordinate space dimension as the corresponding geometic representation context. The solution coordinate system (FEA_AXIS2_PLACEMENT_3D or FEA_AXIS2_-PLACEMENT_2D) is the coordinate system in which the degrees of freedom of the node are defined.

5.6.5 node_with_vector

A **node_with_vector** is a discretisation point with a normal for the field variables of surface elements in a finite element analysis model.

EXPRESS specification:

```
*)
ENTITY node_with_vector
  SUBTYPE OF (node);
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'DIRECTION'
              IN TYPEOF (item))) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be exactly one surface normal associated with a **node_with_vector**. The surface normal (DIRECTION) is defined at the node with respect to the founding placement coordinate system, which is needed only when a node is connected to surface elements that require this information. The **normal** may correspond to the **z** axis of the **solution_coordinate_system**.

5.6.6 geometric_node

A **geometric_node** is a node that is for geometric reference only and therefore shall not be connected to an element.

EXPRESS specification:

```
*)
ENTITY geometric_node
  SUBTYPE OF (node_representation);
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
              'GEOMETRY_SCHEMA.' +
              'POINT'
              IN TYPEOF (item))) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be exactly one position associated with a **geometric_node**. The position (POINT) is the coordinates of the node with respect to the founding placement coordinate system in the using **node_set**.

5.6.7 node_set

A **node_set** is the aggregation of **node_representation** entities for geometric founding in a common coordinate frame.

EXPRESS specification:

```
*)
ENTITY node_set
```

```
  SUBTYPE OF (geometric_representation_item);
  nodes                              : SET [1:?] OF node_representation;
WHERE
  WR1: SIZEOF (QUERY (tmp <* nodes |
              tmp\representation.context_of_items :<>:
              nodes[1]\representation.context_of_items)) = 0;
END_ENTITY;
(*
```

Attribute definitions:

nodes: the nodes to be founded in a common coordinate frame.

Formal propositions:

WR1: the geometric representation context of each of the nodes in the set shall be the same, and the context shall be of the same spatial dimension as the location of the nodes. The founding representation that uses a node set shall also share the same context.

5.7 Structural response representation schema entity definitions: Element representations

Elements are connected to nodes to model the continuum being analyzed. Different types of elements have different characteristics due to different idealisations of the continuum being analyzed. For example, a surface element has a thickness, while a volume element does not. The idealisations are represented in the subtypes of the **element_representation entity**. The subtypes are defined as follows:

- **volume element**: A volume element includes all elements which describe a volume continuum, and have properties appropriate for a volume. For example, a 2D quadrilateral in an axisymmetric analysis is a volume element.

- **surface element**: A surface element includes all elements which describe a surface continuum, and have properties appropriate for a surface. For example, a curve 2D element in a axisymmetric analysis is a surface element.

- **curve element**: A curve element includes all elements which describe a curve of material (cable, truss or beam), and have properties appropriate for a cable, truss or beam. For example, a 2D point in an axisymmetric analysis is a curve element.

- **point element**: A point element includes point masses, grounded springs, and grounded dampers.

- **directionally explicit element**: A directionally explicit element includes all elements which are defined by explicit matrices, and which have an orientation defined in part by the location of the nodes of an element. Two noded springs, dampers, and coupled masses are all directionally explicit elements.

- **explicit element**: An explicit element includes elements which have an explicit matrix definition, such as externally defined elements, and substructures.

The geometry of a volume, surface or curve element is defined by the element shape, variable order, coordinate (and optionally the surface normal) for each of the nodes of an element, and the details of the shape functions of an element for geometric interpolation. A computer readable definition of the mathematical form of element shape functions for geometric interpolation is not provided. However, the exact specifications of these shape functions are important for they determine the element parametric coordinate system and hence the element geometry, any location in the element (if specified with respect to the element parametric coordinate system), and any orientations within the element (if specified with respect to the element parametric coordinate system). The element basis entity for a volume, surface, or curve element has an attribute **variable_shape_function** which can be used to describe the geometric shape function.

Aspects of element geometry (edge, face, volume) may be related to an appropriate geometric shape representation. An example of this is the relation of an edge to a curve.

Volume, surface, and curve element representations are subtyped for 2D (plane stress, plane strain or axisymmetric analyses), and for 3D. This split of subtypes reflects the physical idealisation of the element rather than merely the topology. As a consequence, the physical properties appropriate for each element subtype are stated precisely by the appropriate entity.

NOTE – The **element_representation** entity has subtypes as shown in figure 7.

Figure 7 – Element subtyping

The element coordinate system attributes of volume, surface, and curve elements are specified with the **representation_item** subtype **fea_representation_item**. The association of the proper subtypes of volume, surface, and curve elements with the appropriate subtypes of **representation_item** is specified with EXPRESS rules.

NOTE – The EXPRESS-G Partial Model in figure 8 shows the the relationship of volume elements to FEA and FEA material property representations. Surface and curve elements have similar relationships, and also have a relationship to element property entities.

Figure 8 – Volume element properties EXPRESS-G partial model

Point elements are different representations of nodal response matrices than the volume, surface, and curve elements. Rather than implicitly defining the element nodal response matrices as the volume, surface, and curve elements do, the point elements instead explicitly define the stiffness, mass, and damping nodal response matrices of an element. Thus a point element is a single node element that is a convenient way to define these matrices, rather than the more general matrix definition capability offered by the multi-noded explicit elements. The element coordinate system attributes of the matrices are associated with their **element_representation** by rules, as discussed above for volume, surface, and curve elements.

NOTE – The EXPRESS-G Partial Model in figure 9 illustrates the relationships of the SUBTYPE trees of the point, directionally explicit and explicit elements.

Figure 9 – Point, directionally explicit and explicit elements EXPRESS-G partial model

A nodal response matrix defined by a point, directionally explicit or explicit element, shall be one of:

stiffness: A stiffness between nodes for a directionally explicit or an explicit element, or a grounded spring for a point element, defined as follows:

$$\mathbf{f} = K\mathbf{d}$$

damping: A damping between nodes for a directionally explicit or an explicit element, or a grounded damper for a point element, defined as follows:

$$\mathbf{f} = D\mathbf{v}$$

mass: A mass for a directionally explicit, explicit or point element, defined as follows:

$$\mathbf{f} = M\mathbf{a}$$

where:

K is the element stiffness matrix.

D is the element damping matrix.

M is the element mass matrix.

$\mathbf{f}$ is the vector of generalised forces applied to the element at nodal degrees of freedom.

$\mathbf{d}$ is the vector of generalised displacements imposed upon nodal degrees of freedom

$\mathbf{v}$ is the vector of generalised velocities imposed upon nodal degrees of freedom

$\mathbf{a}$ is the vector of generalised accelerations imposed upon nodal degrees of freedom

5.7.1 element_representation

An **element_representation** is the aspect of a finite element which represents the mathematical relationships between the nodes of a finite element model.

EXPRESS specification:

```
*)
ENTITY element_representation
  SUPERTYPE OF (ONEOF(volume_3d_element_representation,
                      axisymmetric_volume_2d_element_representation,
                      plane_volume_2d_element_representation,
                      surface_3d_element_representation,
                      axisymmetric_surface_2d_element_representation,
                      plane_surface_2d_element_representation,
```

```
                    curve_3d_element_representation,
                    axisymmetric_curve_2d_element_representation,
                    plane_curve_2d_element_representation,
                    point_element_representation,
                    directionally_explicit_element_representation,
                    explicit_element_representation)
                ANDOR
                   (higher_order_element_representation))
  SUBTYPE OF (definitional_representation);
  required_node_list              : LIST [1:?] OF node_representation;
END_ENTITY;
(*
```

Attribute definitions:

required_node_list: the list of nodes which are essential for the element. For a volume, surface, or curve element these are the vertex nodes. For other types of elements, all nodes are required.

NOTE – The required node list (and potentially the additional node list in the **higher_order_element_representation** entity) define the connection of an element to other elements within the model. For volume, surface, or curve elements the positions of the nodes together with the element shape functions defines the shape and orientation of an element. For volume, surface or curve elements, the position of a node in the node arrays depends on the element shape and order as established graphically in the figures in 5.8.

5.7.2 higher_order_element_representation

A **higher_order_element_representation** is the aspect of a finite element which defines the mathematical relationship between the non-vertex nodes of a finite element model.

EXPRESS specification:

```
*)
ENTITY higher_order_element_representation
  SUBTYPE OF (element_representation);
  additional_node_list        : LIST [1:?] OF additional_node;
END_ENTITY;
(*
```

Attribute definitions:

additional_node_list: the list of additional nodes for a volume, surface or curve element. These nodes are NOT at element vertices.

Informal propositions:

IP1: TYPE **dummy_nodes** shall be used where mid-edge, mid-face, and mid-volume nodes are not specified in the **additional_node_list**.

5.7.3 element_geometric_relationship

An **element_geometric_relationship** is the association of an aspect (edge, face, or entire volume) of a particular type of element representation with the appropriate type of geometric shape representation.

EXPRESS specification:

```
*)
ENTITY element_geometric_relationship
 SUBTYPE OF (representation_relationship);
 aspect               : element_aspect;
WHERE
  WR1: 'PRODUCT_PROPERTY_REPRESENTATION_SCHEMA.SHAPE_REPRESENTATION'
       IN TYPEOF (SELF\representation_relationship.rep_1);
  WR2: 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.ELEMENT_REPRESENTATION'
       IN TYPEOF (SELF\representation_relationship.rep_2);
  WR3: consistent_geometric_reference (aspect,
       SELF\representation_relationship.rep_2.items);
  WR4: consistent_element_reference (SELF\representation_relationship.rep_2,
       aspect);
END_ENTITY;
(*
```

Attribute definitions:

aspect: the edge, face, or volume of an element.

Formal propositions:

WR1: the **rep_1** attribute of the **representation_relationship** supertype shall reference a **shape_representation**.

WR2: the **rep_2** attribute of the **representation_relationship** supertype shall reference an **element_representation**.

WR3: the correct type of element aspect shall be combined with the appropriate geometric shape representation. Element edges shall be combined with a curve, element faces with a surface, and the entire volume of an element with a solid model.

WR4: the type of element aspect (edge, face, volume) shall be appropriate to the type of element.

5.7.4 volume_3d_element_representation

A **volume_3d_element_representation** is a three dimensional (3D) shape.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_3d;
  element_descriptor               : volume_3d_element_descriptor;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
```

```
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'PARAMETRIC_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) <= 1;
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) <= 1;
  WR3: (SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'PARAMETRIC_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) = 1);
  WR4: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_3d_nodes (
        SELF\element_representation.required_node_list,
        element_descriptor.shape);
  FU2:  additional_3d_nodes (
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor.shape,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the volume 3d element.

element_descriptor: a collection of information that specifies a **volume_3d_element_representation**.

material: the material properties associated with a **volume_3d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: there may be a parametric material coordinate system associated with a **volume_3d_element_representation**.

The parametric material coordinate system (PARAMETRIC_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the element coordinate system for the material of the **volume_3d_element_representation**. The **x** axis of the element coordinate system shall coincide with the **x** axis of the material coordinate system, the **y** axis of the element coordinate system shall coincide with the **y** axis of the material coordinate system, and the **z** axis of the element coordinate system shall coincide with the **z** axis of the material coordinate system.

WR2: there may be an arbitrary material coordinate system associated with a **volume_3d_element_representation**.

The arbitrary material coordinate system (ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the element coordinate system for the material of the **volume_3d_element_representation**. The **x** axis of the element coordinate system shall coincide with the **x** axis of the material coordinate system, the **y** axis of the element coordinate system shall coincide with the **y** axis of the material coordinate system, and the **z** axis of the element coordinate system shall coincide with the **z** axis of the material coordinate system.

WR3: the material coordinate system shall be either a PARAMETRIC_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM or a ARBITRARY_VOLUME_3D_ELEMENT_COORDINATE_SYSTEM).

WR4: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

FU1: there shall be the correct number of required nodes for the type and shape of a **volume_3d_element_representation**.

FU2: there shall be the correct number of additional nodes for the type, shape, and order of a **volume_3d_element_representation**.

5.7.5 axisymmetric_volume_2d_element_representation

An **axisymmetric_volume_2d_element_representation** is a two dimensional (2D) shape swept about the **j** axis of the (**i**,**j**) plane of the founding coordinate system (see 5.2.3). An **axisymmetric_volume_2d_element_representation** shall lie in the (**i**,**j**) plane of the founding coordinate system.

EXPRESS specification:

```
*)
ENTITY axisymmetric_volume_2d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_2d;
  element_descriptor               : axisymmetric_volume_2d_element_descriptor;
  angle_property                   : axisymmetric_2d_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: 'AXISYMMETRIC' IN TYPEOF(model_ref.type_of_2d_analysis);
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN  TYPEOF (item))) <= 1;
  WR3: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN  TYPEOF (item))) <= 1;
  WR4: (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN  TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN  TYPEOF (item))) = 1);
  WR5: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_2d_nodes (
        SELF\element_representation.required_node_list,
        element_descriptor.shape);
  FU2:  additional_2d_nodes (
```

```
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor.shape,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the axisymmetric volume 2d element.

element_descriptor: a collection of information that specifies an **axisymmetric_volume_2d_element_representation**.

angle_property: the properties for an axisymmetric section. Analysis applications often assume that the value of the angle attribute of this property is 2π.

material: the material properties associated with an **axisymmetric_volume_2d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: the **fea_model** shall be axisymmetric.

WR2: there may be a parametric material coordinate system associated with an **axisymmetric_volume_2d_element_representation**.

The parametric material coordinate system (PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the element coordinate system for the material of the **axisymmetric_volume_2d_element_representation**. The **x** axis of the element coordinate system shall coincide with the **x** axis of the material coordinate system, the **y** axis of the element coordinate system shall coincide with the **y** axis of the material coordinate system, and the **z** axis of the element coordinate system shall coincide with the **z** axis of the material coordinate system.

WR3: there may be an arbitrary material coordinate system associated with an **axisymmetric_volume_2d_element_representation**.

The arbitrary material coordinate system (ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the element coordinate system for the material of the **axisymmetric_volume_2d_element_representation**. The **x** axis of the element coordinate system shall coincide with the **x** axis of the material coordinate system, the **y** axis of the element coordinate system shall coincide with the **y** axis of the material coordinate system, and the **z** axis of the element coordinate system shall coincide with the **z** axis of the material coordinate system.

WR4: the material coordinate system shall be either a PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM or an ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM).

WR5: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

FU1: there shall be the correct number of required nodes for the type and shape of an **axisymmetric_volume_2d_element_representation**.

FU2: there shall be the correct number of additional nodes for the type, shape, and order of an **axisymmetric_volume_2d_element_representation**.

5.7.6 plane_volume_2d_element_representation

A **plane_volume_2d_element_representation** is a two dimensional (2D) shape in the (**i**,**j**) plane of the founding coordinate system (see 5.2.3) with a depth perpendicular to the (**i**,**j**) plane. A **plane_volume_2d_element_representation** shall lie in the (**i**,**j**) plane of the founding coordinate system.

EXPRESS specification:

```
*)
ENTITY plane_volume_2d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_2d;
  element_descriptor               : plane_volume_2d_element_descriptor;
  depth_property                   : plane_2d_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: 'PLANAR' IN TYPEOF(model_ref.type_of_2d_analysis);
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
```

```
  WR3: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR4: (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1);
  WR5: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_2d_nodes (
        SELF\element_representation.required_node_list,
        element_descriptor.shape);
  FU2:  additional_2d_nodes (
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor.shape,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the plane volume 2d element.

element_descriptor: a collection of information that specifies a **plane_volume_2d_element_representation**.

depth_property: the properties for a plane section. Analysis applications often assume that the value of the depth attribute of this property is 1.0.

material: the material properties associated with a **plane_volume_2d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: the **fea_model** shall be planar.

WR2: there may be a parametric material coordinate system associated with a **plane_volume_2d_element_representation**.

The parametric material coordinate system (PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the element coordinate system for the material of the **plane_volume_2d_element_representation**. The **x** axis of the element coordinate system shall coincide with the **x** axis of the material coordinate system, the **y** axis of the element coordinate system shall coincide with the **y** axis of the material coordinate system, and the **z** axis of the element coordinate system shall coincide with the **z** axis of the material coordinate system.

WR3: there may be an arbitrary material coordinate system associated with a **plane_volume_2d_element_representation**.

The arbitrary material coordinate system (ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the element coordinate system for the material of the **plane_volume_2d_element_representation**. The **x** axis of the element coordinate system shall coincide with the **x** axis of the material coordinate system, the **y** axis of the element coordinate system shall coincide with the **y** axis of the material coordinate system, and the **z** axis of the element coordinate system shall coincide with the **z** axis of the material coordinate system.

WR4: the material coordinate system shall be either a PARAMETRIC_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM or an ARBITRARY_VOLUME_2D_ELEMENT_COORDINATE_SYSTEM).

WR5: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

FU1: there shall be the correct number of required nodes for the type and shape of a **plane_volume_2d_element_representation**.

FU2: there shall be the correct number of additional nodes for the type, shape, and order of a **plane_volume_2d_element_representation**.

5.7.7 surface_3d_element_representation

A **surface_3d_element_representation** is an element that is a surface with a specified section that is a 2D shape (a triangle or quadrilateral) in 3D space.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_3d;
  element_descriptor               : surface_3d_element_descriptor;
  property                         : surface_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'CONSTANT_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR3: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR4: (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'CONSTANT_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1);
  WR5: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
```

```
        SIZEOF (['FEA_LINEAR_ELASTICITY',
                 'FEA_MASS_DENSITY',
                 'FEA_AREA_DENSITY',
                 'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                 'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                 'FEA_MOISTURE_ABSORPTION',
                 'FEA_SHELL_MEMBRANE_STIFFNESS',
                 'FEA_SHELL_BENDING_STIFFNESS',
                 'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
                 'FEA_SHELL_SHEAR_STIFFNESS'] * TYPEOF (item2)
                 ) >= 1
       )) = 1
      ))) >= 1;
  FU1:  required_2d_nodes (
        SELF\element_representation.required_node_list,
        element_descriptor.shape);
  FU2:  additional_2d_nodes (
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor.shape,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the surface 3d element.

element_descriptor: a collection of information that specifies a **surface_3d_element_representation**.

property: the section properties of the surface.

material: the material properties associated with a **surface_3d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: there may be an arbitrary property coordinate system associated with a **surface_3d_element_representation**.

An arbitrary property coordinate system (ARBITRARY_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For the surface 3D elements, at each point on the surface of the element the referenced entity defines a coordinate system with its **z** axis normal to the element surface.

WR2: there may be a constant property coordinate system associated with a **surface_3d_element_representation**.

A constant property coordinate system (CONSTANT_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For the surface 3D elements, at each point on the surface of the element the referenced entity defines a coordinate system with its **z** axis normal to the element surface.

WR3: there may be a parametric property coordinate system associated with a **surface_3d_element_representation**.

A parametric property coordinate system (PARAMETRIC_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For the surface 3D elements, at each point on the surface of the element the referenced entity defines a coordinate system with its **z** axis normal to the element surface.

WR4: the property coordinate system shall be either an ARBITRARY_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM, a CONSTANT_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM, or a PARAMETRIC_SURFACE_3D_ELEMENT_COORDINATE_SYSTEM.

WR5: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

fea_shell_membrane_stiffness

fea_shell_bending_stiffness

fea_shell_membrane_bending_coupling_stiffness

fea_shell_shear_stiffness

FU1: there shall be the correct number of required nodes for the type and shape of a **surface_3d_element_representation**.

FU2: there shall be the correct number of additional nodes for the type, shape, and order of a **surface_3d_element_representation**.

5.7.8 axisymmetric_surface_2d_element_representation

An **axisymmetric_surface_2d_element_representation** is an element that is a surface with a specified section that is a 1D shape (curve) that is swept about the **j** axis of the (**i**,**j**) plane of the founding coordinate system (see 5.2.3). An **axisymmetric_surface_2d_element_representation** shall lie in the (**i**,**j**) plane of the founding coordinate system.

EXPRESS specification:

```
*)
ENTITY axisymmetric_surface_2d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_2d;
  element_descriptor               : axisymmetric_surface_2d_element_descriptor;
  property                         : surface_element_property;
  angle_property                   : axisymmetric_2d_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: 'AXISYMMETRIC' IN TYPEOF(model_ref.type_of_2d_analysis);
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) <= 1;
  WR3: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) <= 1;
  WR4: (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
              IN TYPEOF (item))) = 1);
  WR5: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
```

```
                  'FEA_MOISTURE_ABSORPTION',
                  'FEA_SHELL_MEMBRANE_STIFFNESS',
                  'FEA_SHELL_BENDING_STIFFNESS',
                  'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
                  'FEA_SHELL_SHEAR_STIFFNESS'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_1d_nodes (
        SELF\element_representation.required_node_list);
  FU2:  additional_1d_nodes (
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the axisymmetric surface 2d element.

element_descriptor: a collection of information that specifies an **axisymmetric_surface_2d_element_representation**.

property: the section properties of the surface.

angle_property: the properties for an axisymmetric section. Analysis applications often assume that the value of the angle attribute of this property is 2π.

material: the material properties associated with an **axisymmetric_surface_2d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: the **fea_model** shall be axisymmetric.

WR2: there may be an arbitrary property coordinate system associated with an **axisymmetric_surface_2d_element_representation**.

An arbitrary property coordinate system (ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For a **surface_2d_element_representation** the coordinate system is defined such that: the **x** axis is normal to the 2D analysis plane in the direction of the **k** axis of the analysis 2D definition coordinate system, and the **z** axis is normal to the element surface.

WR3: there may be a parametric property coordinate system associated with an **axisymmetric_surface_2d_element_representation**.

A parametric property coordinate system (PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For a **surface_2d_element_representation** the coordinate system is defined such that: the **x** axis is normal to the 2D analysis plane in the direction of the **k** axis of the analysis 2D definition coordinate system, and the **z** axis is normal to the element surface.

WR4: the property coordinate system shall be either an ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM or a PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM.

WR5: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

fea_shell_membrane_stiffness

fea_shell_bending_stiffness

fea_shell_membrane_bending_coupling_stiffness

fea_shell_shear_stiffness

FU1: there shall be the correct number of required nodes for the type and shape of a **axisymmetric_surface_2d_element_representation**.

FU2: there shall be the correct number of additional nodes for the type, shape, and order of an **axisymmetric_surface_2d_element_representation**.

5.7.9 plane_surface_2d_element_representation

A **plane_surface_2d_element_representation** is an element that is a surface with a specified section that is a 1D shape (curve) in the (**i**,**j**) plane of the founding coordinate system (see 5.2.3)

with a depth perpendicular to the (**i**,**j**) plane. A **plane_surface_2d_element_representation** shall lie in the (**i**,**j**) plane of the founding coordinate system.

EXPRESS specification:

```
*)
ENTITY plane_surface_2d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_2d;
  element_descriptor               : plane_surface_2d_element_descriptor;
  property                         : surface_element_property;
  depth_property                   : plane_2d_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: 'PLANAR' IN TYPEOF(model_ref.type_of_2d_analysis);
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR3: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR4: (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1);
  WR5: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION',
                  'FEA_SHELL_MEMBRANE_STIFFNESS',
                  'FEA_SHELL_BENDING_STIFFNESS',
                  'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
                  'FEA_SHELL_SHEAR_STIFFNESS'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_1d_nodes (
        SELF\element_representation.required_node_list);
  FU2:  additional_1d_nodes (
```

```
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the plane surface 2d element.

element_descriptor: a collection of information that specifies a **plane_surface_2d_element_representation**.

property: the section properties of the surface.

depth_property: the properties for a plane section. Analysis applications often assume that the value of the depth attribute of this property is 1.0.

material: the material properties associated with a **plane_surface_2d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: the **fea_model** shall be planar.

WR2: there may be an arbitrary property coordinate system associated with a **plane_surface_2d_element_representation**.

An arbitrary property coordinate system (ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For a **surface_2d_element_representation** the coordinate system is defined such that: the **x** axis is normal to the 2D analysis plane in the direction of the **k** axis of the analysis 2D definition coordinate system, and the **z** axis is normal to the element surface.

WR3: there may be a parametric property coordinate system associated with a **plane_surface_2d_element_representation**.

A parametric property coordinate system (PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties. For a **surface_2d_element_representation** the coordinate system is defined such that: the **x** axis is normal to the 2D analysis plane in the direction of the **k** axis of the analysis 2D definition coordinate system, and the **z** axis is normal to the element surface.

WR4: the property coordinate system shall be either an ARBITRARY_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM or a PARAMETRIC_SURFACE_2D_ELEMENT_COORDINATE_SYSTEM.

WR5: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

fea_shell_membrane_stiffness

fea_shell_bending_stiffness

fea_shell_membrane_bending_coupling_stiffness

fea_shell_shear_stiffness

FU1: there shall be the correct number of required nodes for the type and shape of a **plane_surface_2d_element_representation**.

FU2: there shall be the correct number of additional nodes for the type, shape, and order of a **plane_surface_2d_element_representation**.

5.7.10 curve_3d_element_representation

A **curve_3d_element_representation** is an element that is a 1D shape (a curve) with a specified cross section.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_3d;
  element_descriptor               : curve_3d_element_descriptor;
  property                         : curve_3d_element_property;
  material                         : element_material;
UNIQUE
```

```
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) <= 1;
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_DIRECTION'
             IN TYPEOF (item))) <= 1;
  WR3: (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1)
       XOR
       (SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_DIRECTION'
             IN TYPEOF (item))) = 1);
  WR4: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_1d_nodes (
        SELF\element_representation.required_node_list);
  FU2:  additional_1d_nodes (
        SELF\higher_order_element_representation.additional_node_list,
        element_descriptor\element_descriptor.topology_order);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the curve 3d element.

element_descriptor: a collection of information that specifies a **curve_3d_element_representation**.

property: the cross section properties of a **curve_3d_element_representation**.

material: the material properties associated with a **curve_3d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: there may be a parametric property coordinate system associated with a **curve_3d_element_representation**.

A parametric property coordinate system (PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties.

WR2: there may be a vector defined property coordinate system associated with a **curve_3d_element_representation**.

A vector defined property coordinate system (PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_DIRECTION_VECTOR) is the orientation of the coordinate system for the element material and section properties.

WR3: the property coordinate system shall be either a PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_SYSTEM, or a PARAMETRIC_CURVE_3D_ELEMENT_COORDINATE_DIRECTION_VECTOR.

WR4: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

FU1: there shall be the correct number of required nodes for a **curve_3d_element_representation**.

FU2: there shall be the correct number of additional nodes for a **curve_3d_element_representation**.

5.7.11 axisymmetric_curve_2d_element_representation

An **axisymmetric_curve_2d_element_representation** is an element with a specified cross section that is a 0D shape (a point) that is swept about the **j** axis of the (**i**,**j**) plane of the founding coordinate system (see 5.2.3). An **axisymmetric_curve_2d_element_representation** shall lie in the (**i**,**j**) plane of the founding coordinate system.

EXPRESS specification:

```
*)
ENTITY axisymmetric_curve_2d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_2d;
  element_descriptor               : axisymmetric_curve_2d_element_descriptor;
  property                         : curve_2d_element_property;
  angle_property                   : axisymmetric_2d_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: 'AXISYMMETRIC' IN TYPEOF(model_ref.type_of_2d_analysis);
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'CURVE_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1;
  WR3: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_0d_nodes (
        SELF\element_representation.required_node_list);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the axisymmetric curve 2d element.

element_descriptor: a collection of information that specifies an **axisymmetric_curve_2d_-element_representation**.

property: the cross section properties of an **axisymmetric_curve_2d_element_representation**.

angle_property: the properties for an axisymmetric section. Analysis applications often assume that the value angle attribute of this property is 2π.

material: the material properties associated with an **axisymmetric_curve_2d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: the **fea_model** shall be axisymmetric.

WR2: there shall be a property coordinate system associated with an **axisymmetric_curve_2d_element_representation**.

A property coordinate system (CURVE_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties.

WR3: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

FU1: there shall be the correct number of required nodes for a **curve_2d_element_representation**.

5.7.12 plane_curve_2d_element_representation

A **plane_curve_2d_element_representation** is an element with a specified cross section that is a 0D shape (a point) in the (**i**,**j**) plane of the founding coordinate system (see 5.2.3) with a depth perpendicular to the (**i**,**j**) plane. A **plane_curve_2d_element_representation** shall lie in the (**i**,**j**) plane of the founding coordinate system.

EXPRESS specification:

```
*)
ENTITY plane_curve_2d_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model_2d;
  element_descriptor               : plane_curve_2d_element_descriptor;
  property                         : curve_2d_element_property;
  depth_property                   : plane_2d_element_property;
  material                         : element_material;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: 'PLANAR' IN TYPEOF(model_ref.type_of_2d_analysis);
  WR2: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'CURVE_2D_ELEMENT_COORDINATE_SYSTEM'
             IN TYPEOF (item))) = 1;
  WR3: SIZEOF (QUERY(item1 <* material.properties |
        (SIZEOF (QUERY (item2 <*
        item1\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION'] * TYPEOF (item2)
                  ) >= 1
        )) = 1
       ))) >= 1;
  FU1:  required_0d_nodes (
        SELF\element_representation.required_node_list);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the plane curve 2d element.

element_descriptor: a collection of information that specifies **plane_curve_2d_element_representation**.

property: the cross section properties of a **plane_curve_2d_element_representation**.

depth_property: the properties for a planar section. Analysis applications often assume that the value of the depth attribute of this property is 1.0.

material: the material properties associated with a **plane_curve_2d_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: the **fea_model** shall be planar.

WR2: there shall be a property coordinate system associated with a **plane_curve_2d_element_representation**. A property coordinate system (CURVE_2D_ELEMENT_COORDINATE_SYSTEM) is the orientation of the coordinate system for the element material and section properties.

WR3: there shall be at least one of the following **fea_material_property_representation_item** subtypes referenced by an **material**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

FU1: there shall be the correct number of required nodes for a **curve_2d_element_representation**.

5.7.13 element_descriptor

An **element_descriptor** is a collection of information that specifies an element.

EXPRESS specification:

```
*)
ENTITY element_descriptor
  SUPERTYPE OF (ONEOF (volume_3d_element_descriptor,
                       axisymmetric_volume_2d_element_descriptor,
                       plane_volume_2d_element_descriptor,
                       surface_3d_element_descriptor,
                       axisymmetric_surface_2d_element_descriptor,
                       plane_surface_2d_element_descriptor,
                       curve_3d_element_descriptor,
                       axisymmetric_curve_2d_element_descriptor,
                       plane_curve_2d_element_descriptor));
  topology_order                    : element_order;
  description                       : text;
END_ENTITY;
(*
```

Attribute definitions:

topology_order: the highest degree polynomial interpolation function used to describe the geometric shape of any edge of an element, which in turn is used to relate the additional node list of the element to the appropriate topology diagram in 5.8.

description: additional information about the formulation or purpose of an element.

5.7.14 volume_3d_element_descriptor

A **volume_3d_element_descriptor** is a collection of information that specifies a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF
                                     volume_element_purpose;
  shape                            : volume_3d_element_shape;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **volume_3d_element_representation**.

shape: the geometric shape of a **volume_3d_element_representation**.

5.7.15 axisymmetric_volume_2d_element_descriptor

An **axisymmetric_volume_2d_element_descriptor** is a collection of information that specifies an axisymmetric volume 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_volume_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     volume_element_purpose;
  shape                            : element_2d_shape;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of an **axisymmetric_volume_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

shape: the geometric shape of an **axisymmetric_volume_2d_element_representation**

5.7.16 plane_volume_2d_element_descriptor

A **plane_volume_2d_element_descriptor** is a collection of information that specifies a plane volume 2D element.

EXPRESS specification:

```
*)
ENTITY plane_volume_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     volume_element_purpose;
  shape                            : element_2d_shape;
  assumption                       : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of **plane_volume_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

shape: the geometric shape of a **plane_volume_2d_element_representation**.

assumption: the use of a **plane_volume_2d_element_representation** with two dimensional shape to model a volume.

5.7.17 volume_3d_element_basis

A **volume_3d_element_basis** is the information that form the basis of a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_basis;
  descriptor                       : volume_3d_element_descriptor;
  variable                         : volume_variable;
  variable_order                   : element_order;
  variable_shape_function          : shape_function;
  evaluation_points                : LIST [1:?] OF
                                     volume_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **volume_3d_element_representation**.

variable: the variable to be associated with an order and shape function for a **volume_3d_element_representation**.

variable_order: the mathematical order of the polynomials used to define the **variable_shape_function**.

variable_shape_function: the type of polynomial shape function used to interpolate the variable within a **volume_3d_element_representation**.

evaluation_points: the locations within a **volume_3d_element_representation** where the **variable** is evaluated using the **variable_shape_function**.

5.7.18 volume_2d_element_basis

A **volume_2d_element_basis** is the information that form the basis of a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_basis;
  descriptor                      : volume_2d_element_descriptor;
  variable                        : volume_variable;
  variable_order                  : element_order;
  variable_shape_function         : shape_function;
  evaluation_points               : LIST [1:?] OF
                                    volume_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **volume_2d_element_representation**.

variable: the variable to be associated with an order and shape function for a **volume_2d_element_representation**.

variable_order: the mathematical order of the polynomials used to define the **variable_shape_function**.

variable_shape_function: the type of polynomial shape function used to interpolate the variable within a **volume_2d_element_representation**.

evaluation_points: the locations within a **volume_2d_element_representation** where the **variable** is evaluated using the **variable_shape_function**.

5.7.19 surface_3d_element_descriptor

A **surface_3d_element_descriptor** is a collection of information that specifies a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     surface_element_purpose;
  shape                            : element_2d_shape;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **surface_3d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

shape: the geometric shape of a **surface_3d_element_representation**.

5.7.20 axisymmetric_surface_2d_element_descriptor

An **axisymmetric_surface_2d_element_descriptor** is a collection of information that specifies an axisymmetric surface 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_surface_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     surface_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of an **axisymmetric_surface_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

5.7.21 plane_surface_2d_element_descriptor

A **plane_surface_2d_element_descriptor** is a collection of information that specifies a plane surface 2D element.

EXPRESS specification:

```
*)
ENTITY plane_surface_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     surface_element_purpose;
  assumption                       : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **plane_surface_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

assumption: the use of a curve element to model a surface.

5.7.22 surface_3d_element_basis

A **surface_3d_element_basis** is the information that form the basis of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_basis;
  descriptor                      : surface_3d_element_descriptor;
  variable                        : surface_element_variable;
  variable_order                  : element_order;
  variable_shape_function         : shape_function;
  evaluation_points               : LIST [1:?] OF
                                    surface_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **surface_3d_element_representation**.

variable: the variable to be associated with an order and shape function for a **surface_3d_element_representation**.

variable_order: the mathematical order of the polynomials used to define the **variable_shape_function**.

variable_shape_function: the type of polynomial shape function used to interpolate the variable within a **surface_3d_element_representation**.

evaluation_points: the locations within a **surface_3d_element_representation** where the **variable** is evaluated using the **variable_shape_function**.

5.7.23 surface_2d_element_basis

A **surface_2d_element_basis** is the information that form the basis of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_basis;
  descriptor                      : surface_2d_element_descriptor;
  variable                        : surface_element_variable;
```

```
  variable_order                  : element_order;
  variable_shape_function         : shape_function;
  evaluation_points               : LIST [1:?] OF
                                    surface_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **surface_2d_element_representation**.

variable: the variable to be associated with an order and shape function for a **surface_2d_element_representation**.

variable_order: the mathematical order of the polynomials used to define the **variable_shape_function**.

variable_shape_function: the type of polynomial shape function used to interpolate the variable within a **surface_2d_element_representation**.

evaluation_points: the locations within a **surface_2d_element_representation** where the **variable** is evaluated using the **variable_shape_function**.

5.7.24 curve_3d_element_descriptor

A **curve_3d_element_descriptor** is a collection of information that specifies a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                         : SET [1:?] OF SET [1:?] OF
                                    curve_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **curve_3d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

5.7.25 axisymmetric_curve_2d_element_descriptor

An **axisymmetric_curve_2d_element_descriptor** is a collection of information that specifies an axisymmetric curve 2D element.

EXPRESS specification:

```
*)
ENTITY axisymmetric_curve_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     curve_element_purpose;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of an **axisymmetric_curve_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

5.7.26 plane_curve_2d_element_descriptor

A **plane_curve_2d_element_descriptor** is a collection of information that specifies a plane curve 2D element.

EXPRESS specification:

```
*)
ENTITY plane_curve_2d_element_descriptor
  SUBTYPE OF (element_descriptor);
  purpose                          : SET [1:?] OF SET [1:?] OF
                                     curve_element_purpose;
  assumption                       : plane_2d_element_assumption;
END_ENTITY;
(*
```

Attribute definitions:

purpose: the enumerated value specifying the response of a **plane_curve_2d_element_representation**. The purpose is a SET of SETs in order to allow the appropriate combinations of coupled and uncoupled responses.

assumption: the use of the element to model a curve.

5.7.27 curve_3d_element_basis

A **curve_3d_element_basis** is the information that form the basis of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_element_basis;
  descriptor                       : curve_3d_element_descriptor;
  variable                         : curve_element_variable;
  variable_order                   : element_order;
  variable_shape_function          : shape_function;
  evaluation_points                : LIST [1:?] OF
                                     curve_element_location;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **curve_3d_element_representation**.

variable: the variable to be associated with an order and shape function for a **curve_3d_element_representation**.

variable_order: the mathematical order of the polynomials used to define the **variable_shape_function**.

variable_shape_function: the type of polynomial shape function used to interpolate the variable within a **curve_3d_element_representation**.

evaluation_points: the locations within a **curve_3d_element_representation** where the **variable** is evaluated using the **variable_shape_function**.

5.7.28 curve_2d_element_basis

A **curve_2d_element_basis** is the information that forms the basis of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_basis;
  descriptor                      : curve_2d_element_descriptor;
  variable                        : curve_element_variable;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **curve_2d_element_representation**.

variable: the variable to be associated with an order and shape function for a **curve_2d_element_representation**.

5.7.29 point_element_representation

A **point_element_representation** is a single noded explicit element.

EXPRESS specification:

```
*)
ENTITY point_element_representation
  SUBTYPE OF (element_representation);
  model_ref                       : fea_model;
  description                     : text;
  matrix_set                      : SET [1:?] OF point_element_matrix;
UNIQUE
  UR1: model_ref, SELF\representation.name;
```

```
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
              'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
              'FEA_AXIS2_PLACEMENT_3D'
              IN TYPEOF (item))) = 1;
  FU1:  required_0d_nodes (
        SELF\element_representation.required_node_list);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the point element.

description: information about the **point_element** purpose.

matrix_set: the explicit element matrices.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: there shall be exactly one coordinate system associated with a **point_element_representation**.

A matrix coordinate system (FEA_AXIS2_PLACEMENT_3D) is the coordinate system with respect to which all of the matrices of the point element are defined. The degrees of freedom at a node are defined with respect to the local orthogonal triad of the specified coordinate system at that node.

FU1: there shall be the correct number of required nodes for a **point_element_representation**.

5.7.30 point_element_matrix

A **point_element_matrix** is a matrix that represents a stationary mass, grounded spring, or grounded damper.

EXPRESS specification:

```
*)
ENTITY point_element_matrix
  SUPERTYPE OF (ONEOF(stationary_mass,
                      grounded_spring,
                      grounded_damper));
END_ENTITY;
(*
```

5.7.31 stationary_mass

A **stationary_mass** is a point element nodal response matrix of type stationary mass. The **stationary_mass** matrix is populated as follows:

$$\begin{pmatrix} m_x & & & & & \\ & m_y & & & & \\ & & m_z & & & \\ & & & i_{xx} & i_{xy} & i_{xz} \\ & & & i_{xy} & i_{yy} & i_{yz} \\ & & & i_{xz} & i_{yz} & i_{zz} \end{pmatrix}$$

NOTE – In all physical cases mass acts equally in all directions.

EXPRESS specification:

```
*)
ENTITY stationary_mass
  SUBTYPE OF (point_element_matrix);
  mass                            : ARRAY [1:3] OF context_dependent_measure;
  moments_of_inertia              : symmetric_tensor2_3d;
  coordinate_system               : fea_axis2_placement_3d;
  offset_vector                   : ARRAY[1:3] OF length_measure;
END_ENTITY;
(*
```

Attribute definitions:

mass: the point masses in the different directions is populated as follows:

Array item 1 is the mass acting in the **x** direction, and is denoted m_x in the matrix shown above;
Array item 2 is the mass acting in the **y** direction, and is denoted m_y in the matrix shown above;
Array item 3 is the mass acting in the **z** direction, and is denoted m_z in the matrix shown above;

moments_of_inertia: the moments of inertia of a point (denoted by i_{xx} ... i_{zz} in the matrix shown above) in the different directions. The values of the inertias should be set to 0.0 if unwanted. The inertias are evaluated about the center of mass.

coordinate_system: the coordinate system which is used to specify the offset.

offset_vector: the distance from the node to the position of the center of mass with respect to the local orthogonal triad of the specified coordinate system at the node.

5.7.32 grounded_spring

A **grounded_spring** is a point element nodal response matrix of type grounded spring. The **grounded_spring** matrix is populated as follows:

$$\begin{pmatrix} kt_x & & & & & \\ & kt_y & & & & \\ & & kt_z & & & \\ & & & kr_x & & \\ & & & & kr_y & \\ & & & & & kr_z \end{pmatrix}$$

EXPRESS specification:

```
*)
ENTITY grounded_spring
  SUBTYPE OF (point_element_matrix);
  stiffness_coefficients          : ARRAY [1:6] OF context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

stiffness_coefficients: the stiffnesses in the different directions is populated as follows:

Array item 1 is the translational stiffness in the **x** direction, denoted kt_x in the matrix shown above.
Array item 2 is the translational stiffness in the **y** direction, denoted kt_y in the matrix shown above.
Array item 3 is the translational stiffness in the **z** direction, denoted kt_z in the matrix shown above.
Array item 4 is the rotational stiffness about the **x** direction, denoted kr_x in the matrix shown above.
Array item 5 is the rotational stiffness about the **y** direction, denoted kr_y in the matrix shown above.
Array item 6 is the rotational stiffness about the **z** direction, denoted kr_z in the matrix shown above.

5.7.33 grounded_damper

A **grounded_damper** is a point element nodal response matrix of type grounded damper. The **grounded_damper** matrix is populated as follows:

$$\begin{pmatrix} dt_x & & & & & \\ & dt_y & & & & \\ & & dt_z & & & \\ & & & dr_x & & \\ & & & & dr_y & \\ & & & & & dr_z \end{pmatrix}$$

EXPRESS specification:

```
*)
ENTITY grounded_damper
  SUBTYPE OF (point_element_matrix);
  damping_coefficients            : ARRAY [1:6] OF context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

damping_coefficients: the damping in the different directions is populated as follows:

Array item 1 is the translational damping in the **x** direction, denoted dt_x in the matrix shown above.
Array item 2 is the translational damping in the **y** direction, denoted dt_y in the matrix shown above.
Array item 3 is the translational damping in the **z** direction, denoted dt_z in the matrix shown above.
Array item 4 is the rotational damping about the **x** direction, denoted dr_x in the matrix shown above.
Array item 5 is the rotational damping about the **y** direction, denoted dr_y in the matrix shown above.
Array item 6 is the rotational damping about the **z** direction, denoted dr_z in the matrix shown above.

5.7.34 directionally_explicit_element_representation

A **directionally_explicit_element_representation** is a two noded element for which the matrices (stiffness, damping and mass) are characterized by a single coefficient **k**. The **x** direction of the matrix is defined with respect to a coordinate system.

All elements of the matrix are populated from the single coefficient **k**. The matrix is of the form:

$$\begin{array}{cc} \begin{pmatrix} +k & -k \\ \\ -k & +k \end{pmatrix} & \begin{array}{l} \cdots \text{ degree of freedom 1, node 1} \\ \\ \cdots \text{ degree of freedom 2, node 2} \end{array} \\ \begin{array}{cc} \vdots & \vdots \\ \text{degree of} & \text{degree of} \\ \text{freedom 1,} & \text{freedom 2,} \\ \text{node 1} & \text{node 2} \end{array} & \end{array}$$

Different degrees of freedom can be specified for the two nodes. For a simple axial spring or damper the **x** translation degree of freedom in the aligned system is specified for both nodes. A two node explicit element can be used instead of a directionally explicit element, but an

explicit element is less convenient because it is necessary to define each of the four matrix terms separately. Also, an aligned coordinate system cannot be defined for an explicit element.

EXPRESS specification:

```
*)
ENTITY directionally_explicit_element_representation
  SUBTYPE OF (element_representation);
  model_ref                  : fea_model;
  systems_and_freedoms       : LIST [2:2] OF system_and_freedom;
  description                : text;
  coefficient_set            : directionally_explicit_element_coefficient;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  FU1:  required_1d_nodes (
        SELF\element_representation.required_node_list);
  FU2:  additional_1d_nodes (
        SELF\higher_order_element_representation.additional_node_list,
        linear);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the directionally explicit element.

systems_and_freedoms: the coordinate systems and degrees of freedom with respect to which all of the matrices of the **directionally_explicit_element_representation** are defined. List item 1 specifies the coordinate system and degree of freedom at node 1 and list item 2 specifies the coordinate system and degree of freedom at node 2.

description: information about the purpose of a **directionally_explicit_element_representation**.

coefficient_set: the coefficients which define the matrices of a **directionally_explicit_element_representation**.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

FU1: there shall be the correct number of required nodes for a **directionally_explicit_element_representation**.

FU2: there shall be the correct number of additional nodes for **directionally_explicit_element_representation**.

5.7.35 system_and_freedom

A **system_and_freedom** is the coordinate system and degree of freedom for a **directionally_-explicit_element_representation** node.

EXPRESS specification:

```
*)
ENTITY system_and_freedom;
  matrix_coordinate_system    :
                     directionally_explicit_element_coordinate_system;
  freedom                     : degree_of_freedom;
END_ENTITY;
(*
```

Attribute definitions:

matrix_coordinate_system: the coordinate system with respect to which all of the matrices of the **directionally_explicit_element_representation** are defined for a node. This system can be an aligned system derived form the positions of the element nodes, or it can be an arbitrary system. An aligned system shall not be specified if the two nodes of the element are coincident.

freedom: the degree of freedom at a node of a **directionally_explicit_element_-representation**.

5.7.36 directionally_explicit_element_coefficient

A **directionally_explicit_element_coefficient** is the coefficients which define the matrices of the directionally explicit element.

EXPRESS specification:

```
*)
ENTITY directionally_explicit_element_coefficient;
  property_type                    : matrix_property_type;
  coefficient                      : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

property_type: the enumerated value specifying the type of matrix (stiffness, damping or mass).

coefficient: the coefficient which defines the matrix.

5.7.37 explicit_element_representation

An **explicit_element_representation** is an element in which the associated matrices (stiffness, mass, and damping) are specified directly.

EXPRESS specification:

```
*)
ENTITY explicit_element_representation
  SUBTYPE OF (element_representation);
  model_ref                        : fea_model;
  description                      : text;
  matrix_set                       : explicit_element_matrix;
UNIQUE
  UR1: model_ref, SELF\representation.name;
WHERE
  WR1: SIZEOF (QUERY(item <* SELF\representation.items |
             'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.' +
             'FEA_AXIS2_PLACEMENT_3D)'
             IN TYPEOF (item))) = 1;
  WR2:  (SIZEOF (matrix_set.node_dof_list) = SIZEOF (
        SELF\element_representation.required_node_list)) AND
        (SIZEOF (matrix_set.matrix_values) =
        number_of_terms (matrix_set.node_dof_list, matrix_set.symmetry));
  FU1:  additional_0d_nodes (
        SELF\higher_order_element_representation.additional_node_list);
END_ENTITY;
(*
```

Attribute definitions:

model_ref: an application defined identifier of the **fea_model** which possesses the explicit element.

description: information about an **explicit_element_representation** purpose.

matrix_set: the explicit element matrices.

NOTE – The element id, a unique application defined identifier of an **element_representation**, is specified by the **name** attribute of the **representation** supertype.

Formal propositions:

UR1: the combination of an **fea_model** and element id shall be unique within a finite element analysis model.

WR1: there shall be exactly one matrix coordinate system associated with an **explicit_element_representation**.

A matrix coordinate system (FEA_AXIS2_PLACEMENT_3D) is the coordinate system with respect to which all of the matrices of an **explicit_element_representation** are defined. The degrees of freedom at a node are defined with respect to the local orthogonal triad of the specified coordinate system at that node.

WR2: the length of the degree of freedom and required node lists shall be the same, and the length of the matrix values list shall be appropriate for the size and symmetry of the matrix.

FU1: there shall be no additional nodes for an **explicit_element**.

5.7.38 explicit_element_matrix

An **explicit_element_matrix** is a matrix of an **explicit_element_representation**.

EXPRESS specification:

```
*)
ENTITY explicit_element_matrix;
  property_type                    : matrix_property_type;
  symmetry                         : matrix_symmetry;
  node_dof_list                    : LIST [1:?] OF LIST [1:?] OF
                                     degree_of_freedom;
  matrix_values                    : LIST [1:?] OF context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

property_type: the type of matrix (stiffness, mass, or damping).

symmetry: the symmetry of the matrix (symmetric or diagonal).

node_dof_list: the degree of freedom at the nodes. The first list shall have the dimension of the number of **explicit_element_representation** nodes, the second list shall have the dimension of the number of freedoms for that node.

matrix_values: the coefficients which define the matrices of the **explicit_element_representation**.

NOTE – The matrix is square with freedoms in the sequence specified by the node_dof_list. The number of independent terms in the matrix and their position in the list depends on the attribute **matrix_symmetry**. Refer to the **matrix_symmetry** TYPE for the definitions of how these matrices are populated.

5.8 Structural response representation schema definitions: Element topologies

The relationship between a node and the element geometry is determined by its position in the element node lists. This relationship depends on the element shape and order, and is shown below for each element shape and order.

The orientation of the parametric coordinate system has been defined for each type of element. This orientation has the relationship to the nodes referenced from the element node lists as shown in the element node sequence diagrams below. The sequence of vertex nodes has the same relationship to the orientation of the parametric coordinate system for each order of a particular element shape, hence for clarity of presentation the orientation of the parametric coordinate system of an element is only shown in the diagram for the linear element of a particular shape. The axes of the element parametric coordinate system are denoted by (ξ,η,ζ). The values of (ξ,η,ζ) shall range from -1.0 to 1.0.

Triangular, wedge, tetrahedron, and pyramid elements are defined with collapsed rectangular coordinate systems. The figures for these elements denote which node is the singularity by

specifying which edge of the element an axis of the parametric coordinate system is parallel to.

The sequence of the nodes in the element node lists has been defined for each type of element. In each case the following classes of nodes are in the sequences:

- Vertex nodes. These nodes are required for the definition of the element geometry and shall be specified.

- Edge nodes. These nodes are additional and may be omitted.

- Face nodes. These nodes are additional and may be omitted.

- Body nodes. These nodes are additional and may be omitted.

References to nodes shall be held in the element nodes lists such that:

- The vertex nodes are referenced from the **required_node_list** attribute of the **element_representation** entity where the position in the list is the number in the diagram.

- Any additional nodes (edge, face or body) are referenced by the **additional_node_list** attribute of the **higher_order_element** entity where the position in the list is governed by:

  position = (number in diagram) - (number of vertex nodes)

- Any missing additional edge, face, or body nodes are specified by referencing **dummy_node** entities.

  For each element SUBTYPE the WHERE clause ensures that all the required vertex nodes are supplied, and that only those edge, face, and body nodes are supplied for which a relationship to the element geometry is defined.

The relationship of element faces, and for surface elements also edges, is determined by the order of the vertex nodes. The element node sequence, face, and edge diagrams follow in figures 10 through 39:

Figure 10 – Curve 3D or surface 2D elements - linear

Figure 11 – Curve 3D or surface 2D elements - quadratic

Figure 12 – Curve 3D or surface 2D elements - cubic

Figure 13 – Surface 2D elements - faces and edges

Figure 14 – Surface 3D or volume 2D elements - triangle - linear

Figure 15 – Surface 3D or volume 2D elements - triangle - quadratic

Figure 16 – Surface 3D or volume 2D elements - triangle - cubic

Figure 17 – Surface 3D elements - triangle faces and edges

Figure 18 – Volume 2D elements - triangle faces

Figure 19 – Surface 3D or volume 2D elements - quadrilateral - linear

Figure 20 – Surface 3D or volume 2D elements - quadrilateral - quadratic

Figure 21 – Surface 3D or volume 2D elements - quadrilateral - cubic

Figure 22 – Surface 3D elements - quadrilateral faces and edges

Figure 23 – Volume 2D elements - quadrilateral faces

Figure 24 – Volume 3D elements - hexahedron - linear

Figure 25 – Volume 3D elements - hexahedron - quadratic

Figure 26 – Volume 3D elements - hexahedron - cubic

Figure 27 – Volume 3D elements - hexahedron faces

Figure 28 – Volume 3D elements - wedge - linear

Figure 29 – Volume 3D elements - wedge - quadratic

Figure 30 – Volume 3D elements - wedge - cubic

Figure 31 – Volume 3D elements - wedge faces

Figure 32 – Volume 3D elements - tetrahedron - linear

Figure 33 – Volume 3D elements - tetrahedron - quadratic

Figure 34 – Volume 3D elements - tetrahedron - cubic

Figure 35 – Volume 3D elements - tetrahedron faces

Figure 36 – Volume 3D elements - pyramid - linear

Figure 37 – Volume 3D elements - pyramid - quadratic

Figure 38 – Volume 3D elements - pyramid - cubic

Figure 39 – Volume 3D elements - pyramid faces

5.9 Structural response representation schema entity definitions: Element coordinate systems

The coordinate systems that are used for element information fall into three different categories:

- An arbitrary Placement Coordinate System which is defined without reference to the geometry of the element. Such a system can be used to orient property information associated with a volume 3D element.

- An aligned orthogonal system in which the orientation of the axes for a surface or curve element coordinate system at each point in the element is partly defined by the element geometry.

- A parametric coordinate system at each point within an element is derived from the orientation of the parametric axes for the element at that point. The parametric axes are not necessarily orthogonal. Such a system can be used when a curved surface of orthotropic or anisotropic material is modeled by volume or surface 3D elements.

If a non-cartesian coordinate system is specified for an element then information at a point within an element is defined with respect to the local orthogonal cartesian coordinate system whose axes are coincident to the specified system at that point. An aligned system at a point is also derived from an orthogonal cartesian coordinate system whose axes are coincident to the specified coordinate system and the element geometry at that point.

Element information may be defined with respect to an orthogonal system of coordinate axes derived from the parametric coordinate system of the element. A parametric coordinate system orientation is established graphically for each element type along with the element node sequence in the figures in 5.8. In this clause only two axes for each orthogonal coordinate system are specified. The third axis is then chosen according to right hand rule to complete the triad. The axes of an orthogonal element coordinate system are denoted (**x**,**y**,**z**). The orthogonal system may be an intermediate orthogonal system, with the element orthogonal system related to the intermediate orthogonal coordinate system by one (for surface elements) or three (for volume elements) angles.

Finite element information is defined with respect to an orthogonal coordinate system that shall obey the following rules:

- For a surface element, the **z** axis of the coordinate system is defined to remain normal to the element surface. Hence for a non-planar (curved) element the orientation of the coordinate system varies over the element surface. The **z** axis of a surface element is normal to the surface in both 2D and 3D analyses.

- For a curve element, the **x** axis of the coordinate system is defined to be tangential to the curve. Hence for a curved element the orientation of the coordinate system varies along the curve.

5.9.1 aligned_axis_tolerance

An **aligned_axis_tolerance** is a value for the tolerance of the alignment of coordinate systems

EXPRESS specification:

```
*)
ENTITY aligned_axis_tolerance;
  model_ref                        : fea_model;
  tolerance                        : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

model_ref: the model to which the tolerance applies.

tolerance: the value of the alignment tolerance.

5.9.2 arbitrary_volume_3d_element_coordinate_system

An **arbitrary_volume_3d_element_coordinate_system** is an arbitrary orthogonal coordinate system for a volume 3D element.

EXPRESS specification:

```
*)
ENTITY arbitrary_volume_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  coordinate_system                : fea_axis2_placement_3d;
END_ENTITY;
(*
```

Attribute definitions:

coordinate_system: the coordinate system for the volume 3D element. At a point within the element information is defined with respect to the local orthogonal coordinate system triad of the specified coordinate system at that point.

5.9.3 parametric_volume_3d_element_coordinate_system

A **parametric_volume_3d_element_coordinate_system** is the orthogonal coordinate system for a volume 3D element. At each point in the element an intermediate orthogonal coordinate system is derived from the parametric coordinate system of an element. Let the triad (ξ,η,ζ) denote the axes of the parametric coordinate system, and $(\mathbf{x}',\mathbf{y}',\mathbf{z}')$ denote the axis of an intermediate orthogonal coordinate system. The orthogonal coordinate system used to define information for the element, denoted $(\mathbf{x},\mathbf{y},\mathbf{z})$, is related to this intermediate orthogonal coordinate system $(\mathbf{x}',\mathbf{y}',\mathbf{z}')$ by Euler angles.

For the purpose of reference by attributes **axis_1** and **axis_2**, let $\xi = 1$, $\eta = 2$ and, $\zeta = 3$. The intermediate orthogonal coordinate system shall be derived from the parametric coordinate system by the various combinations of **axis_1** and **axis_2** as follows:

– **axis_1** = 1 and **axis_2** = 2:

$$\begin{aligned} \mathbf{x}' &= \xi \\ \mathbf{z}' &= \langle \xi \times \eta \rangle \end{aligned}$$

– **axis_1** = 2 and **axis_2** = 3:

$$\begin{aligned} \mathbf{y}' &= \eta \\ \mathbf{x}' &= \langle \eta \times \zeta \rangle \end{aligned}$$

– **axis_1** = 3 and **axis_2** = 1:

$$\begin{aligned} \mathbf{z}' &= \zeta \\ \mathbf{y}' &= \langle \zeta \times \xi \rangle \end{aligned}$$

– **axis_1** = 1 and **axis_2** = 3:

$$\begin{aligned} \mathbf{x}' &= \xi \\ \mathbf{y}' &= \langle \zeta \times \xi \rangle \end{aligned}$$

– **axis_1** = 2 and **axis_2** = 1:

$$\begin{aligned} \mathbf{y}' &= \eta \\ \mathbf{z}' &= \langle \xi \times \eta \rangle \end{aligned}$$

– **axis_1** = 3 and **axis_2** = 2:

$$\begin{aligned} \mathbf{z}' &= \zeta \\ \mathbf{x}' &= \langle \eta \times \zeta \rangle \end{aligned}$$

The derivation of an orthogonal coordinate system for **axis_1** = 1 and **axis_2** = 2 is shown in figure 40 for a linear hexahedron element. In this example, the (**x**,**y**,**z**) system is not rotated from the (**x**′,**y**′,**z**′) system and is therefore coincident. The **z** and **z**′ axes are perpendicular to the $\xi\eta$ plane. The **y** and **y**′ axes are perpendicular to the **x**′**z**′ and **xz** planes, positive right-hand rule.

EXPRESS specification:

```
*)
ENTITY parametric_volume_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  axis_1                          : INTEGER;
  axis_2                          : INTEGER;
  angles                          : euler_angles;
WHERE
  WR1:  (axis_1 >= 1) AND (axis_1 <= 3) AND
        (axis_2 >= 1) AND (axis_2 <= 3) AND
        NOT (axis_1 = axis_2);
END_ENTITY;
(*
```

Figure 40 – Orthogonal volume 3D element coordinate system derivation for axis_1 = 1 and axis_2 = 2

Attribute definitions:

axis_1: the first parametric axis used to derive the orthogonal coordinate system.

axis_2: the second parametric axis used to derive the orthogonal coordinate system.

angles: three Euler angles that define the orthogonal element coordinate system ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$) with respect to ($\mathbf{x}'$,$\mathbf{y}'$,$\mathbf{z}'$).

Formal propositions:

WR1: the two defining axes of the coordinate system shall be 1, 2, or 3, and they shall not be the same.

5.9.4 arbitrary_volume_2d_element_coordinate_system

An **arbitrary_volume_2d_element_coordinate_system** specifies an arbitrary orthogonal coordinate system for a volume 2D element, for which the **x** axis shall be normal to the 2D analysis plane and in the direction of the **k** axis of the 2D analysis plane definition coordinate system.

The orientation of the coordinate systems is shown in figure 41. The **j** axis is the axis of symmetry, and the **ij** plane is the 2d analysis plane. The **yz** plane is coincident with the **ij** plane. The **y** axis is oriented by the vector **direction**.

EXPRESS specification:

```
*)
ENTITY arbitrary_volume_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  orientation                      : direction;
WHERE
  WR1: SELF\geometric_representation_item.dim=2;
END_ENTITY;
(*
```

Attribute definitions:

orientation: the direction used to orient the orthogonal coordinate system at each point within the element.

Formal propositions:

WR1: the space dimensionality of the direction shall be 2.

5.9.5 parametric_volume_2d_element_coordinate_system

A **parametric_volume_2d_element_coordinate_system** is an orthogonal coordinate system for a volume 2D element. An intermediate orthogonal coordinate system ($\mathbf{x}'$,$\mathbf{y}'$,$\mathbf{z}'$) is derived from the parametric coordinate system (ξ,η) of an element. The $\mathbf{z}'$ axis of the intermediate orthogonal coordinate system is normal to the 2D analysis plane and in the direction of the **k** axis of the 2D analysis plane definition coordinate system. An axis of the intermediate orthogonal coordinate system in the 2D analysis plane shall be derived according to the **axis** attribute as follows:

Figure 41 – Arbitrary volume 2D element coordinate system orientation

- For **axis** = 1 the $\mathbf{y}'$ axis of the intermediate orthogonal system is on the same direction as the axis ξ of the parametric coordinate system.

- For **axis** = 2 the $\mathbf{z}'$ axis of the intermediate orthogonal system is in the same direction as the axis η of the parametric system.

The orthogonal coordinate system defined for the element, denoted ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$) is related to the intermediate orthogonal system ($\mathbf{x}'$,$\mathbf{y}'$,$\mathbf{z}'$) by an angle.

The derivation of an orthogonal coordinate system for axis = 1 is shown in figure 42. In this example, the ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$) system is not rotated from the ($\mathbf{x}'$,$\mathbf{y}'$,$\mathbf{z}'$) system and is therefore coincident. The $\mathbf{x}$ and $\mathbf{x}'$ axes are perpendicular to the $\xi\eta$ plane. The $\mathbf{z}$ and $\mathbf{z}'$ axes are perpendicular to the $\mathbf{x}'\mathbf{y}'$ and $\mathbf{xy}$ planes, positive right-hand rule.

Figure 42 – Parametric volume 2D element coordinate system derivation for axis = 1

EXPRESS specification:

```
*)
ENTITY parametric_volume_2d_element_coordinate_system
```

```
  SUBTYPE OF (fea_representation_item);
  axis                              : INTEGER;
  angle                             : plane_angle_measure;
WHERE
  WR1:  (axis >= 1) AND (axis <= 2);
END_ENTITY;
(*
```

Attribute definitions:

axis: the axis of the parametric and intermediate coordinate systems that are aligned.

angle: the angle from the $\mathbf{x}'$ to the $\mathbf{x}$ axis measured in a positive sense about the $\mathbf{z}'$ axis.

Formal propositions:

WR1: the defining axis of the coordinate system shall be either 1 or 2.

5.9.6 aligned_surface_3d_element_coordinate_system

An **aligned_surface_3d_element_coordinate_system** is an aligned orthogonal coordinate system for a surface 3D element which is derived from the local orthogonal triad of the specified arbitrary coordinate system and the normal to the surface of an element. Let the triad ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$) denote the axes of the aligned orthogonal system, and let ($\mathbf{X}$,$\mathbf{Y}$,$\mathbf{Z}$) denote the local orthogonal triad of the specified arbitrary system. At each point in the element the $\mathbf{z}$ axis of the aligned orthogonal system shall be normal to the surface of an element such that:

$$\mathbf{z} \cdot \mathbf{Z} > \mathbf{0}$$

The $\mathbf{y}$ axis of the aligned orthogonal system is such that:

$$\mathbf{y} = \langle \mathbf{z} \times \mathbf{X} \rangle$$

Hence the $\mathbf{X}$ axis is projected on the surface of the element to form the $\mathbf{x}$ axis of the aligned orthogonal system. It is important that the aligned orthogonal coordinate system be well conditioned. To ensure this $\mathbf{Z}$ shall be specified such that:

$$\mathbf{z} \cdot \mathbf{Z} > \mathbf{tolerance}\,|\mathbf{z}|\,|\mathbf{Z}|$$

where the tolerance is specified by an **aligned_axis_tolerance** entity for the model.

The derivation of a surface 3D element aligned orthogonal coordinate system from an arbitrary coordinate system is shown in figure 43.

EXPRESS specification:

```
*)
ENTITY aligned_surface_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  coordinate_system                 : fea_axis2_placement_3d;
END_ENTITY;
(*
```

Figure 43 – Aligned surface 3D element coordinate system derivation

Attribute definitions:

coordinate_system: the coordinate system for the element.

5.9.7 parametric_surface_3d_element_coordinate_system

A **parametric_surface_3d_element_coordinate_system** is an orthogonal coordinate system for a surface 3D element. An intermediate orthogonal coordinate system (ξ,η,ζ) at any point on the element is derived from the parametric coordinate system. This intermediate orthogonal system shall be derived from the parametric system according to the value of **axis** attribute as follows:

- For **axis** = 1 the $\mathbf{x}'$ axis of the intermediate orthogonal system is in the same direction as the axis ξ of the parametric coordinate system.

- For **axis** = 2 the $\mathbf{y}'$ axis of the intermediate orthogonal system is in the same direction as the axis η of the parametric coordinate system.

The $\mathbf{z}'$ intermediate orthogonal axis shall be normal to the surface in the direction of the ζ parametric coordinate system axis. The orthogonal coordinate system defined for the element, denoted ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$) is related to the intermediate orthogonal coordinate system ($\mathbf{x}'$,$\mathbf{y}'$,$\mathbf{z}'$) by the **angle** attribute.

The derivation of an orthogonal coordinate system for axis = 1 is shown in figure 44. In this example, the ($\mathbf{x}$,$\mathbf{y}$,$\mathbf{z}$) system is not rotated from the ($\mathbf{x}'$,$\mathbf{y}'$,$\mathbf{z}'$) system and is therefore coincident. The $\mathbf{z}$ and $\mathbf{z}'$ axes are perpendicular to the $\xi\eta$ plane. The $\mathbf{y}$ and $\mathbf{y}'$ axes are perpendicular to the $\mathbf{x}'\mathbf{z}'$ and $\mathbf{xz}$ planes, positive right-hand rule.

EXPRESS specification:

```
*)
ENTITY parametric_surface_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  axis                              : INTEGER;
  angle                             : plane_angle_measure;
WHERE
  WR1:  (axis >= 1) AND (axis <= 2);
END_ENTITY;
(*
```

Attribute definitions:

axis: the axis of the parametric and intermediate orthogonal coordinate systems that are aligned.

angle: the angle from the $\mathbf{x}'$ to the $\mathbf{x}$ axis measured in a positive sense about the $\mathbf{z}'$ axis.

Formal propositions:

WR1: the defining axis of the coordinate system shall be either 1 or 2.

Figure 44 – Parametric surface 3D element coordinate system derivation for axis = 1

5.9.8 constant_surface_3d_element_coordinate_system

A **constant_surface_3d_element_coordinate_system** is an aligned orthogonal coordinate system for a surface 3D element, and is defined for each point on the surface as follows:

- An intermediate orthogonal coordinate system is derived from the parametric coordinate system at the centroid of the element in the way specified for the **parametric_surface_3d_element_coordinate_system**;

- an aligned orthogonal coordinate system is derived from this intermediate orthogonal system in the way specified for entity **aligned_surface_3d_element_coordinate_system**.

EXPRESS specification:

```
*)
ENTITY constant_surface_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  axis                              : INTEGER;
  angle                             : plane_angle_measure;
WHERE
  WR1:  (axis >= 1) AND (axis <= 2);
END_ENTITY;
(*
```

Attribute definitions:

axis: the axis of the parametric and intermediate orthogonal coordinate systems that are aligned.

angle: the angle from the $\mathbf{x}'$ to the $\mathbf{x}$ axis measured in a positive sense about the $\mathbf{z}'$ axis.

Formal propositions:

WR1: the defining axis of the coordinate system shall be either 1 or 2.

5.9.9 aligned_surface_2d_element_coordinate_system

An **aligned_surface_2d_element_coordinate_system** is an aligned orthogonal coordinate system for a surface 2D element.

For the **aligned_surface_2d_element_coordinate_system** the $\mathbf{x}$ axis shall be normal to the 2D analysis plane and in the direction of the $\mathbf{k}$ axis of the 2D analysis plane. At each point in the element the $\mathbf{z}$ axis of the aligned orthogonal coordinate system shall be normal to the surface of an element in the $\mathbf{z}\cdot > \mathbf{0}$ direction, and $\mathbf{y} = \mathbf{z} \times \mathbf{x}$ such that:

$$\mathbf{z} \cdot \mathbf{d} > \mathbf{0}$$

where $\mathbf{d}$ is the specified direction.

It is important that the aligned orthogonal coordinate system be well conditioned. To ensure this, $\mathbf{d}$ shall be specified such that:

$$\mathbf{d} \cdot \mathbf{x} > \mathbf{tolerance}\,|\mathbf{d}|\,|\mathbf{x}|$$

where the tolerance is specified by an **aligned_axis_tolerance** entity for the model.

The derivation of a surface 2D element aligned coordinate system is shown in figure 45.

Figure 45 – Aligned surface 2D element coordinate system derivation

EXPRESS specification:

```
*)
ENTITY aligned_surface_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  orientation                      : direction;
WHERE
  WR1: SELF\geometric_representation_item.dim=2;
END_ENTITY;
(*
```

Attribute definitions:

orientation: the direction used to orient the orthogonal coordinate system at each point within the element.

Formal propositions:

WR1: the space dimensionality of the direction shall be 2.

5.9.10 parametric_surface_2d_element_coordinate_system

A **parametric_surface_2d_element_coordinate_system** is an orthogonal coordinate system for a surface 2D element. The **y** orthogonal axis shall be in the direction of the parametric axis ξ. The **x** orthogonal axis is normal to the 2D analysis plane and in the direction of the **k** axis of the 2D analysis plane definition system. The derivation of an orthogonal coordinate system is shown in figure 46.

Figure 46 – Parametric surface 2D element coordinate system derivation

EXPRESS specification:

```
*)
ENTITY parametric_surface_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
END_ENTITY;
(*
```

5.9.11 aligned_curve_3d_element_coordinate_system

An **aligned_curve_3d_element_coordinate_system** is an aligned orthogonal coordinate system for a curve 3D element. Let the triad (**x**,**y**,**z**) denote the axes of the aligned orthogonal system and let (**X**,**Y**,**Z**) denote the local orthogonal triad of the specified arbitrary system. At each point in the element the **x** axis is tangential to the curve such that:

$$\mathbf{x} \cdot \mathbf{X} > \mathbf{0}$$

The **z** axis of the aligned orthogonal system is such that:

$$\mathbf{z} = \langle \mathbf{x} \times \mathbf{Y} \rangle$$

The derivation of the aligned orthogonal system needs to be well conditioned. To ensure this **X** shall be specified such that:

$$\mathbf{x} \cdot \mathbf{X} > \mathbf{tolerance}\,|\mathbf{x}|\,|\mathbf{X}|$$

where the tolerance is specified by an **aligned_axis_tolerance** entity for the model.

The derivation of a curve 3D element aligned orthogonal coordinate system is shown in figure 47.

EXPRESS specification:

```
*)
ENTITY aligned_curve_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  coordinate_system                : fea_axis2_placement_3d;
END_ENTITY;
(*
```

Attribute definitions:

coordinate_system: the coordinate system for the element.

5.9.12 parametric_curve_3d_element_coordinate_system

A **parametric_curve_3d_element_coordinate_system** is an orthogonal coordinate system that is derived from the parametric coordinate system for a curve 3D element. Let the triad (**x**,**y**,**z**) denote axes of the orthogonal system and let **d** denote the specified direction. The **x** orthogonal axis is tangential to the curve in the direction of the parametric axis ξ. The **z** orthogonal axis is defined such that:

$$\mathbf{z} = \langle \mathbf{x} \times \mathbf{d} \rangle$$

The derivation of the orthogonal system needs to be well conditioned. To ensure this **d** shall be specified such that:

Figure 47 – Aligned curve 3D element coordinate system derivation

$$\mathbf{y} \cdot \mathbf{d} > \mathbf{tolerance}\,|\mathbf{y}|\,|\mathbf{d}|$$

where the tolerance is specified by an **aligned_axis_tolerance** entity for the model.

The derivation of an orthogonal coordinate system is shown in figure 48.

Figure 48 – Parametric curve 3D element coordinate system derivation

EXPRESS specification:

```
*)
ENTITY parametric_curve_3d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  direction                        :
                  parametric_curve_3d_element_coordinate_direction;
END_ENTITY;
(*
```

Attribute definitions:

direction: a direction used to orient the orthogonal coordinate system at each point in the element.

5.9.13 parametric_curve_3d_element_coordinate_direction

A **parametric_curve_3d_element_coordinate_direction** is the parametric direction used to orient the orthogonal 3D coordinate system for a curve 3D element by the use of a direction.

EXPRESS specification:

```
*)
ENTITY parametric_curve_3d_element_coordinate_direction
  SUBTYPE OF (fea_representation_item);
  orientation                      : direction;
WHERE
  WR1: SELF\geometric_representation_item.dim=3;
END_ENTITY;
(*
```

Attribute definitions:

orientation: a direction used to orient the orthogonal coordinate system at each point in the element.

Formal propositions:

WR1: the space dimensionality of the direction shall be 3.

5.9.14 curve_2d_element_coordinate_system

A **curve_2d_element_coordinate_system** is an orthogonal coordinate system for a curve 2D element. The **x** axis is tangential to the curve in the direction of the **k** axis of the 2D analysis plane. The **y** axis of the aligned orthogonal system shall be in the specified direction. The derivation of an orthogonal coordinate system is shown in figure 49.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_coordinate_system
  SUBTYPE OF (fea_representation_item);
  orientation                      : direction;
WHERE
  WR1: SELF\geometric_representation_item.dim=2;
END_ENTITY;
(*
```

Attribute definitions:

orientation: the direction of the **y** axis, which is defined in the 2D analysis plane, used to orient the orthogonal coordinate system at each point in the element.

Formal propositions:

WR1: the space dimensionality of the direction shall be 2.

Figure 49 – Curve 2D element coordinate system derivation

5.9.15 directionally_explicit_element_coordinate_system

A **directionally_explicit_element_coordinate_system** is a directionally explicit element coordinate system that shall be either an aligned or arbitrary coordinate system that is used for orienting property information associated with a directionally explicit element.

EXPRESS specification:

```
*)
TYPE directionally_explicit_element_coordinate_system = SELECT
  (directionally_explicit_element_coordinate_system_arbitrary,
   directionally_explicit_element_coordinate_system_aligned);
END_TYPE;
(*
```

5.9.16 directionally_explicit_element_coordinate_system_-arbitrary

A **directionally_explicit_element_coordinate_system_arbitrary** is an arbitrary orthogonal coordinate system for a directionally explicit element.

EXPRESS specification:

```
*)
ENTITY directionally_explicit_element_coordinate_system_arbitrary
  SUBTYPE OF (fea_representation_item);
  arbitrary_system                : fea_axis2_placement_3d;
END_ENTITY;
(*
```

Attribute definitions:

arbitrary_system: an arbitrary coordinate system. Information is defined with respect to the local orthogonal triad of the specified system. The information is evaluated at the nodes of an element.

5.9.17 directionally_explicit_element_coordinate_system_-aligned

A **directionally_explicit_element_coordinate_system_aligned** is an aligned orthogonal coordinate system for orienting a directionally explicit element.

EXPRESS specification:

```
*)
ENTITY directionally_explicit_element_coordinate_system_aligned
  SUBTYPE OF (fea_representation_item);
  aligned_system                  : curve_3d_element_coordinate_system;
END_ENTITY;
(*
```

Attribute definitions:

aligned_system: a curve 3D element coordinate system. The information is evaluated at the nodes of an element. A coordinate system is defined for the directionally explicit element as if it were a linear curve element.

5.9.18 euler_angles

An **euler_angles** is used to define a coordinate system relative to a defining coordinate system. Typically the defining coordinate system will be an element coordinate system, and the relative coordinate system used to define anisotropic material directions.

EXPRESS specification:

```
*)
ENTITY euler_angles;
  euler_angles                    : ARRAY [1:3] OF plane_angle_measure;
END_ENTITY;
(*
```

Attribute definitions:

euler_angles: the relative orientation of a new (rotated) coordinate system with respect to a defining coordinate system using a series of three consecutive rotations. This method is known as Euler angles. The original coordinate system **x1**, **y1**, **z1** is rotated about the **z1** axis over angle euler_angles[1], with the angle positive right hand rule. This yields coordinate system **x2**, **y2**, **z2**. Coordinate system **x2**, **y2**, **z2** is rotated about the **y2** axis over angle euler_angles[2], with the angle positive right hand rule. This yields coordinate system **x3**, **y3**, **z3**. Coordinate system **x3**, **y3**, **z3** is rotated about the **x3** axis over angle euler_angles[3], positive right hand rule. This yields the final coordinate system.

5.10 Structural response representation schema entity definitions: Element matrix integration

A finite element is typically composed of many different matrices. Two common examples of these matrices are mass and stiffness. In some of the more complex elements the stiffness matrix may be partitioned into several matrices. Each of these matrices typically needs to be integrated, mostly over the element area or volume. Many of these matrices are so mathematically complex that explicit integration is not done, and, therefore, numerical integration techniques are used. For each matrix which is integrated, the element integrated matrix definition can specify the interpolation rule for the quantity within the element, and the method of integration of the quantity. An element may have different interpolation rules for different matrices. Diagonal (lumped) matrices are often used for element mass and damping. Such matrices are not treated as a special case, but are specified by numerical integration points at the nodes of an element.

As described in 5.8, a volume, surface or curve element has shape functions for geometric interpolation. The definitions of these shape functions depend on the element type, and in some element formulations, the actual element node coordinates and normals. The element descriptor entity for the particular element type specifies the element shape (e.g. triangle or quadrilateral), the element order, and a text description that can be used to make the element geometric shape functions definitions precise. All locations within an element are specified using the parametric

axes system derived from the element shape functions for geometric interpolation. However, the interpolation of other quantities within an element need not use the same shape functions. Each quantity may have a different interpolation rule which is defined by the shape functions and the element geometry. The element shape function specifies the interpolation order for the quantity and a text description which can be used to make the shape function definitions for the quantity precise.

NOTE – For volume, surface, and curve elements it is possible to define an explicit numerical integration by an arbitrary selection of integration points and weights within the volume of the element. This is shown in figure 50 for volume elements.

Figure 50 – Volume element integration points

However for surface and curve elements it is more common to define field and section integration methods separately where **field integration** is integration over the surface or along the length of an element, and **section integration** is the integration through the thickness of a surface element or over the cross section of a curve element. The separate field and section integration definitions are used even when each point of a numerical field is using the same integration method. For linear analyses, section integrations are usually carried out algebraically; thus only one integration point would be defined through the section.

NOTE – Figure 51 illustrates the location of integration points in the field of a surface element and through the section (thickness) of the element. The section positions are with respect to the entire section depth, which is the sum of all the plies in a layered material. A series of extent searches needs to be made to correlate integration points with a given ply. See the Fundamental Concepts and Assumptions subclause of this clause for discussion on why this modeling method was selected.

Figure 51 – Surface element integration points

There are three choices of defining the location of an integration point within a volume element.

NOTE – The EXPRESS-G Partial Model shown in figure 52 is presented as an aid in visualising the integration options for volume elements.

The algebraic integration option does not have integration points as this option defines that the integration is done exactly. The rule option defines that an integration rule (such as Gaussian) and integration order is used to integrate the matrix. The explicit option is the most general numerical integration option where the integration positions and weights are explicitly defined. This option is included to avoid having to catalogue all the many types of integration rules. The explicit integration option will form the foundation for the description of analysis output points when that portion of this standard is complete.

Figure 52 – Volume element integration EXPRESS-G partial model

The way in which the location of an integration point is defined within a surface or curve element depends on whether surface-section or surface-field integration is being used.

NOTE – The curve and surface element integration EXPRESS-G Partial Model is shown in figure 53 to aid in visualising the integration options available.

Figure 53 – Surface and curve element integration EXPRESS-G partial model

For surface-section integration there are no integration points within an element as the integration for both surface and section is carried out exactly.

For surface-field integration the location of an integration point within an element is defined by both a field location specified using the element parametric coordinate system and a section location with respect to the element coordinate system. In a surface 3D element a field location has two parametric coordinates, a surface 2D element or curve 3D element has a single parametric coordinate, and a field location is undefined for a 2D curve element. The field integration can be carried out by either algebraic (exact integration), rule or explicit integration options. The definition of these options is identical to that presented in the discussion on volume elements above.

In a surface element a section location is defined by the distance from the reference plane of

the element in the direction of the **z** axis of the surface element property coordinate system. In a curve element a section location is defined by the distances from the reference axis of the element in the y and z directions of the curve element property coordinate system. A section location is defined using physical distances, not unit distances. The field and section axes when used in combination to define a location within a surface or curve element do not necessarily form a right hand triad.

The following paragraphs are included as a reference for the theoretical basis of the integration referenced in this standard.

The numerical integration of a quantity within the volume of an element requires the evaluation of that quantity at one or more points within the element. The integral is a weighted sum of these values. The location of the integration points and the weighting assigned to them are defined either by a numerical integration rule (e.g. Gauss's or Simpson's) or by an explicit list of locations and weights.

The integration rules specify points within a unit space, usually within the range of -1.0 to 1.0. If section integration by rule is specified then these unit distances are converted to physical distances separately for each field integration point. Hence section integration by rule can only be specified for a surface with a defined thickness.

The form of the summation and the weighting depend of the type of the element. The specific forms for volume, surface and curve elements follow:

5.10.1 Volume 3D Element

For a volume 3D element, the integral quantity can be obtained numerically by a summation of the form:

$$Q = \sum_{i=1}^{n} j_i \, w_i \, q_i$$

where:

n is the number of points within the volume;

q_i is the value of the quantity at point i;

w_i is the weighting for the evaluation at point i;

j_i is the determinant of the Jacobian matrix at point i.

This form of integration may also be used for a surface 3D element or curve 3D element, however it is more usual to use separate field and section integrations for these elements.

5.10.2 Volume 2D Element

For a volume 2D element, the integral quantity can be obtained numerically by a summation of the form:

$$Q = \sum_{i=1}^{n} s_i \, j_i \, w_i \, q_i$$

where:

n is the number of points within the volume;

q_i is the value of the quantity at point i;

w_i is the weighting for the evaluation at point i;

j_i is the determinant of the Jacobian matrix at point i;

s_i is either the depth of section for a plane stress or strain element or the distance from the axis of symmetry for an axisymmetric element, of point i.

This form of integration may also be used for a surface 2D element, however it is more usual to use separate field and section integrations for these elements.

5.10.3 Curve 2D Element

For a curve 2D element, the integral quantity can be obtained numerically by a summation of the form:

$$Q = \sum_{i=1}^{n} s_i \, w_i \, q_i$$

where:

n is the number of points within the volume;

q_i is the value of the quantity at point i;

w_i is the weighting for the evaluation at point i;

s_i is either the depth of section for a plane stress or strain element or the distance from the axis of symmetry for an axisymmetric element, of point i.

5.10.4 Surface 3D or Curve Element

For a surface 3D or curve element with separate integration the integral quantity can be obtained numerically by a summation of the form:

$$Q = \sum_{i=1}^{n} j_i \, w_i^f \, Q_i^s$$

where Q_i^s is the integral of the quantity through the section at field point i, which can be obtained numerically by a summation of the form:

$$Q_i^s = \sum_{j=1}^{m} w_j^s \, q_{ji}$$

where:

n is the number of points over the field;

m is the number of points through the section;

q_{ji} is the value of the quantity at section point j and field point i;

w_i^f is the weighting for the evaluation of the field integration at point i;

w_j^s is the weighting for the evaluation of the section integration at point j;

j_i is the determinant of the Jacobian matrix at field point i.

5.10.5 Surface 2D Element

For a surface 2D element with separate integration the integral quantity can be obtained numerically by a summation of the form:

$$Q = \sum_{i=1}^{n} j_i \, w_i^f \, Q_i^s$$

where Q_i^s is the integral of the quantity through the section at field point i, which can be obtained numerically by a summation of the form:

$$Q_i^s = \sum_{j=1}^{m} s_{ji} \, w_j^s \, q_{ji}$$

where:

n is the number of points over the field;

m is the number of points through the section;

q_{ji} is the value of the quantity at section point j and field point i;

s_{ji} is either the depth of section for a plane stress or strain element or the distance from the axis of symmetry for an axisymmetric element, of section point j at field point i.

w_i^f is the weighting for the evaluation of the field integration at point i;

w_j^s is the weighting for the evaluation of the section integration at point j;

j_i is the determinant of the Jacobian matrix at field point i.

5.10.6 volume_3d_element_integrated_matrix

A volume_3d_element_integrated_matrix is the matrix to be integrated for a volume 3D element, and the method of integration.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_integrated_matrix;
  descriptor                      : volume_3d_element_descriptor;
  matrix_property_type            : matrix_property_type;
  integration_description         : text;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **volume_3d_element_element_representation**.

matrix_property_type: the type of matrix being evaluated.

integration_description: the interpolation rule and integration method.

5.10.7 volume_3d_element_integrated_matrix_with_definition

A volume_3d_element_integrated_matrix_with_definition is the method of integration.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_integrated_matrix_with_definition
  SUBTYPE OF (volume_3d_element_integrated_matrix);
  integration_definition          : volume_3d_element_field_integration;
END_ENTITY;
(*
```

Attribute definitions:

integration_definition: a definition of the integration within the 3D volume.

5.10.8 volume_3d_element_field_integration

A **volume_3d_element_field_integration** is a volume 3D field integration shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE volume_3d_element_field_integration = SELECT
  (element_integration_algebraic,
   volume_3d_element_field_integration_rule,
   volume_3d_element_field_integration_explicit);
END_TYPE;
(*
```

5.10.9 element_integration_algebraic

An **element_integration_algebraic** is an element integration that is exact; therefore, no numerical integration information is required.

EXPRESS specification:

```
*)
TYPE element_integration_algebraic = ENUMERATION OF (algebraic);
END_TYPE;
(*
```

5.10.10 volume_3d_element_field_integration_rule

A **volume_3d_element_field_integration_rule** is the integration rule and order for a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_field_integration_rule;
  integration_rule                : integration_rule;
  integration_order               : ARRAY [1:3] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: the integration rule for the quantity being integrated.

integration_order: the order of the specified rule for the quantity being integrated. A separate integration order is specified for each parametric axis direction established graphically in 5.8 in the sequence (ξ,η,ζ).

5.10.11 volume_3d_element_field_integration_explicit

A **volume_3d_element_field_integration_explicit** is the explicit numerical integration for a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_field_integration_explicit;
  integration_positions_and_weights: SET [1:?] OF volume_position_weight;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: the integration positions for the quantity being integrated, and the corresponding weights for each integration position.

5.10.12 volume_position_weight

A **volume_position_weight** is an integration position within a volume element, and its weighting factor.

EXPRESS specification:

```
*)
ENTITY volume_position_weight;
  integration_position            : volume_element_location;
  integration_weight              : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

integration_position: the integration position for the quantity being integrated.

integration_weight: the weight for the integration position.

5.10.13 volume_2d_element_integrated_matrix

A **volume_2d_element_integrated_matrix** is the matrix to be integrated for a volume 2D element, and the method of integration.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_integrated_matrix;
  descriptor                      : volume_2d_element_descriptor;
  matrix_property_type            : matrix_property_type;
  integration_description         : text;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **volume_2d_element_representation**.

matrix_property_type: the type of matrix being evaluated.

integration_description: the interpolation rule and integration method.

5.10.14 volume_2d_element_integrated_matrix_with_definition

A **volume_2d_element_integrated_matrix_with_definition** is the method of integration.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_integrated_matrix_with_definition
  SUBTYPE OF (volume_2d_element_integrated_matrix);
  integration_definition          : volume_2d_element_field_integration;
END_ENTITY;
(*
```

Attribute definitions:

integration_definition: a definition of the integration within the 2D volume.

5.10.15 volume_2d_element_field_integration

A **volume_2d_element_field_integration** is a volume 2D field integration that shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE volume_2d_element_field_integration = SELECT
  (element_integration_algebraic,
   volume_2d_element_field_integration_rule,
   volume_2d_element_field_integration_explicit);
END_TYPE;
(*
```

5.10.16 volume_2d_element_field_integration_rule

A **volume_2d_element_field_integration_rule** is the integration rule and order for a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_field_integration_rule;
  integration_rule                : integration_rule;
  integration_order               : ARRAY [1:2] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: the integration rule for the quantity being integrated.

integration_order: the order of the specified rule for the quantity being integrated. A separate integration order is specified for each parametric axis direction established graphically in 5.8 in the sequence (ξ,η).

5.10.17 volume_2d_element_field_integration_explicit

A **volume_2d_element_field_integration_explicit** is the explicit numerical integration for a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_field_integration_explicit;
  integration_positions_and_weights: SET [1:?] OF volume_position_weight;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: the integration positions for the quantity being integrated, and the corresponding weights for each integration position.

5.10.18 surface_3d_element_integrated_matrix

A **surface_3d_element_integrated_matrix** is the matrix to be integrated for a surface 3D element, and the method of integration.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_integrated_matrix;
  descriptor                      : surface_3d_element_descriptor;
  matrix_property_type            : surface_matrix_property_type;
  integration_description         : text;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **surface_3d_element_representation**.

matrix_property_type: the type of matrix being evaluated.

integration_description: the interpolation rule and integration method.

5.10.19 surface_3d_element_integrated_matrix_with_definition

A **surface_3d_element_integrated_matrix_with_definition** is the method of integration.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_integrated_matrix_with_definition
  SUBTYPE OF (surface_3d_element_integrated_matrix);
  integration_definition          : surface_3d_element_integration;
END_ENTITY;
(*
```

Attribute definitions:

integration_definition: a definition of the integration within the 3D surface.

5.10.20 surface_3d_element_integration

A **surface_3d_element_integration** is the method of integration for the field and section of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_integration;
  field                           : surface_3d_element_field_integration;
  section                         : surface_section_integration;
END_ENTITY;
(*
```

Attribute definitions:

field: the integration of the quantity being integrated over the field (surface) of the element.

section: the integration of the quantity being integrated through the section (thickness) of the element.

5.10.21 surface_3d_element_field_integration

A **surface_3d_element_field_integration** is a surface 3D field integration that shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE surface_3d_element_field_integration = SELECT
  (element_integration_algebraic,
   surface_3d_element_field_integration_rule,
   surface_3d_element_field_integration_explicit);
END_TYPE;
(*
```

5.10.22 surface_section_integration

A **surface_section_integration** is a surface 3D section integration that shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE surface_section_integration = SELECT
  (element_integration_algebraic,
   surface_section_integration_rule,
   surface_section_integration_explicit);
END_TYPE;
(*
```

5.10.23 surface_3d_element_field_integration_rule

A **surface_3d_element_field_integration_rule** is the integration rule and order for a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_field_integration_rule;
  integration_rule                 : integration_rule;
  integration_order                : ARRAY [1:2] OF INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: the integration rule for the quantity being integrated.

integration_order: the order of the specified rule for the quantity being integrated. A separate integration order is specified for each parametric axis direction established graphically in 5.8 in the sequence (ξ,η).

5.10.24 surface_3d_element_field_integration_explicit

A **surface_3d_element_field_integration_explicit** is the explicit numerical integration for a surface 3D element field.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_field_integration_explicit;
  integration_positions_and_weights: SET [1:?] OF surface_position_weight;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: the integration positions for the quantity being integrated, and the corresponding weights for each integration position.

5.10.25 surface_position_weight

A **surface_position_weight** is an integration position within a surface element, and its weighting factor.

EXPRESS specification:

```
*)
ENTITY surface_position_weight;
  integration_position            : surface_element_location;
  integration_weight              : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

integration_position: the integration position for the quantity being integrated.

integration_weight: the weight for the integration position.

5.10.26 surface_section_integration_rule

A **surface_section_integration_rule** is the integration rule and order through the section for a surface element.

EXPRESS specification:

```
*)
ENTITY surface_section_integration_rule;
```

```
  integration_rule                : integration_rule;
  integration_order               : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: the integration rule for the quantity being integrated through the surface element section.

integration_order: the order of the specified rule for the quantity being integrated through the surface element section.

5.10.27 surface_section_integration_explicit

A **surface_section_integration_explicit** is the explicit numerical integration for a surface element section.

EXPRESS specification:

```
*)
ENTITY surface_section_integration_explicit;
  integration_position_and_weights : SET [1:?] OF
                                     surface_section_position_weight;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: the integration positions and weights through the thickness of the section for the quantity being integrated.

5.10.28 surface_section_position_weight

A **surface_section_position_weight** is the explicit numerical integration for a position in a surface element section.

EXPRESS specification:

```
*)
ENTITY surface_section_position_weight;
  integration_position            : surface_section_element_location;
  integration_weight              : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

integration_position: the integration position through the thickness of the section for the quantity being integrated.

integration_weight: the weight for the integration position.

5.10.29 surface_2d_element_integrated_matrix

A **surface_2d_element_integrated_matrix** is the matrix to be integrated for a surface 2D element, and the method of integration.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_integrated_matrix;
  descriptor                      : surface_2d_element_descriptor;
  matrix_property_type            : surface_matrix_property_type;
  integration_description         : text;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **surface_2d_element_representation**.

matrix_property_type: the type of matrix being evaluated.

integration_description: interpolation rule and integration method.

5.10.30 surface_2d_element_integrated_matrix_with_definition

A **surface_2d_element_integrated_matrix_with_definition** is the method of integration.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_integrated_matrix_with_definition
  SUBTYPE OF (surface_2d_element_integrated_matrix);
  integration_definition          : surface_2d_element_integration;
END_ENTITY;
(*
```

Attribute definitions:

integration_definition: the integration within the surface.

5.10.31 surface_2d_element_integration

A **surface_2d_element_integration** is the method of integration for the length of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_integration;
  element_length                  : surface_2d_element_length_integration;
  section                         : surface_section_integration;
END_ENTITY;
(*
```

Attribute definitions:

element_length: the integration of the quantity being integrated along the length of the curve that defines the surface 2D element in the 2D analysis plane.

section: the integration of the quantity being integrated through the section (thickness) of the element.

5.10.32 surface_2d_element_length_integration

A **surface_2d_element_length_integration** is a surface 2D length integration that shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE surface_2d_element_length_integration = SELECT
  (element_integration_algebraic,
   surface_2d_element_length_integration_rule,
   surface_2d_element_length_integration_explicit);
END_TYPE;
(*
```

5.10.33 surface_2d_element_length_integration_rule

A **surface_2d_element_length_integration_rule** is the integration rule and order for a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_length_integration_rule;
  integration_rule                : integration_rule;
  integration_order               : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: the integration rule for the quantity being integrated.

integration_order: the order of the specified rule for the quantity being integrated.

5.10.34 surface_2d_element_length_integration_explicit

A **surface_2d_element_length_integration_explicit** is the explicit numerical integration for a surface 2D element length.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_length_integration_explicit;
  integration_positions_and_weights: SET [1:?] OF surface_position_weight;
```

```
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: the integration positions for the quantity being integrated, and the corresponding weights for each integration position.

5.10.35 curve_3d_element_integrated_matrix

A **curve_3d_element_integrated_matrix** is the matrix to be integrated for a curve 3D element, and the method of integration.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_integrated_matrix;
  descriptor                      : curve_3d_element_descriptor;
  matrix_property_type            : curve_matrix_property_type;
  integration_description         : text;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **curve_3d_element_representation**.

matrix_property_type: the type of matrix being evaluated.

integration_description: the interpolation rule and integration method.

5.10.36 curve_3d_element_integrated_matrix_with_definition

A **curve_3d_element_integrated_matrix_with_definition** is the method of integration.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_integrated_matrix_with_definition
  SUBTYPE OF (curve_3d_element_integrated_matrix);
  integration_definition          : curve_3d_element_integration;
END_ENTITY;
(*
```

Attribute definitions:

integration_definition: a definition of the integration within the curve.

5.10.37 curve_3d_element_integration

A **curve_3d_element_integration** the method of integration for the length and section of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_integration;
  element_length                  : curve_3d_element_length_integration;
  section                         : curve_section_integration_explicit;
END_ENTITY;
(*
```

Attribute definitions:

element_length: the integration of the quantity being integrated along the length of the element.

section: the integration of the quantity being integrated over the section of the element.

5.10.38 curve_3d_element_length_integration

A **curve_3d_element_length_integration** is a curve 3D length integration that shall be either algebraic, by rule, or explicit.

EXPRESS specification:

```
*)
TYPE curve_3d_element_length_integration = SELECT
  (element_integration_algebraic,
   curve_3d_element_length_integration_rule,
   curve_3d_element_length_integration_explicit);
END_TYPE;
(*
```

5.10.39 curve_3d_element_length_integration_rule

A **curve_3d_element_length_integration_rule** is the integration rule and order for a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_length_integration_rule;
  integration_rule                : integration_rule;
  integration_order               : INTEGER;
END_ENTITY;
(*
```

Attribute definitions:

integration_rule: the integration rule for the quantity being integrated.

integration_order: the order of the specified rule for the quantity being integrated.

5.10.40 curve_3d_element_length_integration_explicit

A **curve_3d_element_length_integration_explicit** is the explicit numerical integration for a curve 3D element length.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_length_integration_explicit;
  integration_positions_and_weights: SET [1:?] OF
                                     curve_3d_element_position_weight;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions_and_weights: the integration positions for the quantity being integrated, and the corresponding weights for each integration position.

5.10.41 curve_3d_element_position_weight

A **curve_3d_element_position_weight** is an integration position within a 3D curve element, and its weighting factor.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_position_weight;
  integration_position             : curve_volume_element_location;
  integration_weight               : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

integration_position: the integration positions for the quantity being integrated.

integration_weight: the weight for each integration position.

5.10.42 curve_section_integration_explicit

A **curve_section_integration_explicit** is the explicit numerical integration for a curve element cross section.

EXPRESS specification:

```
*)
ENTITY curve_section_integration_explicit;
  integration_positions            : SET [1:?] OF
                                     curve_section_element_location;
END_ENTITY;
(*
```

Attribute definitions:

integration_positions: the integration positions across the section for the quantity being integrated.

5.10.43 curve_2d_element_integrated_matrix

A **curve_2d_element_integrated_matrix** is the matrix to be integrated for a curve 2D element, and the method of integration.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_integrated_matrix;
  descriptor                      : curve_2d_element_descriptor;
  matrix_property_type            : curve_matrix_property_type;
  integration_description         : text;
END_ENTITY;
(*
```

Attribute definitions:

descriptor: the association to the information describing a **curve_2d_element_representation**.

matrix_property_type: the type of matrix being evaluated.

integration_description: the interpolation rule and integration method.

5.10.44 curve_2d_element_integrated_matrix_with_definition

A **curve_2d_element_integrated_matrix_with_definition** is the method of integration.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_integrated_matrix_with_definition
  SUBTYPE OF (curve_2d_element_integrated_matrix);
  integration_definition          : curve_2d_element_integration;
END_ENTITY;
(*
```

Attribute definitions:

integration_definition: a definition of the integration within the curve.

5.10.45 curve_2d_element_integration

A **curve_2d_element_integration** is the method of integration for the section of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_integration;
```

```
  section                          : LIST [1:?] OF
                                     curve_section_element_location;
END_ENTITY;
(*
```

Attribute definitions:

section: a definition of the integration of the quantity being integrated over the section of the element.

5.11 Structural response representation schema entity definitions: Element locations

A field variable may be evaluated for a list of locations within an element. This list of locations is specified by reference to the entity within this section, which is appropriate to the element type.

5.11.1 fea_parametric_point

An **fea_parametric_point** is a position within a finite element, and specifies its parametric coordinates ξ, η and ζ.

EXPRESS specification:

```
*)
ENTITY fea_parametric_point
  SUBTYPE OF (point);
  coordinates                 :  LIST[1:3] OF parameter_value;
WHERE
  WR1: valid_parametric_coordinate (coordinates);
  WR2:  (NOT( 'GEOMETRY_SCHEMA.CARTESIAN_POINT'   IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_ON_CURVE'    IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_ON_SURFACE'  IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.DEGENERATE_PCURVE' IN (TYPEOF(SELF)))) AND
        (NOT( 'GEOMETRY_SCHEMA.POINT_REPLICA'     IN (TYPEOF(SELF)))) AND
        (NOT( 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.SPHERICAL_POINT'
                                                  IN (TYPEOF(SELF)))) AND
        (NOT( 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.CYLINDRICAL_POINT'
              IN (TYPEOF(SELF))));
END_ENTITY;
(*
```

Attribute definitions:

coordinates: the coordinates of the position. The first value of the coordinate is the ξ coordinate, the second the η coordinate, and the third the ζ coordinate (if applicable).

Formal propositions:

WR1: each coordinate shall lie in the range [-1.0 , +1.0].

WR2: only the SUBTYPE of **fea_parametric_point** shall exist in this instance.

5.11.2 volume_element_location

A **volume_element_location** is a location within a volume element, and specifies its parametric coordinates ξ, η and ζ.

EXPRESS specification:

```
*)
ENTITY volume_element_location;
  coordinates                  :  fea_parametric_point;
END_ENTITY;
(*
```

Attribute definitions:

coordinates: the coordinates of the location. The first value of the coordinate is the ξ coordinate, the second the η coordinate, and the third the ζ coordinate (if applicable).

5.11.3 surface_volume_element_location

A **surface_volume_element_location** is a location within the volume of a surface element.

EXPRESS specification:

```
*)
ENTITY surface_volume_element_location;
  field_location               :  surface_element_location;
  section_location             :  surface_section_element_location;
END_ENTITY;
(*
```

Attribute definitions:

field_location: the location over the surface of the element.

section_location: the location through the thickness (section) of the element.

5.11.4 surface_element_location

A **surface_element_location** is a location on the surface of a surface element, that specifies its parametric coordinates ξ and, if appropriate, η.

EXPRESS specification:

```
*)
ENTITY surface_element_location;
  coordinates                  : fea_parametric_point;
END_ENTITY;
(*
```

Attribute definitions:

coordinates: the coordinates of the location. The first value of the pair is the ξ coordinate, and the second the η coordinate (if applicable).

5.11.5 surface_section_element_location

A **surface_section_element_location** is a location through the section of a surface element. A section location can be specified as an offset from the nominal plane of the surface, or as a dimensionless section coordinate.

Within a surface of a composite material, a field variable may be discontinuous at a material boundary. If a section location is specified at a material boundary, a flag can be set to indicate whether the value of the variable is to be evaluated above or below the discontinuity.

EXPRESS specification:

```
*)
ENTITY surface_section_element_location
  SUPERTYPE OF (ONEOF (surface_section_element_location_absolute,
                       surface_section_element_location_dimensionless));
  above_material_discontinuity :  LOGICAL;
END_ENTITY;
(*
```

Attribute definitions:

above_material_discontinuity: the side of a material discontinuity on which the field variable is to be evaluated.

If the value is TRUE, then the variable is evaluated on the side of the discontinuity in the direction of the positive **z** axis of the property coordinate system for the element.

If the value is FALSE, then the variable is evaluated on the other side of the discontinuity.

5.11.6 surface_section_element_location_absolute

A **surface_section_element_location_absolute** is a location through the section of a surface element that is specified by giving a distance from the nominal plane of the surface.

EXPRESS specification:

```
*)
ENTITY surface_section_element_location_absolute
  SUBTYPE OF (surface_section_element_location);
  offset                       :  length_measure;
END_ENTITY;
(*
```

Attribute definitions:

offset: the distance of the location from the nominal plane of the element.

A positive value indicates an offset in the direction of the positive **z** axis of the property coordinate system for the element.

5.11.7 surface_section_element_location_dimensionless

A **surface_section_element_location_dimensionless** is a location through the section of a surface element that is specified by giving a dimensionless section coordinate.

EXPRESS specification:

```
*)
ENTITY surface_section_element_location_dimensionless
  SUBTYPE OF (surface_section_element_location);
  coordinate                  : LIST [1:1] OF parameter_value;
WHERE
  WR1: valid_parametric_coordinate (coordinate);
END_ENTITY;
(*
```

Attribute definitions:

coordinate: the section coordinate of the location. A section coordinate of 1.0 indicates the surface of the element in the direction of the positive **z** axis of the property coordinate system for the element. A section coordinate of -1.0 indicates the other surface.

Formal propositions:

WR1: the section coordinate shall be within the range [-1.0, +1.0].

5.11.8 curve_volume_element_location

A **curve_volume_element_location** is a location within the volume of a curve element.

EXPRESS specification:

```
*)
ENTITY curve_volume_element_location;
  field_location              :  curve_element_location;
  section_location            :  curve_section_element_location;
END_ENTITY;
(*
```

Attribute definitions:

field_location: the location along the length of the element.

section_location: the location through the section of the element.

5.11.9 curve_element_location

A **curve_element_location** is a location along the length of a curve element, that is specified by its parametric coordinate ξ.

EXPRESS specification:

```
*)
ENTITY curve_element_location;
  coordinate                  :  fea_parametric_point;
```

```
END_ENTITY;
(*
```

Attribute definitions:

coordinates: the coordinate of the location. The value is the ξ coordinate.

5.11.10 curve_section_element_location

A **curve_section_element_location** is a location within the section of a curve element. A section location is specified by giving distances from the nominal axis of the curve.

EXPRESS specification:

```
*)
ENTITY curve_section_element_location;
  offsets                      :  ARRAY[1:2] of length_measure;
END_ENTITY;
(*
```

Attribute definitions:

offsets: the distances of the location from the nominal axis of the element.

A positive value for the first coordinate indicates an offset in the direction of the positive **y** axis of the property coordinate system for the element.

A positive value for the second coordinate item indicates an offset in the direction of the positive **z** axis of the property coordinate system for the element.

5.12 Structural response representation schema entity definitions: Finite element analysis material properties

The material specifications and properties for an element are referenced through a set of **fea_material_property_representation** entities. The material specifications are inherited from entities associated with a **material_property_representation**, which in turn may have an **fea_material_property_representation** subtype.

NOTE – See figure 54 for a perspective on how the **fea_material_property_representation** entity associates finite element analysis model information with entities from ISO 10303-45.

EXAMPLE 1 – Each **fea_material_property_representation** is defined with respect to an **environment** entity that is specified by a reference from an instantiation of a **material_property_representation** and the associated material product **representation** related entities. The environment variables (e.g. temperature, moisture content) are associated to the **material_property_representation** by the set of **property_definition_representation** entities referenced by the **environment** entity. There is a single FEA material property representation item for each material property and associated environment variables. per where rule WR1 in the **fea_material_property_representation** entity. An **element_material** aggregates together **fea_material_property_representation** entities to represent all the material properties necessary to define an element.

Figure 54 – Relationship between FEA and ISO 10303-45

5.12.1 element_material

An **element_material** is the aggregation of the material properties of the continuum that a finite element represents.

EXPRESS specification:

```
*)
ENTITY element_material;
  material_id           : identifier;
  description           : text;
  properties            : SET[1:?] OF material_property_representation;
END_ENTITY;
(*
```

Attribute definitions:

material_id: a unique application defined identifier for the material to be used during the finite element analysis.

description: a text narrative about the substance.

properties: the properties of the material properties for a finite element.

Informal propositions:

IP1: the combination of **fea_model** and **material_id** shall be unique within a finite element analysis model.

5.12.2 fea_material_property_representation

An **fea_material_property_representation** is a property of the substance of the continuum that a finite element represents. The property is defined with respect to the data environment specified in the **material_property_representation** supertype.

NOTE – For layered composite material properties for surface elements the total thickness for the element is defined by adding up all the ply thicknesses in the material. The plies are defined by a defining Application Protocol.

EXPRESS specification:

```
*)
ENTITY fea_material_property_representation
  SUBTYPE OF (material_property_representation);
WHERE
  WR1: SIZEOF (QUERY (item <*
       SELF\property_definition_representation.used_representation.items |
         SIZEOF (['FEA_LINEAR_ELASTICITY',
                  'FEA_MASS_DENSITY',
                  'FEA_AREA_DENSITY',
                  'FEA_TANGENTIAL_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_SECANT_COEFFICIENT_OF_LINEAR_THERMAL_EXPANSION',
                  'FEA_MOISTURE_ABSORPTION',
                  'FEA_SHELL_MEMBRANE_STIFFNESS',
```

```
                  'FEA_SHELL_BENDING_STIFFNESS',
                  'FEA_SHELL_MEMBRANE_BENDING_COUPLING_STIFFNESS',
                  'FEA_SHELL_SHEAR_STIFFNESS'] * TYPEOF (item)
                  ) = 1

      )) = 1;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be exactly one of the following **fea_material_property_representation_item** subtypes associated with an **fea_material_property_representation**:

fea_linear_elasticity

fea_mass_density

fea_area_density

fea_tangential_coefficient_of_linear_thermal_expansion

fea_secant_coefficient_of_linear_thermal_expansion

fea_moisture_absorption

fea_shell_membrane_stiffness

fea_shell_bending_stiffness

fea_shell_membrane_bending_coupling_stiffness

fea_shell_shear_stiffness

Therefore there also shall be a one to one relationship between a referenced **fea_material_property_representation_item** subtype entity and the entity inherited from the **definition** attribute of the **property_definition_representation** supertype of an **fea_material_property_representation**.

5.12.3 fea_material_property_representation_item

An **fea_material_property_representation_item** is the representation of mechanical material properties suitably idealized for finite element analysis.

EXPRESS specification:

```
*)
ENTITY fea_material_property_representation_item
  SUPERTYPE OF (ONEOF (fea_linear_elasticity,
                       fea_mass_density,
```

```
                        fea_area_density,
                        fea_tangential_coefficient_of_linear_thermal_expansion,
                        fea_secant_coefficient_of_linear_thermal_expansion,
                        fea_moisture_absorption,
                        fea_shell_membrane_stiffness,
                        fea_shell_bending_stiffness,
                        fea_shell_membrane_bending_coupling_stiffness,
                        fea_shell_shear_stiffness))
  SUBTYPE OF (representation_item);
END_ENTITY;
(*
```

5.12.4 fea_linear_elasticity

An **fea_linear_elasticity** is the collection of mechanical properties representing a linear response of an **fea_material** to a small change in strain at a constant temperature. The response is a change in stress caused by a small change in strain as follows:

$$\Delta\sigma_{ij} = \sum_{i,j=1}^{3} d_{ij}\,\Delta\varepsilon_{ij}$$

where:

d: is the elasticity tensor;

$\Delta\sigma$: is a change in the stress tensor;

$\Delta\varepsilon$: is a change in the strain tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_linear_elasticity
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor4_3d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the elasticity properties.

5.12.5 fea_mass_density

An **fea_mass_density** is the mass of **fea_material** per unit volume of an element in a strain free state. The **fea_mass_density** is a scalar.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_mass_density
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constant                    : scalar;
END_ENTITY;
(*
```

Attribute definitions:

fea_constant: the value of the FEA mass density property.

5.12.6 fea_area_density

An **fea_area_density** is the mass of **fea_material** per unit area of a surface element in a strain free state. The **fea_area_density** is a scalar.

NOTE – This property is usually defined in a strain free state that has a reference temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_area_density
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constant                    : scalar;
END_ENTITY;
(*
```

Attribute definitions:

fea_constant: the value of the FEA area density property.

5.12.7 fea_tangential_coefficient_of_linear_thermal_expansion

An **fea_tangential_coefficient_of_linear_thermal_expansion** is the relationship of a change in thermal strain in an element to a a small change in temperature as follows:

$$\Delta\varepsilon_{ij} = \alpha_{ij}(T)\,\Delta T$$

where:

α: is the tangential coefficient of linear thermal expansion for the material at temperature T;

$\Delta\varepsilon$: is the change in the strain tensor caused by the small change in temperature at temperature T;

ΔT: is a change in temperature in the neighborhood of T in which the incremental thermal expansion is linear;

T: is the temperature.

An **fea_tangential_coefficient_of_linear_thermal_expansion** is a 3D second order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_tangential_coefficient_of_linear_thermal_expansion
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor2_3d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA tangential coefficient of linear thermal expansion tensor.

5.12.8 fea_secant_coefficient_of_linear_thermal_expansion

An **fea_secant_coefficient_of_linear_thermal_expansion** is the relationship of the thermal strain of an element caused by a change from a reference temperature to a specified temperature as follows:

$$\varepsilon_{ij} = \alpha_{ij}\,(T - T_o)$$

where:

$\alpha(T)$**:** is the secant coefficient of linear thermal expansion for the material at temperature T;

ε**:** is the thermal strain caused by a change in temperature from its reference value T_o to T;

T_o**:** is the reference temperature;

T**:** is the temperature.

The secant and the tangential coefficients of linear thermal expansion are different if the coefficients are temperature dependent. An **fea_secant_coefficient_of_linear_thermal_expansion** is a 3D second order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_secant_coefficient_of_linear_thermal_expansion
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor2_3d;
  reference_temperature            : thermodynamic_temperature_measure;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA secant coefficient of linear thermal expansion tensor.

reference_temperature: the value of the reference temperature.

5.12.9 fea_moisture_absorption

An **fea_moisture_absorption** is the relationship of a change in moisture strain in an element to a small change in moisture as follows:

$$\Delta\varepsilon_{ij} = \beta_{ij}(M)\,\Delta M$$

where:

β: is the coefficient of linear moisture expansion for the material at moisture content M;

$\Delta\varepsilon$: is the change in the strain tensor caused by the small change in moisture at moisture content M;

ΔM: is the change in moisture;

M: is the moisture content.

An **fea_moisture_absorption** is a 3D second order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_moisture_absorption
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor2_3d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA linear moisture expansion tensor.

5.12.10 fea_shell_membrane_stiffness

An **fea_shell_membrane_stiffness** is the collection of mechanical properties representing a linear response of a **fea_material** to a small change in membrane strain. The response is a change in surface membrane force caused by a small change in membrane strain as follows:

$$\Delta N_{ij} = \sum_{k,l=1}^{2} A_{ijkl}\,\Delta\varepsilon_{kl}$$

where:

A: is the shell membrane elasticity tensor;

N: is the 2D second order surface membrane force tensor;

ε: is the 2D second order membrane strain tensor:

$$\varepsilon = \frac{1}{2}(J + J^t)$$

where:

$$J = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{pmatrix}$$

and where:

J: is the Jacobian;

J^t: is the transpose of the Jacobian;

u: is displacement in the x direction;

v: is displacement in the y direction.

An **fea_shell_membrane_stiffness** is a 2D fourth order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_shell_membrane_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor4_2d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA surface membrane elasticity tensor.

5.12.11 fea_shell_bending_stiffness

An **fea_shell_bending_stiffness** is the collection of mechanical properties representing a linear response of an **fea_material** to a small change in curvature. The response is a change in surface aggregate bending moment caused by a small change in curvature as follows:

$$\Delta M_{ij} = \sum_{k,l=1}^{2} A_{ijkl}\, \Delta\varepsilon_{kl}$$

where:

A: is the shell bending elasticity tensor;

M: is the 2D second order aggregate surface bending moment tensor;

ε: is the symmetric 2D second order surface curvature tensor:

$$\begin{pmatrix} \frac{\partial^2 w}{\partial x^2} & -\frac{\partial^2 w}{\partial x \partial y} \\ -\frac{\partial^2 w}{\partial x \partial y} & \frac{\partial^2 w}{\partial y^2} \end{pmatrix}$$

where:

w: is displacement in the z direction.

An **fea_shell_bending_stiffness** is a 2D fourth order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_shell_bending_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor4_2d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA shell bending elasticity tensor.

5.12.12 fea_shell_membrane_bending_coupling_stiffness

An **fea_shell_membrane_bending_coupling_stiffness** is the collection of mechanical properties representing a linear response of an **fea_material** to a small change in curvature. The response is a change in surface membrane force caused by a small change in curvature as follows:

$$\Delta N_{ij} = \sum_{k,l=1}^{2} A_{ijkl}\, \Delta\varepsilon_{kl}$$

where:

A: is the shell membrane bending coupling elasticity tensor;

N: is the 2D second order surface membrane force tensor;

ε: is the symmetric 2D second order surface curvature tensor.

An **fea_shell_membrane_bending_coupling_stiffness** is a 2D fourth order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_shell_membrane_bending_coupling_stiffness
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor4_2d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA shell membrane bending coupling elasticity tensor.

5.12.13 fea_shell_shear_stiffness

An **fea_shell_shear_stiffness** is the collection of mechanical properties representing a linear response of an **fea_material** to a small change in surface out of plane shear strain. The response is a change in surface aggregate out of plane shear forces caused by a small change in surface out of plane shear strain as follows:

$$\Delta F_i = \sum_{j=1}^{2} A_{ij}\, \Delta\varepsilon_j$$

where:

A: is the shell shear elasticity tensor;

F: is the 2D surface aggregate out of plane shear force vector;

ε: is the 2D surface out of plane shear strain vector:

$$\begin{pmatrix} \dfrac{\partial w}{\partial x} + \dfrac{\partial u}{\partial z} \\ \dfrac{\partial w}{\partial y} + \dfrac{\partial v}{\partial z} \end{pmatrix}$$

where:

u: is displacement in the x direction.

v: is displacement in the y direction.

w: is displacement in the z direction.

An **fea_shell_shear_stiffness** is the is a 2D fourth order material response tensor.

NOTE – This property is usually a function of temperature and other independent variables.

EXPRESS specification:

```
*)
ENTITY fea_shell_shear_stiffness
```

```
  SUBTYPE OF (fea_material_property_representation_item);
  fea_constants                    : symmetric_tensor2_2d;
END_ENTITY;
(*
```

Attribute definitions:

fea_constants: the value of the FEA shell shear elasticity tensor.

5.13 Structural response representation schema entity definitions: Element properties

Element properties are information associated with elements that are not derivable from the element connectivity and the associated node geometry.

Volume elements have no associated physical properties other than material properties, so no property entities exist for them. The volume elements refer directly to a material to supply all other needed property information.

5.13.1 Surface element properties

Surface elements are idealised volumes, and as such require other property values, such as thickness, to complete their definition.

A surface element can model a volume of material where the geometry of the volume is specified by a reference surface which is defined by the positions of the nodes, the element shape functions, and a thickness at each point on the surface. The thickness is often defined by nodal values which are then interpolated by the element shape functions.

> NOTE – Other independent definitions of thickness over the surface of an element are possible, such as one described by bicubic polynomials. These independent definitions are not in this part at the present, though a subtyping has been constructed to allow inclusion at a later date with a minimum of impact on the current information model.

Alternatively a surface may be a composite or fabricated shell for which the geometric and section properties may be separated in a simple way. A full description of the material and section shall be contained in a referencing application protocol. In this case an aggregate section thickness is not required within a finite element analysis model, and is not supplied.

The material model may contain constitutive matrices to describe directly the aggregate behaviour of a composite or fabricated surface.

It is also possible to specify constitutive matrices for a non-homogeneous surface within the finite element model. This is done by supplying a separate effective thickness and effective material for each of the membrane, bending, shear, and membrane-coupling matrices. A precise definition for these constitutive matrices and for the different effective thicknesses and effective materials is given in the paragraphs below.

The engineering theory for the thermo-mechanical behaviour of a shell defines the kinematics of deformation within the entire volume of the surface element with respect to variables defined with respect to the reference surface of an element. For example in classical thin shell theory normals remain straight and normal to the reference surface. The principal underlying the finite

element method - the variational principal of minimum potential energy for the displacement formulation elements - is not affected by such kinematic assumptions. However the kinematic assumptions allow strains and stresses to be generalised to give quantities such as curvatures, and bending and twisting moments.

For an analysis of a surface which assumes linear material behaviour it is possible to consider separately the integration through the thickness and the integration over the surface. For example the strain energy for a surface element is given by an integral of the form:

$$V = \frac{1}{2}\,\mathbf{d}^t \int_{surface} \int_{thickness} B^t\, D\, B\, dt\, ds\, \mathbf{d}$$

where:

$\mathbf{d}$ represents the vector of nodal generalised displacements for the element;

B represents the element shape function derivatives which relate the nodal generalised displacements to strains at any point within the element such that:

$$\varepsilon = B\,\mathbf{d}$$

D represents the elasticity matrix for the material which relates the strains to the stresses at a point within the element such that:

$$\sigma = D\,\varepsilon$$

where:

σ is the stress tensor components in a vector form;

ε is the strain tensor components in a vector form.

Alternatively, for an analysis with linear material behaviour the integration through the thickness can be carried out algebraically so that the strain energy is given by an integral of the form:

$$V = \frac{1}{2}\,\mathbf{d}^t \int_{surface} B'^t\, D'\, B'\, ds\, \mathbf{d}$$

where:

B' represents the element shape function derivatives which relate the nodal generalised displacements to strains at any point within the element such that:

$$\varepsilon' = B'\,\mathbf{d}$$

D' represents the constitutive matrix for the surface which relates the generalised strains to generalised stresses at a point on the surface such that:

$$\sigma' = D'\,\varepsilon'$$

where:

σ' is the vector of membrane force and bending moment components:

$$\begin{pmatrix} N_{xx} \\ N_{yy} \\ N_{xy} \\ M_{xx} \\ M_{yy} \\ M_{xy} \end{pmatrix}$$

within which:

N_{xx}, N_{yy}, N_{xy} are components of membrane force;

M_{xx}, M_{yy}, M_{xy} are components of bending moment.

ε' is the vector of membrane strain and curvature components:

$$\begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \\ \chi_{xx} \\ \chi_{yy} \\ \tau_{xy} \end{pmatrix}$$

within which:

ε_{xx}, ε_{yy}, γ_{xy} are the components of membrane strain;

χ_{xx}, χ_{yy}, τ_{xy} are the components of curvature.

At a point on the surface with thickness h and offset p, the limits for section integration are ($p - \frac{h}{2}$ to $p + \frac{h}{2}$). At this point the D' matrix is defined with respect to the coordinate axes for the surface section properties by:

$$D' = \begin{pmatrix} h\,D_m & h\,p\,D_m \\ h\,p\,D_m & (\frac{1}{12}h^3 + h\,p^2)\,D_m \end{pmatrix}$$

where:

D_m is the membrane elasticity matrix, which for a isotropic material has the form:

$$D_m = \frac{E}{(1-\nu^2)} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{pmatrix}$$

where:

E is Young's modulus;

ν is Poisson's ratio.

For some applications it may be desired to use different material and section properties to define the different sub-matrices of D'. When this is desired, the following form of D' is obtained:

$$D' = \begin{pmatrix} h_m\, D_{mm} & h_b\, p\, D_{mc} \\ h_b\, p\, D_{mc} & (\frac{1}{12} h_b^3 + h_b\, p^2)\, D_{mb} \end{pmatrix}$$

where:

D_{mm} is the membrane elasticity matrix;

D_{mb} is the bending elasticity matrix;

D_{mc} is the membrane - bending coupling matrix;

h_m is the membrane thickness;

h_b is the bending thickness.

For some applications it may be desirable to use separate material and section properties for shear behaviour. In this case an aggregate constitutive matrix can be formed to relate shear strain to shear force as follows:

$$\begin{pmatrix} V_x \\ V_y \end{pmatrix} = h_s\, D_s \begin{pmatrix} \varepsilon_{xz} \\ \varepsilon_{yz} \end{pmatrix}$$

where:

V_x **and** V_y are components of shear force;

ε_{xz} **and** ε_{yz} are components of shear strain;

h_s is the effective shear thickness for transverse shear;

D_s is the elasticity matrix for shear.

For an isotropic material the elasticity matrix D_s has the form:

$$D_s = G \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

where:

G is the shear modulus.

NOTE – The relationships of the surface element property subtypes are shown in the EXPRESS-G Partial Model in figure 55.

Figure 55 – Surface element properties EXPRESS-G partial model

5.13.2 Curve element properties

Curve elements, like surface elements, are idealised volumes, and as such require other property values, such as cross-sectional area, to complete their definition.

A curve element has a single parametric axis which is established graphically in 5.8. The position of this axis is defined by the positions of the nodes of an element, the shape functions of an element, and any element end offsets.

The parametric axis runs between the end positions of the element.

NOTE – The end positions can be offset from the end nodes of an element as shown in figure 56.

Figure 56 – Curve element end offsets

The parametric axis cannot be offset from the mid-span nodes of higher order curve elements.

Curve element section properties are defined with respect to the y and z axes of the property coordinate system of an element, whose origin is on the parametric axis of the element.

NOTE – Neither the section centroid or shear center need to lie on the parametric axis of the element as shown in figure 57.

Figure 57 – Curve element property coordinate system origin orientation

A curve 3D element can be divided into intervals which each may have a separate material and property specification. The intervals within a curve element shall be defined in a sequence from node 1 to node 2. For each interval except the last the position of the interval end closest to the end 2 of the element shall be specified. A position is not specified for the last interval to ensure that it ends at the end of the element. A position within a curve element is defined with respect to the parametric coordinate system for the element such that:

- position -1.0 is the end of the element closest to node 1
- position 1.0 is the end of the element closest to node 2

NOTE – An element divided into 2 equal length intervals is shown in figure 58.

Figure 58 – Curve element interval specification conventions

The end fixings of a curve 3D element determine the way in which it is connected to its nodes. A fixing is described by an end offset and an end release, as described in the following two paragraphs.

The end of a curve element can be offset from its node so that a rigid body provides the link between the end of the element and the node. The offset is defined as the distance from the node

to the end position of the element. This distance can be specified with respect to an arbitrary coordinate system or the property coordinate system at the end of the element. The orientation of the property coordinate system may itself depend on the offset.

One or more degrees of freedom can be released between a node and the end of the element. For each released degree of freedom, a connecting stiffness shall be specified (which may be zero). The released degrees of freedom may be specified with respect to an arbitrary coordinate system or the property coordinate system at the end of the element. The strain energy V associated with the release is given by:

$$V = \frac{1}{2}\, k\, d^2$$

where:

k is the release stiffness;

d is the difference between the displacement for the degree of freedom on either side of the release.

If a release and offset are specified at the same end of the element then the release acts between the node and the rigid body linking the end of the element to the node.

NOTE – A release and offset are both shown in figure 59.

NOTE – The relationships of the surface element property subtypes are shown in the EXPRESS-G Partial Model in figure 60.

5.13.3 surface_element_property

A **surface_element_property** is the section properties for a surface element.

NOTE – The material of the surface is specified by subtypes of **fea_material_property_representation_item**. Material property subtypes that apply to surface elements have a rule associating the property to an element. If several material property matrices are needed (such as membrane, bending, membrane-bending coupling, or shear) then each would be an instance of a separate environment in an **fea_material_property_representation_item**

EXPRESS specification:

```
*)
ENTITY surface_element_property;
  property_id                      : identifier;
  description                      : text;
  section                          : surface_section_field;
END_ENTITY;
(*
```

Attribute definitions:

property_id: an application defined identifier of the surface element properties.

description: the surface element properties.

Figure 59 – Curve element end release and end offset

Figure 60 – Curve element properties EXPRESS-G partial model

section: the properties over the field of the surface element.

Informal propositions:

IP1: the combination of an **fea_model** and **property_id** shall be unique within a finite element analysis model.

5.13.4 surface_section_field

A **surface_section_field** is a surface section field that shall be either constant or varying.

EXPRESS specification:

```
*)
ENTITY surface_section_field
  SUPERTYPE OF (ONEOF(surface_section_field_constant,
                      surface_section_field_varying));
END_ENTITY;
(*
```

5.13.5 surface_section_field_constant

A **surface_section_field_constant** is a surface section field that has a constant value over the field of a surface element.

EXPRESS specification:

```
*)
ENTITY surface_section_field_constant
  SUBTYPE OF (surface_section_field);
  definition                      : surface_section;
END_ENTITY;
(*
```

Attribute definitions:

definition: the surface section definition for the surface element. There is only one value which is constant over the entire surface.

5.13.6 surface_section_field_varying

A **surface_section_field_varying** is a surface section field that varies over the field of a surface element. The values in the surface section definition list (which correspond to the required node list) are interpolated using the element shape functions.

EXPRESS specification:

```
*)
ENTITY surface_section_field_varying
  SUBTYPE OF (surface_section_field);
  definition                      : LIST [1:?] OF
                                    surface_section;
  additional_node_values          : BOOLEAN;
END_ENTITY;
```

```
(*
```

Attribute definitions:

definition: the surface section definition for the surface element. The values in the surface section definition list (which corresponds in order to the required node list) are interpolated using the element shape functions.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or just the required nodes (FALSE).

5.13.7 surface_section

A **surface_section** is a surface section definition that shall be uniform, with or without layers.

EXPRESS specification:

```
*)
ENTITY surface_section
  SUPERTYPE OF (ONEOF(uniform_surface_section,
                      uniform_surface_section_layered));
  offset                          : measure_or_unspecified_value;
  non_structural_mass             : measure_or_unspecified_value;
  non_structural_mass_offset      : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

offset: the distance between the neutral plane for bending and the reference surface which is defined by the positions of the nodes and the element shape functions. Surface element properties are defined with respect to the property coordinate system of the element. If the offset is positive, the neutral plane for bending is on the same side of the reference plane as the positive 3 direction of the property coordinate system.

non_structural_mass: the non-structural mass per unit area of the reference plane of the surface element. This mass is in addition to the structural mass which is calculated from the volume of the element and the density of the material of the element.

non_structural_mass_offset: the offset direction which is defined with respect to the reference plane of the surface which in turn is defined by the positions of the nodes and the element shape functions. Surface element properties are defined with respect to the property coordinate system of the element. If the offset is positive, the offset direction is on the same side of the reference plane as the positive 3 direction of the property coordinate system.

5.13.8 uniform_surface_section

A **uniform_surface_section** is a surface section field that is uniform through the thickness of a surface element.

EXPRESS specification:

```
*)
ENTITY uniform_surface_section
  SUBTYPE OF (surface_section);
  thickness                        : length_measure;
  bending_thickness                : measure_or_unspecified_value;
  shear_thickness                  : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

thickness: the membrane thickness of the surface element.

bending_thickness: this attribute specifies the effective thickness for bending of the surface. If it is not supplied, it is assumed to be of the same value as **thickness**.

shear_thickness: the effective thickness for transverse shear (shear normal to the reference plane). This attribute shall be supplied if the surface element permits shear deformation. If it is not supplied for an element that supports shear deformation, then the analysis program shall use the appropriate value for a uniform shell.

5.13.9 uniform_surface_section_layered

A **uniform_surface_section_layered** is a surface section field that is layered through the thickness of a surface element. All layer information shall be specified in a using application protocol. The thickness, and if necessary the bending and shear thickness of the element shall be specified by the sum of the layer thicknesses and the layer properties and orientations.

EXPRESS specification:

```
*)
ENTITY uniform_surface_section_layered
  SUBTYPE OF (surface_section);
END_ENTITY;
(*
```

5.13.10 curve_3d_element_property

A **curve_3d_element_property** is a cross-section for a curve 3D element.

NOTE – The material of the curve is specified by subtypes of **fea_material_property_representation_item**. Material property subtypes that apply to curve elements have a rule associating the property to an element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_property;
  property_id                  : identifier;
  description                  : text;
  interval_definitions         : LIST [1:?] OF curve_element_interval;
  end_offsets                  : ARRAY [1:2] OF
```

```
                                  curve_element_end_offset;
  end_releases                  : ARRAY [1:2] OF
                                  curve_element_end_release;
END_ENTITY;
(*
```

Attribute definitions:

property_id: an application defined identifier for the property, and is unique within the finite element analysis model.

description: the curve element properties.

interval_definitions: the material and section properties for each of the element intervals. The intervals shall be defined in sequence beginning with the interval closest to node 1 and ending with the interval closest to node 2.

end_offsets: the end offsets for the curve element. The item 1 of the array references the definition of the end offset at node 1 of the element. Item 2 references the definition of the end offset at node 2 of the element.

end_releases: the end releases for the curve element. The item 1 of the array references the definition of the end release at node 1 of the element. Item 2 references the definition of the end release at node 2 of the element.

Informal propositions:

IP1: the combination of an **fea_model** and **property_id** shall be unique within a finite element analysis model.

5.13.11 curve_element_interval

A **curve_element_interval** is a curve element interval that shall be either constant or linearly varying.

EXPRESS specification:

```
*)
ENTITY curve_element_interval
  SUPERTYPE OF (ONEOF(curve_element_interval_constant,
                      curve_element_interval_linearly_varying));
  finish_position               : curve_element_location;
  angles                        : euler_angles;
END_ENTITY;
(*
```

Attribute definitions:

material: the material of this interval.

finish_position: the position of the end of the interval closest to node 2 of the curve element. The position is defined by the coordinate with respect to the parametric axis established graphically in 5.8. The intervals of a curve element are listed in the attribute **interval_definitions** of

the curve_3d_element_property. The start position for each interval in the list (except the first) is the finish position of the previous interval. The start position for the first interval in the list has a parametric coordinate -1.0. This is the end at node 1. The finish position for the last interval has a parametric coordinate of 1.0. A finish position of 1.0 shall be supplied for the last interval in the list.

angles: three Euler angles that define a relative coordinate system with respect to the element parametric coordinate system that is used to define the material directions. The **x** material direction shall be aligned with the relative coordinate system **x** axis, similarly the material **y** direction shall be aligned with the relative coordinate system **y** axis, and the material **z** direction shall be aligned with the relative coordinate systems **z** axis.

5.13.12 curve_element_interval_constant

A **curve_element_interval_constant** is a curve section that is constant over an interval of a curve element.

EXPRESS specification:

```
*)
ENTITY curve_element_interval_constant
  SUBTYPE OF (curve_element_interval);
  section                      : curve_element_section_definition;
END_ENTITY;
(*
```

Attribute definitions:

section: the section definition for the interval.

5.13.13 curve_element_interval_linearly_varying

A **curve_element_interval_linearly_varying** is a curve section that varies linearly over an interval of a curve element.

EXPRESS specification:

```
*)
ENTITY curve_element_interval_linearly_varying
  SUBTYPE OF (curve_element_interval);
  sections                     : ARRAY [1:2] OF
                                 curve_element_section_definition;
END_ENTITY;
(*
```

Attribute definitions:

section: the section definitions at each end of the interval. Item 1 of the array references the section at the end of the interval nearest to node 1 of the element. Item 2 of the array references the section at the end of the interval nearest to node 2 of the element.

5.13.14 curve_2d_element_property

A **curve_2d_element_property** is the section properties of a curve 2D element.

NOTE – The material of the curve is specified by subtypes of **fea_material_property_representation_item**. Material property subtypes that apply to curve elements have a rule associating the property to an element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_property;
  property_id                  : identifier;
  description                  : text;
  section                      : curve_element_section_definition;
END_ENTITY;
(*
```

Attribute definitions:

property_id: an application defined identifier for the property, which is unique within the finite element analysis model.

description: the curve element properties.

material: the material for the element.

section: the section definition for the element.

Informal propositions:

IP1: the combination of an **fea_model** and **property_id** shall be unique within a finite element analysis model.

5.13.15 curve_element_section_definition

A **curve_element_section_definition** is the section at a point on the curve element.

EXPRESS specification:

```
*)
ENTITY curve_element_section_definition
  SUPERTYPE OF (curve_element_section_derived_definitions);
  description                  : text;
  section_angle                : plane_angle_measure;
END_ENTITY;
(*
```

Attribute definitions:

description: the curve element definition.

section_angle: the orientation angle of the curve element section about the curve element **x** axis. The angle is positive with respect to the right hand rule.

5.13.16 curve_element_section_derived_definitions

A **curve_element_section_derived_definitions** is cross-sectional information for a curve element.

EXPRESS specification:

```
*)
ENTITY curve_element_section_derived_definitions
  SUBTYPE OF (curve_element_section_definition);
  cross_sectional_area         : context_dependent_measure;
  shear_area                   : ARRAY [1:2] OF measure_or_unspecified_value;
  second_moment_of_area        : ARRAY [1:3] OF context_dependent_measure;
  torsional_constant           : context_dependent_measure;
  warping_constant             : measure_or_unspecified_value;
  location_of_centroid         : ARRAY[1:2] OF measure_or_unspecified_value;
  location_of_shear_centre     : ARRAY[1:2] OF measure_or_unspecified_value;
  location_of_non_structural_mass
                               : ARRAY[1:2] OF measure_or_unspecified_value;
  non_structural_mass          : measure_or_unspecified_value;
  polar_moment                 : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

cross_sectional_area: the cross sectional area of the section.

shear_area: the shear area of the section. Array item 1 specifies the shear area effective for deformation in the **xy** plane of the property coordinate system. Array item 2 specifies the shear area effective for deformation in the **xz** plane of the property coordinate system. An Eulerian beam formulation assumes that there is no shear deformation, i.e. that the shear area is infinite. For such elements no shear area need be specified.

second_moment_of_area: the second moments of area about the section centroid. The components of the inertia tensor with respect to the **y** and **z** axes of the property coordinate system are stored as follows: array item 1 contains I_{yy}, array item 2 contains I_{zz}, array item 3 contains I_{yz}.

torsional_constant: the torsional stiffness of the section. A precise definition of this attribute without any warping contributions is as follows:

$$\frac{\partial \theta}{\partial L} = \frac{t}{J\,G}$$

where:

θ is the angle of twist;

t is the applied torsional moment;

L is the length of the curve;

J is the torsional constant whose units are $(length)^4$;

G is the shear modulus of the material.

warping_constant: the warping coefficient of the section. The torsional behaviour of a curve element is described by the torsional constant and the warping coefficient using the following equation:

$$t = G\,\frac{\partial}{\partial x}(J\,\frac{\partial \theta}{\partial x}) - E\,\frac{\partial^2}{\partial x^2}(C_w\,\frac{\partial \theta}{\partial x})$$

where:

t is the applied torsional moment;

J is the torsional constant;

G is the shear modulus of the material;

C_w is the warping constant whose units are $(length)^6$;

E is the elastic modulus of the material;

θ is the angle of twist;

x is distance along the **x** axis of the curve element.

location_of_centroid: The distance of the section centroid from the parametric axis. Array item 1 specifies the distance in the direction of the property coordinate system **y** axis. Array item 2 specifies the distance in the direction of the property coordinate system mass from the parametric axis. Array item 1 specifies the distance in the direction of the property coordinate system **y** axis. Array item 2 specifies the distance in the direction of the property coordinate system **z** axis.

non_structural_mass: the non structural mass per unit length of the curve element parametric axis.

polar_moment: the polar moment of area of the curve element. This attribute is used to calculate the dynamic inertia of the curve element. It is NOT used to calculate the torsional stiffness. The polar moment is the sum of the perpendicular moments unless there are contributions from non-structural mass.

5.13.17 curve_element_end_offset

A **curve_element_end_offset** is the offset of the end attachment of a curve element.

EXPRESS specification:

*)

```
ENTITY curve_element_end_offset;
  coordinate_system            : curve_element_end_coordinate_system;
  offset_vector                : ARRAY[1:3] OF length_measure;
END_ENTITY;
(*
```

Attribute definitions:

coordinate_system: the coordinate system which is used to specify the end offset.

offset_vector: the distance from the node to the end position of the element with respect to the local orthogonal triad of the specified coordinate system at the node.

5.13.18 curve_element_end_release

A **curve_element_end_release** is the releases of a curve element end.

EXPRESS specification:

```
*)
ENTITY curve_element_end_release;
  coordinate_system            : curve_element_end_coordinate_system;
  releases                     : LIST [1:?] OF
                                 curve_element_end_release_packet;
END_ENTITY;
(*
```

Attribute definitions:

coordinate_system: the coordinate system which is used to specify the end release.

releases: pairs of the degrees of freedom released between the node and the element with respect to the orthogonal triad of the specified coordinate system evaluated at the node, and the associated release stiffness.

5.13.19 curve_element_end_release_packet

A **curve_element_end_release_packet** is the release degree of freedom and stiffness of a curve element end. If a degree of freedom is not released then the joint stiffness is infinite.

EXPRESS specification:

```
*)
ENTITY curve_element_end_release_packet;
  release_freedom              : curve_element_freedom;
  release_stiffness            : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

release_freedom: the degrees of freedom released between the node and the element with respect to the orthogonal triad of the specified coordinate system evaluated at the node. If there are no releases then the enumeration of **none** shall be used.

release_stiffnes: the stiffness associated with the degree of freedom. If no stiffnesses are desired, the value is set to 0.0.

5.13.20 axisymmetric_2d_element_property

An **axisymmetric_2d_element_property** is the properties for all types of axisymmetric 2D elements.

EXPRESS specification:

```
*)
ENTITY axisymmetric_2d_element_property;
  angle                        : plane_angle_measure;
END_ENTITY;
(*
```

Attribute definitions:

angle: the segment considered in an axisymmetric analysis.

5.13.21 plane_2d_element_property

A **plane_2d_element_property** defines the properties for all types of plane 2D elements.

EXPRESS specification:

```
*)
ENTITY plane_2d_element_property
  SUPERTYPE OF (simple_plane_2d_element_property);
  depth                        : length_measure;
END_ENTITY;
(*
```

Attribute definitions:

depth: a depth of a plane stress or plane strain section.

5.13.22 simple_plane_2d_element_property

A **simple_plane_2d_element_property** has no further properties.

NOTE – This entity is supplied for extensibility of the information model.

EXPRESS specification:

```
*)
ENTITY simple_plane_2d_element_property
  SUBTYPE OF (plane_2d_element_property);
END_ENTITY;
(*
```

5.14 Structural response representation schema entity definitions: Groups

A group is a set of elements, nodes or groups within the FEA information model. A group may be used to collect entities that have a common attribute such as colour.

5.14.1 fea_group

An **fea_group** is a group that shall be either be a group of elements or nodes.

EXPRESS specification:

```
*)
ENTITY fea_group
  SUPERTYPE OF (ONEOF(element_group,
                      node_group))
  SUBTYPE OF (group);
END_ENTITY;
(*
```

Informal propositions:

IP1: the combination of an **fea_model** and group id shall be unique within a finite element analysis model.

NOTE – The group id, a unique application defined identifier of an **fea_group** is specified by the **name** attribute of the **group** supertype. Similarly a description is specified by the **description** attribute of the **group** supertype. For example, possible values of the description might be red, set 1, or wing. It is whatever the analyst chooses as a label for group identification.

5.14.2 element_group

An **element_group** is a group containing only elements.

EXPRESS specification:

```
*)
ENTITY element_group
  SUBTYPE OF (fea_group);
  elements                     : SET [1:?] OF element_representation;
END_ENTITY;
(*
```

Attribute definitions:

elements: the elements being grouped.

5.14.3 node_group

A **node_group** is a group containing only nodes.

EXPRESS specification:

```
*)
ENTITY node_group
  SUBTYPE OF (fea_group);
  nodes                        : SET [1:?] OF node_representation;
END_ENTITY;
(*
```

Attribute definitions:

nodes: the nodes being grouped.

5.14.4 fea_group_relation

An **fea_group_relation** is an aggregation of FEA groups.

EXPRESS specification:

```
*)
ENTITY fea_group_relation
  SUBTYPE OF (group_relationship);
WHERE
  WR1: 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.FEA_GROUP' IN
       TYPEOF (SELF\group_relationship.relating_group);
  WR2: 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.FEA_GROUP' IN
       TYPEOF (SELF\group_relationship.related_group);
END_ENTITY;
(*
```

Formal propositions:

WR1: only **fea_group** entities and their subytpes may be a **relating_group**.

WR2: only **fea_group** entities and their subytpes may be a **related_group**.

5.14.5 volume_3d_element_group

An **volume_3d_element_group** is a group containing only volume 3d elements.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_group
  SUBTYPE OF (element_group);
WHERE
  WR1: SIZEOF(query(item <* elements |
       NOT ('STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.'+
            'VOLUME_3D_ELEMENT_REPRESENTATION' IN
            TYPEOF (item))))=0;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be only **volume_3d_element_representation** entities in the group.

5.14.6 volume_2d_element_group

A **volume_2d_element_group** is a group containing only volume 2D elements.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_group
  SUBTYPE OF (element_group);
WHERE
  WR1: SIZEOF(query(item <* elements |
       NOT ('STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.'+
            'VOLUME_2D_ELEMENT_REPRESENTATION' IN
            TYPEOF (item))))=0;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be only **volume_2d_element_representation** entities in the group.

5.14.7 surface_3d_element_group

A **surface_3d_element_group** is a group containing only surface 3D elements.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_group
  SUBTYPE OF (element_group);
WHERE
  WR1: SIZEOF(query(item <* elements |
       NOT ('STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.'+
            'SURFACE_3D_ELEMENT_REPRESENTATION' IN
            TYPEOF (item))))=0;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be only **surface_3d_element_representation** entities in the group.

5.14.8 surface_2d_element_group

A **surface_2d_element_group** is a group containing only surface 2D elements.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_group
  SUBTYPE OF (element_group);
WHERE
  WR1: SIZEOF(query(item <* elements |
       NOT ('STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.'+
            'SURFACE_2D_ELEMENT_REPRESENTATION' IN
```

```
            TYPEOF (item))))=0;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be only **surface_2d_element_representation** entities in the group.

5.14.9 curve_3d_element_group

A **curve_3d_element_group** is a group containing only curve 3D elements.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_group
  SUBTYPE OF (element_group);
WHERE
  WR1: SIZEOF(query(item <* elements |
       NOT ('STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.'+
            'CURVE_3D_ELEMENT_REPRESENTATION' IN
            TYPEOF (item))))=0;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be only **curve_3d_element_representation** entities in the group.

5.14.10 curve_2d_element_group

A **curve_2d_element_group** is a group containing only curve 2D elements.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_group
  SUBTYPE OF (element_group);
WHERE
  WR1: SIZEOF(query(item <* elements |
       NOT ('STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.'+
            'CURVE_2D_ELEMENT_REPRESENTATION' IN
            TYPEOF (item))))=0;
END_ENTITY;
(*
```

Formal propositions:

WR1: there shall be only **curve_2d_element_representation** entities in the group.

5.15 Structural response representation schema function definitions

These definitions are used to ensure the correctness of information in many of the finite element analysis model schema information model entities.

5.15.1 required_0d_nodes

A **required_0d_nodes** checks that the node list for a 0D (point) element contains a single node.

EXPRESS specification:

```
*)
FUNCTION required_0d_nodes
  (required_node_list : LIST [1:?] OF node): BOOLEAN;
  LOCAL
    number_of_nodes            : INTEGER;
    status                     : BOOLEAN;
  END_LOCAL;
  number_of_nodes := SIZEOF (required_node_list);
  IF (number_of_nodes = 1) THEN
    status := TRUE;
  ELSE
    status := FALSE;
  END_IF;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 0D element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.2 required_1d_nodes

A **required_1d_nodes** checks that the node list for a 1D (curve) element contains all the required nodes.

EXPRESS specification:

```
*)
FUNCTION required_1d_nodes
  (required_node_list : LIST [1:?] OF node): BOOLEAN;
  LOCAL
    number_of_nodes            : INTEGER;
    status                     : BOOLEAN;
  END_LOCAL;
  number_of_nodes := SIZEOF (required_node_list);
```

```
  IF (number_of_nodes = 2) THEN
    status := TRUE;
  ELSE
    status := FALSE;
  END_IF;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 1D element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.3 required_2d_nodes

A **required_2d_nodes** checks that the node list for a 2D element (triangle or quadrilateral) contains all the required nodes.

EXPRESS specification:

```
*)
FUNCTION required_2d_nodes
  (required_node_list : LIST [1:?] OF node;
   element_shape : element_2d_shape) : BOOLEAN;
  LOCAL
    required_nodes             : INTEGER;
    number_of_nodes            : INTEGER;
    status                     : BOOLEAN;
  END_LOCAL;
  IF (element_shape = triangle) THEN
    required_nodes := 3;
  END_IF;
  IF (element_shape = quadrilateral) THEN
    required_nodes := 4;
  END_IF;
  number_of_nodes := SIZEOF (required_node_list);
  IF (number_of_nodes = required_nodes) THEN
    status := TRUE;
  ELSE
    status := FALSE;
  END_IF;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 2D element.

shape: input argument defining the geometric shape of the element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.4 required_3d_nodes

A **required_3d_nodes** checks that the node list for a 3D (hexahedron, wedge, tetrahedron, or pyramid) element contains all the required nodes.

EXPRESS specification:

```
*)
FUNCTION required_3d_nodes
  (required_node_list : LIST [1:?] OF node;
   element_shape : volume_3d_element_shape) : BOOLEAN;
  LOCAL
    required_nodes            : INTEGER;
    number_of_nodes           : INTEGER;
    status                    : BOOLEAN;
  END_LOCAL;
  IF (element_shape = hexahedron) THEN
    required_nodes := 8;
  END_IF;
  IF (element_shape = wedge) THEN
    required_nodes := 6;
  END_IF;
  IF (element_shape = tetrahedron) THEN
    required_nodes := 4;
  END_IF;
  IF (element_shape = pyramid) THEN
    required_nodes := 5;
  END_IF;
  number_of_nodes := SIZEOF (required_node_list);
  IF (number_of_nodes = required_nodes) THEN
    status := TRUE;
  ELSE
    status := FALSE;
  END_IF;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 3D element.

element_shape: input argument defining the geometric shape of the element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.5 additional_0d_nodes

An **additional_0d_nodes** checks that no items in the additional_node_list are specified for a 0D (point) element, directionally explicit element, or explicit element.

EXPRESS specification:

```
*)
FUNCTION additional_0d_nodes
  (additional_node_list : LIST [1:?] OF node) : BOOLEAN;
  LOCAL
    status                    : BOOLEAN;
  END_LOCAL;
  status := SIZEOF (additional_node_list) = 0;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 0D element.

5.15.6 additional_1d_nodes

An **additional_1d_nodes** checks that the total number of additional nodes for a 1D (curve) element is consistent with the element order.

EXPRESS specification:

```
*)
FUNCTION additional_1d_nodes
  (additional_node_list : LIST [1:?] OF node;
   order : element_order) : BOOLEAN;
  LOCAL
    maximum_nodes             : INTEGER;
    status                    : BOOLEAN;
  END_LOCAL;
  IF (order = linear) THEN
    maximum_nodes := 0;
  END_IF;
  IF (order = quadratic) THEN
    maximum_nodes := 1;
  END_IF;
  IF (order = cubic) THEN
    maximum_nodes := 2;
  END_IF;
  status := SIZEOF (additional_node_list) = maximum_nodes;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 1D element.

order: input argument defining the geometric order of interpolation of the element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.7 additional_2d_nodes

An **additional_2d_nodes** checks that the total number of additional nodes for a 2D (triangle or quadrilateral) element is consistent with the element shape and order.

EXPRESS specification:

```
*)
FUNCTION additional_2d_nodes
  (additional_node_list : LIST [1:?] OF node_representation;
   element_shape : element_2d_shape;
   order         : element_order) : BOOLEAN;
  LOCAL
    maximum_nodes              : INTEGER;
    status                     : BOOLEAN;
  END_LOCAL;
  IF (element_shape = triangle) THEN
     IF (order = linear) THEN
       maximum_nodes := 0;
     END_IF;
     IF (order = quadratic) THEN
       maximum_nodes := 3;
     END_IF;
     IF (order = cubic) THEN
       maximum_nodes := 7;
     END_IF;
  END_IF;
  IF (element_shape = quadrilateral) THEN
     IF (order = linear) THEN
       maximum_nodes := 0;
     END_IF;
     IF (order = quadratic) THEN
       maximum_nodes := 5;
     END_IF;
     IF (order = cubic) THEN
       maximum_nodes := 12;
     END_IF;
  END_IF;
  status := SIZEOF (additional_node_list) = maximum_nodes;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 2D element.

element_shape: input argument defining the geometric shape of the element.

order: input argument defining the geometric order of interpolation of the element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.8 additional_3d_nodes

An **additional_3d_nodes** checks that the total number of additional nodes for a 3D (hexahedron, wedge, tetrahedron, or pyramid) element is consistent with the element shape and order.

EXPRESS specification:

```
*)
FUNCTION additional_3d_nodes
  (additional_node_list : LIST [1:?] OF node;
   element_shape : volume_3d_element_shape;
   order : element_order) : BOOLEAN;
  LOCAL
    maximum_nodes              : INTEGER;
    status                     : BOOLEAN;
  END_LOCAL;
  IF (element_shape = hexahedron) THEN
     IF (order = linear) THEN
       maximum_nodes := 0;
     END_IF;
     IF (order = quadratic) THEN
       maximum_nodes := 19;
     END_IF;
     IF (order = cubic) THEN
       maximum_nodes := 56;
     END_IF;
  END_IF;
  IF (element_shape = wedge) THEN
     IF (order = linear) THEN
       maximum_nodes := 0;
     END_IF;
     IF (order = quadratic) THEN
       maximum_nodes := 12;
     END_IF;
     IF (order = cubic) THEN
       maximum_nodes := 34;
     END_IF;
  END_IF;
  IF (element_shape = tetrahedron) THEN
     IF (order = linear) THEN
       maximum_nodes := 0;
```

```
    END_IF;
    IF (order = quadratic) THEN
      maximum_nodes := 6;
    END_IF;
    IF (order = cubic) THEN
      maximum_nodes := 16;
    END_IF;
  END_IF;
  IF (element_shape = pyramid) THEN
    IF (order = linear) THEN
      maximum_nodes := 0;
    END_IF;
    IF (order = quadratic) THEN
      maximum_nodes := 9;
    END_IF;
    IF (order = cubic) THEN
      maximum_nodes := 25;
    END_IF;
  END_IF;
  status := SIZEOF (additional_node_list) = maximum_nodes;
RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

required_node_list: input argument consisting of a LIST[1:?] of the required nodes in a 3D element.

element_shape: input argument defining the geometric shape of the element.

order: input argument defining the geometric order of interpolation of the element.

status: boolean output argument that is TRUE if there are the proper number of required nodes in the **required_node_list**.

5.15.9 number_of_terms

A **number_of_terms** is used to derive the number of terms in a matrix.

EXPRESS specification:

```
*)
FUNCTION number_of_terms
  (node_dof_list : LIST [1:?] OF LIST [1:?] OF degree_of_freedom;
   matrix_symmetry: matrix_symmetry) : INTEGER;
  LOCAL
    number_of_terms            : INTEGER;
    number_of_freedoms         : INTEGER;
  END_LOCAL;

  number_of_freedoms := 0; (* loop for each item in the outer list*)
```

```
  REPEAT I := 1 TO SIZEOF (node_dof_list); (* find size of inner list*)
    number_of_freedoms := number_of_freedoms + SIZEOF (node_dof_list[I]);
  END_REPEAT;
  IF ('STRUCTURAL_RESPONSE_DEFINITION_SCHEMA.SYMMETRIC'
       IN TYPEOF(matrix_symmetry)) THEN
    number_of_terms := (number_of_freedoms*(number_of_freedoms+1))/2;
  END_IF;

  IF ('STRUCTURAL_RESPONSE_DEFINITION_SCHEMA.DIAGONAL'
       IN TYPEOF(matrix_symmetry)) THEN
    number_of_terms := number_of_freedoms;
  END_IF;

RETURN (number_of_terms);

END_FUNCTION;
(*
```

Argument definitions:

node_dof_list: input argument consisting of a LIST[1:?] of of LIST[1:?] of the degrees of freedom in a matrix.

matrix_symmetry: input argument defining the type of matrix symmetry.

Number_of_terms: integer output argument that is set to the total number of terms in the matrix.

5.15.10 valid_parametric_coordinate

A **valid_parametric_coordinate** ensures each of the parametric coordinates lies in the range [-1.0, +1.0].

EXPRESS specification:

```
*)

FUNCTION valid_parametric_coordinate
    (coordinates    :  LIST[1:3] OF parameter_value): BOOLEAN;

    LOCAL
     status            : BOOLEAN;
     i                 : INTEGER;
    END_LOCAL;

    status := TRUE;
    REPEAT i:=1 TO HIINDEX(coordinates);
        IF (1.0 < coordinates[i]) OR
           (coordinates[i] < -1.0) THEN
            status := FALSE;
        END_IF;
```

```
    END_REPEAT;
    RETURN (status);
END_FUNCTION;
(*
```

Argument definitions:

coordinates: input argument that is the element parametric coordinates that are being checked.

5.15.11 build_direction_node

A **build_direction_node** calculates the components of a direction that is defined by a vector from **node_1** to **node_2**. The components are expressed in a cartesian coordinate space whose axes are coincident with those of the cartesian, cylindrical, or spherical coordinate space defining the location of the first node in the argument list.

EXPRESS specification:

```
*)
FUNCTION build_direction_node
    (node_1             : node_representation;
     node_2             : node_representation): LIST [2:3] OF REAL;
    LOCAL
     test                   : LOGICAL;
     nodal_direction_ratios : LIST [2:3] OF REAL;
     u                      : direction;
     node_1_coordinates     : cartesian_point;
     node_2_coordinates     : cartesian_point;
    END_LOCAL;

    test := map_to_common_cartesian (node_1, node_2,
            node_1_coordinates, node_2_coordinates);

    u.direction_ratios[1] := node_2_coordinates.coordinates[1] -
                             node_1_coordinates.coordinates[1];
    u.direction_ratios[2] := node_2_coordinates.coordinates[2] -
                             node_1_coordinates.coordinates[2];
    u.direction_ratios[3] := node_2_coordinates.coordinates[3] -
                             node_1_coordinates.coordinates[3];

    u := normalise (u);

    nodal_direction_ratios[1] := u.direction_ratios[1];
    nodal_direction_ratios[2] := u.direction_ratios[2];
    nodal_direction_ratios[3] := u.direction_ratios[3];

    RETURN (nodal_direction_ratios);

END_FUNCTION;
(*
```

Argument definitions:

node_1: input argument that is the first node defining the direction. This node is the origin point of the direction.

node_2: input argument that is the second node defining the direction. This node is the point defining the positive sense of the direction.

5.15.12 map_to_common_cartesian

A **map_to_common_cartesian** maps both nodes in the argument list to a common cartesian coordinate space. The origin of the common space will be established at the location of the first node in a cartesian coordinate space whose axes are coincident with those of the cartesian, cylindrical, or spherical coordinate space defining the location of that node.

EXPRESS specification:

```
*)
FUNCTION map_to_common_cartesian
    (node_1             : node_representation;
     node_2             : node_representation;
     node_1_coordinates : cartesian_point;
     node_2_coordinates : cartesian_point) : LOGICAL;

(*  To be completed by individual implementations of application protocols.
    The function will have to consider the appropriate set of shape
    representations supported by the application protocol being implemented.
    This function will have to follow the founding chains that relate each
    point to the basic coordinate space before any other calculations
    are to be carried out.
*)
  RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

node_1: input argument that is the first node which defines the origin of the common cartesian coordinate space.

node_2: input argument that is the second node that is to be transformed to a cartesian space whose origin is at **node_1**.

node_1_coordinates: the coordinates of the first node in the common cartesian space.

node_2_coordinates: the coordinates of the second node in the common cartesian space.

5.15.13 consistent_geometric_reference

A **consistent_geometric_reference** ensures that the correct type of element aspect (volume, face, edge) is related to the correct type of geometric representation.

EXPRESS specification:

```
*)
FUNCTION consistent_geometric_reference (
  aspect        : GENERIC;
  item          : SET[1:?] OF geometric_representation_item): BOOLEAN;

LOCAL
  status        : BOOLEAN;
  srrs          : STRING;
  i             : INTEGER;
END_LOCAL;

status := FALSE;
srrs := 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.';

REPEAT i := 1 TO SIZEOF (item);
IF (srrs + 'ELEMENT_VOLUME' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRRIC_MODEL_SCHEMA.SOLID_MODEL' IN TYPEOF (item[i])) THEN
    status := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_3D_FACE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.SURFACE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_2D_FACE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.SURFACE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_3D_FACE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.SURFACE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_2D_FACE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.SURFACE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_3D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.CURVE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;
```

```
IF (srrs + 'VOLUME_2D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.CURVE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_3D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.CURVE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_2D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.CURVE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'CURVE_EDGE' IN TYPEOF (aspect)) THEN
  IF ('GEOMETRY_SCHEMA.CURVE' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_3D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_2D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_3D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_2D_EDGE' IN TYPEOF (aspect)) THEN
  IF ('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'CURVE_EDGE' IN TYPEOF (aspect)) THEN
  IF ('TOPOLOGY_SCHEMA.POLY_LOOP' IN TYPEOF (item[i])) THEN
    status  := TRUE;
  END_IF;
```

```
END_IF;

END_REPEAT;

RETURN (status);

END_FUNCTION;
(*
```

5.15.14 consistent_element_reference

A **consistent_element_reference** ensures that the correct type of element aspect (face, edge, volume) is combined with the correct type of **element_representation**.

EXPRESS specification:

```
*)
FUNCTION consistent_element_reference (
  element       : element_representation;
  aspect        : GENERIC): BOOLEAN;

LOCAL
  status        : BOOLEAN;
  srrs          : STRING;
END_LOCAL;

status := FALSE;
srrs := 'STRUCTURAL_RESPONSE_REPRESENTATION_SCHEMA.';

IF (srrs + 'ELEMENT_VOLUME' IN TYPEOF (aspect)) THEN
  IF (srrs + 'ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_3D_FACE' IN TYPEOF (aspect)) THEN
  IF (srrs + 'VOLUME_3D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_2D_FACE' IN TYPEOF (aspect)) THEN
  IF (srrs +'VOLUME_2D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_3D_FACE' IN TYPEOF (aspect)) THEN
  IF (srrs +'SURFACE_3D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;
```

```
IF (srrs + 'SURFACE_2D_FACE' IN TYPEOF (aspect)) THEN
  IF (srrs +'SURFACE_2D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_3D_EDGE' IN TYPEOF (aspect)) THEN
  IF (srrs +'VOLUME_3D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'VOLUME_2D_EDGE' IN TYPEOF (aspect)) THEN
  IF (srrs +'VOLUME_2D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_3D_EDGE' IN TYPEOF (aspect)) THEN
  IF (srrs +'SURFACE_3D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'SURFACE_2D_EDGE' IN TYPEOF (aspect)) THEN
  IF (srrs +'SURFACE_2D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'CURVE_EDGE' IN TYPEOF (aspect)) THEN
  IF (srrs +'CURVE_3D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

IF (srrs + 'CURVE_EDGE' IN TYPEOF (aspect)) THEN
  IF (srrs +'CURVE_2D_ELEMENT_REPRESENTATION' IN TYPEOF (element)) THEN
    status  := TRUE;
  END_IF;
END_IF;

RETURN (status);

END_FUNCTION;
(*
```

EXPRESS specification:

```
*)
END_SCHEMA; -- structural_response_representation_schema
(*
```

6 Finite element analysis control and result schema

The following EXPRESS declaration begins the **finite_element_analysis_control_and_-result_schema** and identifies the necessary external references.

EXPRESS specification:

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

  REFERENCE FROM measure_schema
      (context_dependent_measure,
       length_measure,
       plane_angle_measure);

  REFERENCE FROM geometry_schema
      (point,
       curve,
       surface);

  REFERENCE FROM geometric_model_schema
      (solid_model);

  REFERENCE FROM structural_response_representation_schema
      (node_representation,
       element_representation,
       degree_of_freedom,
       axisymmetric_volume_2d_element_representation,
       plane_volume_2d_element_representation,
       axisymmetric_surface_2d_element_representation,
       plane_surface_2d_element_representation,
       axisymmetric_curve_2d_element_representation,
       plane_curve_2d_element_representation,
       point_element_representation,
       directionally_explicit_element_representation,
       explicit_element_representation,
       volume_3d_element_representation,
       surface_3d_element_representation,
       curve_3d_element_representation,
       volume_2d_element_representation,
       surface_2d_element_representation,
       curve_2d_element_representation,
       fea_axis2_placement_3d,
       volume_3d_element_coordinate_system,
       volume_2d_element_coordinate_system,
       surface_3d_element_coordinate_system,
       surface_2d_element_coordinate_system,
       curve_3d_element_coordinate_system,
```

```
      curve_2d_element_coordinate_system,
      volume_element_location,
      surface_volume_element_location,
      curve_volume_element_location,
      surface_section_element_location,
      surface_element_location,
      curve_element_location,
      curve_section_element_location,
      fea_model,
      node_group,
      element_group,
      volume_3d_element_group,
      volume_2d_element_group,
      surface_3d_element_group,
      surface_2d_element_group,
      curve_3d_element_group,
      curve_2d_element_group,
      element_descriptor,
      volume_3d_element_descriptor,
      volume_2d_element_descriptor,
      axisymmetric_volume_2d_element_descriptor,
      plane_volume_2d_element_descriptor,
      surface_3d_element_descriptor,
      surface_2d_element_descriptor,
      axisymmetric_surface_2d_element_descriptor,
      plane_surface_2d_element_descriptor,
      curve_3d_element_descriptor,
      curve_2d_element_descriptor,
      axisymmetric_curve_2d_element_descriptor,
      plane_curve_2d_element_descriptor);

  REFERENCE FROM fea_scalar_vector_tensor_schema
     (scalar,
      tensor1,
      tensor1_2d,
      tensor1_3d,
      symmetric_tensor2_2d,
      anisotropic_symmetric_tensor2_2d,
      symmetric_tensor2_3d,
      isotropic_symmetric_tensor2_3d,
      orthotropic_symmetric_tensor2_3d,
      anisotropic_symmetric_tensor2_3d,
      symmetric_tensor4_2d,
      anisotropic_symmetric_tensor4_2d);

(*
```

NOTE – The schemas referenced above can be found in the following Parts of ISO 10303:

support_resource_schema	ISO 10303-41
measure_schema	ISO 10303-41
geometry_schema	ISO 10303-42
structural_response_representation_schema	Clause 5 of this Part of ISO 10303

fea_scalar_vector_tensor_schema Clause 7 of this Part of ISO 10303

6.1 Introduction

The subjects of the **finite_element_analysis_control_and_result_schema** are the controls necessary to perform a finite element analysis, and the analysis output generated from the analysis of a combination of finite element model and control.

6.2 Fundamental concepts and assumptions

The fundamental concepts and assumptions include the concepts of a control, a step, a state, a process, a result, the roles of a **state_definition** in defining and containing analysis output information, and units and measures.

6.2.1 Control

A control for a finite element analysis describes the operations carried out upon the model. A model may have one or more sets of control information. A control specifies the operations to be carried out upon a model as a set of analysis steps.

The control information also includes the administrative and configuration control information, and the constraints upon the model which act for each analysis step.

6.2.2 Process

A process defines the way in which a model is caused to depart from its initial state. For linear analysis (no other type is within scope) a process specifies a final model state.

The objective for the finite element analysis application is to calculate the response of the model to the change from the initial state to the final state of the following model state information:

- external loads, such as forces and moments in a stress/displacement analysis;
- independent field variables which effect the material of an element, e.g., by causing initial stresses or strains;
- the values of constraints.

6.2.3 Result

A result for a finite element analysis consists of the response of a model to a control as calculated by a finite element analysis application.

A result has the following structure:

- A control has one or more sets of result information. Each set of result information corresponds to a separate execution of an finite element analysis application.

- A result has one or more analysis sub-steps. The finite element analysis application divides the analysis steps specified by the control into sub-steps in order to carry out the calculation. For a linear analysis (no other type is in scope), each analysis step has a single sub-step.

- For each analysis sub-step there is a single state, which describes the model at the end of the sub-step.

6.2.4 State

A state describes the value of an analysis variable of a model at an instant.

The state information includes:

- the nodal variable values (such as deflections);

- the field variable values within the elements;

- the values of constraints.

The state information is defined in **state_definition**s.

6.2.5 Step

An analysis step may either be a **control_analysis_step** or a **result_analysis_step**. A control analysis step defines an operation on the model by specifying an initial state and a process. Nodal constraints are associated with a **control_analysis_step**, as are output requests. A **result_analysis_step** may only reference a **calculated_state**.

6.2.6 State definition

The type of information to be output is specified by **state_definition_without_value** entities.

The control state information may be either specified, derived, or a linearly superimposed combination. Control state or analysis output state information are contained in **state_definition_-with_value** entities.

A **state_definition** therefore either:

- defines an output request set by specifying the variable for which values are to be calculated;

- defines a state of a node, element, or a portion of the model by specifying the values of a variable;

- defines an analysis output message.

The variable may be:

- a variable expressed with respect to solution degrees of freedom at a node of the model;

- a field variable at a point within an element of the model, or aggregated within one or more elements;

– analysis status messages (errors, warnings, or notes).

6.2.7 Units and measures

Units are assigned in a context, and the value of the measure is assigned directly.

NOTE – An EXPRESS-G Partial Model illustrating the control and result relationships is shown in figure 61

Figure 61 – Finite element analysis control EXPRESS-G partial model

6.3 Finite element analysis control and result schema type definitions

The following are a resource for the entities in the finite element analysis control and result schema of the FEA information model.

6.3.1 model_or_control_element

A **model_or_control_element** enables an attribute to reference either a control (constraint element) or a model element.

beginecode

```
*)
TYPE model_or_control_element = SELECT
     (element_representation,
      constraint_element);
END_TYPE;
(*
```

6.3.2 cylindrical_harmonic_number

A **cylindrical_harmonic_number** is a cylindrical harmonic number for a **cylindrical_harmonic_control** and a **state** of an axisymmetric analysis.

beginecode

```
*)
TYPE cylindrical_harmonic_number = INTEGER;
WHERE
  WR1: SELF >= 0;
END_TYPE;
(*
```

Formal propositions:

WR1: a harmonic number shall be positive or zero.

6.3.3 volume_3d_face

A **volume_3d_face** is an element face of a volume 3D element.

The diagrams 27, 31, 35, and 39 in clause 5.8 establish graphically the relationship between the element face number and the positions in the volume 3D element **required_node_list** for the vertex nodes. The element faces are in the same positions with respect to this list for each element order.

beginecode

```
*)
```

```
TYPE volume_3d_face = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 6);
END_TYPE;
(*
```

Formal propositions:

WR1: a volume 3D element face shall be in the range [1, 6].

6.3.4 volume_2d_face

A **volume_2d_face** is an element face of a volume 2D element.

The diagrams 18 and 23 in clause 5.8 establish graphically the relationship between the element face number and the positions in the volume 2D element **required_node_list** for the vertex nodes. The element faces are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE volume_2d_face = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 4);
END_TYPE;
(*
```

Formal propositions:

WR1: a 2D volume element face shall be in the range [1, 4].

6.3.5 volume_3d_edge

A **volume_3d_edge** is an element edge of a volume 3D element.

The diagrams 27, 31, 35, and 39 in clause 5.8 establish graphically the relationship between the element edge number and the element edge positions in the volume 3D element. The element edges are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE volume_3d_edge = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 12);
END_TYPE;
(*
```

Formal propositions:

WR1: a volume 3D edge shall be in the range [1,12].

6.3.6 volume_2d_edge

A **volume_2d_edge** is an element edge of a volume 2D element.

The diagrams 18 and 23 in clause 5.8 establish graphically the relationship between the element edge number and the element edge positions in the volume 2D element. The element edges are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE volume_2d_edge = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 4);
END_TYPE;
(*
```

Formal propositions:

WR1: a volume 2D edge shall be in the range [1,4].

6.3.7 surface_3d_face

A **surface_3d_face** is a top or bottom of a surface 3D element.

The diagrams 17 and 22 in clause 5.8 establish graphically the relationship between between the element face number, and the top and bottom of a surface 3D element. The element faces are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE surface_3d_face = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 2);
END_TYPE;
(*
```

Formal propositions:

WR1: a surface 3D element face shall be in the range [1, 2]. An element face of 1 shall be the positive **z** parametric axis side of the element, the 2 element face shall be the negative **z** parametric axis side of the element.

6.3.8 surface_3d_edge

A **surface_3d_edge** is an element edge of a surface 3D element.

The diagrams 17 and 22 in clause 5.8 establish graphically the relationship between between the element edge number and the element edge positions in the surface 3D element. The element edges are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE surface_3d_edge = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 4);
END_TYPE;
(*
```

Formal propositions:

WR1: a surface 3D element edge shall be in the range [1, 4].

6.3.9 surface_2d_face

A **surface_2d_face** is the top or bottom of a surface 2D element.

The diagram 13 in clause 5.8 establishes graphically the relationship between between the face number, and the top and bottom of a surface 2D element. The faces are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE surface_2d_face = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 2);
END_TYPE;
(*
```

Formal propositions:

WR1: a surface 2D face shall be in the range [1, 2]. An element face of 1 shall be the positive **z** parametric axis side of the element, an element face of 2 shall be the negative **z** parametric axis side of the element.

6.3.10 surface_2d_edge

A **surface_2d_edge** is an element edge of a surface 2D element.

The diagram 13 in clause 5.8 establishes graphically the relationship between the element edge number and the edge positions in the surface 2D element. The element edges are in the same positions with respect to this list for each element order.

beginecode

```
*)
TYPE surface_2d_edge = INTEGER;
WHERE
  WR1: (SELF >= 1) AND (SELF <= 2);
END_TYPE;
(*
```

Formal propositions:

WR1: a surface 2D element edge shall be in the range [1, 2].

6.3.11 field_value

A **field_value** is the value of the field variable.

beginecode

```
*)
TYPE field_value = SELECT
  (unspecified_value,
   scalar,
   plane_angle_measure,
   tensor1_2d,
   tensor1_3d,
   symmetric_tensor2_2d,
   symmetric_tensor2_3d);
END_TYPE;
(*
```

6.3.12 unspecified_value

An **unspecified_value** is a value that is not specified.

beginecode

```
*)
TYPE unspecified_value = ENUMERATION OF
  (unspecified);
END_TYPE;
(*
```

6.3.13 measure_or_unspecified_value

A **measure_or_unspecified_value** is a value that is either a measure or not specified.

beginecode

```
*)
TYPE measure_or_unspecified_value = SELECT
  (context_dependent_measure,
   unspecified_value);
END_TYPE;
(*
```

6.3.14 boundary_variable

A **boundary_variable** is a field variable on a boundary of an element.

beginecode

```
*)
TYPE boundary_variable = SELECT
   (boundary_surface_scalar_variable,
   boundary_surface_vector_3d_variable,
   user_defined_scalar_variable,
   user_defined_vector_3d_variable);
END_TYPE;
(*
```

6.3.15 boundary_aggregated_variable

A **boundary_aggregated_variable** is an aggregated field variable on a boundary of an element.

beginecode

```
*)
TYPE boundary_aggregated_variable = SELECT
   (aggregated_vector_3d_variable,
   user_defined_vector_3d_variable);
END_TYPE;
(*
```

6.3.16 volume_variable

A **volume_variable** is a field variable within the volume of an element.

beginecode

```
*)
TYPE volume_variable = SELECT
  (volume_scalar_variable,
   volume_angular_variable,
   volume_vector_3d_variable,
   volume_tensor2_3d_variable,
   user_defined_scalar_variable,
   user_defined_vector_3d_variable,
   user_defined_tensor2_3d_variable);
END_TYPE;
(*
```

6.3.17 volume_aggregated_variable

A **volume_aggregated_variable** is an aggregated field variable within an element.

beginecode

```
*)
```

```
TYPE volume_aggregated_variable = SELECT
  (aggregated_scalar_variable,
   aggregated_angular_variable,
   aggregated_vector_3d_variable,
   aggregated_tensor2_3d_variable,
   user_defined_scalar_variable,
   user_defined_vector_3d_variable,
   user_defined_tensor2_3d_variable);
END_TYPE;
(*
```

6.3.18 surface_element_variable

A **surface_element_variable** is a field variable at a point on the surface of an element.

NOTE – Many user defined variables are selected in the **volume_variable**.

beginecode

```
*)
TYPE surface_element_variable = SELECT
  (volume_variable,
   surface_scalar_variable,
   surface_vector_2d_variable,
   surface_vector_3d_variable,
   surface_tensor2_2d_variable,
   user_defined_tensor2_2d_variable);
END_TYPE;
(*
```

6.3.19 boundary_edge_variable

A **boundary_edge_variable** is a field variable on an element edge of a surface.

beginecode

```
*)
TYPE boundary_edge_variable = SELECT
   (boundary_curve_scalar_variable,
    boundary_curve_vector_3d_variable,
    user_defined_scalar_variable,
    user_defined_vector_3d_variable);
END_TYPE;
(*
```

6.3.20 curve_element_variable

A **curve_element_variable** is a field variable at a point on a curve element.

NOTE – Many user defined variables are selected in the **volume_variable**.

beginecode

```
*)
TYPE curve_element_variable = SELECT
  (volume_variable,
   curve_scalar_variable,
   curve_vector_2d_variable,
   user_defined_vector_2d_variable,
   curve_vector_3d_variable);
END_TYPE;
(*
```

6.3.21 curve_scalar_variable

A **curve_scalar_variable** is a scalar field variable that is aggregated over the section of a curve element at a point along its length.

beginecode

```
*)
TYPE curve_scalar_variable = ENUMERATION OF
     (curve_axial_force,
      curve_axial_strain,
      torque,
      curve_warping,
      bi_moment,
      twist);
END_TYPE;
(*
```

Enumerated item definitions:

curve_axial_force: the total direct force acting across a plane normal to the axis of a curve element. A positive value denotes tension.

curve_axial_strain: the elastic strain along the axis of a curve element. This is given by:

$$\varepsilon = \frac{\partial u}{\partial x}$$

where:

u is translation of the element axis in the x direction of the curve element axis system;

x is the axis of the curve element axis system tangential to the element.

torque: the total moment of shear force about the axis of a curve element acting across a plane normal to the axis. A positive value results from a deformation such that the rotation of the section about a direction tangential to the axis increases with distance in that direction.

curve_warping: the warping of the section of a curve element.

bi_moment: the conjugate of warping of the section of a curve element.

twist: the torsional (twisting) of a curve element which is given by:

$$\phi = \frac{\partial \theta}{\partial x}$$

where:

θ is rotation of the element section about the x direction of the curve element axis system;

x is the axis of the curve element axis system tangential to the element.

6.3.22 surface_scalar_variable

A **surface_scalar_variable** is a scalar field variable that is aggregated through the section of a surface element at a point on its surface.

beginecode

```
*)
TYPE surface_scalar_variable = ENUMERATION OF
     (thickness,
      surface_thermal_gradient,
      reference_surface_thermal_gradient);
END_TYPE;
(*
```

Enumerated item definitions:

thickness: the thickness at a point on the surface of a surface element.

surface_thermal_gradient: a scalar quantity that is defined over a surface element. The surface thermal gradient is defined as follows:

$$\frac{\partial \theta}{\partial z}$$

where:

θ**:** is the temperature at a point within the thickness of the surface.

z**:** is distance through the thickness in the surface element z axis direction.

reference_surface_thermal_gradient: the reference temperature for the surface thermal gradient.

6.3.23 volume_scalar_variable

A **volume_scalar_variable** is a scalar field variable that is evaluated at a point within the volume of an element.

beginecode

```
*)
TYPE volume_scalar_variable = ENUMERATION OF
     (temperature,
      moisture,
      reference_temperature,
      strain_energy_per_unit_volume);
END_TYPE;
(*
```

Enumerated item definitions:

temperature: the temperature at a point.

moisture: the moisture at a point.

reference_temperature: the reference temperature at a point.

strain_energy_per_unit_volume: the strain energy density evaluated at a point.

6.3.24 boundary_curve_scalar_variable

A **boundary_curve_scalar_variable** is a scalar field variable that is evaluated at a point along the element edge of a surface 3D element.

beginecode

```
*)
TYPE boundary_curve_scalar_variable = ENUMERATION OF
     (normal_force_per_unit_length);
END_TYPE;
(*
```

Enumerated item definitions:

normal_force_per_unit_length: a scalar that is defined over a boundary element edge. The normal force per unit length acts into the element.

Normal_force_per_unit_length is a special case of
applied_force_per_unit_length.

6.3.25 boundary_surface_scalar_variable

A **boundary_surface_scalar_variable** is a scalar field variable that is evaluated at a point on the element face a surface or volume element.

beginecode

```
*)
TYPE boundary_surface_scalar_variable = ENUMERATION OF
     (pressure);
END_TYPE;
(*
```

Enumerated item definitions:

pressure: a scalar that is defined over a boundary surface or element edge. The pressure is the force per unit area acting normal to the surface into the element.

Pressure is a special case of **applied_force_per_unit_area**.

6.3.26 aggregated_scalar_variable

An **aggregated_scalar_variable** is a scalar variable that is obtained by aggregation of a field within an element or over an element boundary. The variable may be evaluated for an element face, a single element, an element group or an entire model.

beginecode

```
*)
TYPE aggregated_scalar_variable = ENUMERATION OF
     (total_strain_energy,
      mass,
      volume);
END_TYPE;
(*
```

Enumerated item definitions:

total_strain_energy: the total strain energy within a volume of the model.

mass: the mass of a single element, an element group, or an entire model.

volume: the volume of a single element, an element group, or an entire model.

6.3.27 volume_angular_variable

A **volume_angular_variable** is a scalar variable that is applied to a volume about the origin of the founding placement of the using entity, and about the Z axis of that placement.

beginecode

```
*)
TYPE volume_angular_variable = ENUMERATION OF
     (constant_angular_acceleration,
      user_defined_angular_scalar_variable);
END_TYPE;
```

```
(*
```

Enumerated item definitions:

constant_angular_acceleration: a constant angular acceleration applied to a volume.

user_defined_angular_scalar_variable: a scalar variable that is arbitrarily defined.

6.3.28 aggregated_angular_variable

An **aggregated_angular_variable** is an aggregated scalar variable that is applied to a volume about the origin of the founding placement of the using entity, and about the Z axis of that placement.

beginecode

```
*)
TYPE aggregated_angular_variable = ENUMERATION OF
      (total_applied_moment,
       user_defined_aggregated_angular_scalar_variable);
END_TYPE;
(*
```

Enumerated item definitions:

total_applied_moment: the total applied force applied to a volume.

user_defined_aggregated_angular_scalar_variable: A scalar variable that is arbitrarily defined.

6.3.29 user_defined_scalar_variable

A **user_defined_scalar_variable** is a scalar variable that is arbitrarily defined. The string shall be suitable for a tabular column heading.

beginecode

```
*)
TYPE user_defined_scalar_variable = STRING;
END_TYPE;
(*
```

6.3.30 curve_vector_2d_variable

A **curve_vector_2d_variable** is a 2D vector field variable that is aggregated over the section of a curve element at a point along its length.

beginecode

```
*)
TYPE curve_vector_2d_variable = ENUMERATION OF
```

```
     (curve_shear_force,
      curve_bending_moment,
      curve_element_curvature,
      curve_thermal_gradient,
      reference_curve_thermal_gradient);
END_TYPE;
(*
```

Enumerated item definitions:

curve_shear_force: the aggregate shear force at a point on a curve element. The sign convention is defined by the following equation:

$$f = \mathbf{F}^t \begin{pmatrix} n_y \\ n_z \end{pmatrix}$$

where:

F is the aggregate shear force vector at a point on the curve with respect to the specified curve coordinate system;

f is the force applied by the curve on the side of the point with the higher x coordinate of the specified system to the side with the lower x coordinate, in the direction of unit vector **n** normal to the curve;

n_y **and** n_z denote the components of unit vector **n** in the directions of the axes y and z of the specified curve coordinate system.

The first component of **F** is the force in the direction of the curve coordinate system **y** axis, and the second component of **F** is the force in the direction of the curve coordinate system **z** axis.

curve_bending_moment: the aggregate bending moment at a point on a curve element. The sign convention is defined by the following equation:

$$m = \mathbf{M}^t \begin{pmatrix} n_y \\ n_z \end{pmatrix}$$

where:

M is the aggregate bending moment at a point on the curve with respect to the specified curve coordinate system;

m is the moment applied by the curve on the side of the point with the higher x coordinate of the specified system to the side with the lower x coordinate, evaluated about the unit vector **n** normal to the curve;

n_y **and** n_z denote the components of unit vector **n** in the directions of the axes y and z of the specified curve coordinate system.

The first component of **M** is the moment about the curve coordinate system **y** axis, and the second component of **M** is the moment about the curve coordinate system **z** axis.

curve_element_curvature: this is the curvature vector for a curve which is defined as:

$$\mathbf{c} = \begin{pmatrix} \frac{\partial^2 v}{\partial x^2} \\ \frac{\partial^2 w}{\partial x^2} \end{pmatrix}$$

where:

x is distance along the curve in the direction of the **x** axis;

v is displacement in the direction of the **y** axis of the specified curve coordinate system;

w is displacement in the direction of the **z** axis of the specified curve coordinate system.

This definition of curvature is only valid for small deformations to a curve with a small initial curvature.

curve_thermal_gradient: a 2D vector that is defined over a curve element. The curve thermal gradient is defined as follows:

$$\begin{pmatrix} \frac{\partial \theta}{\partial y} \\ \frac{\partial \theta}{\partial z} \end{pmatrix}$$

where:

θ**:** is the temperature at a point within the section of the curve.

y**:** is distance through the section in the curve element y axis direction.

z**:** is distance through the section in the curve element z axis direction.

reference_curve_thermal_gradient: the reference temperature for the curve thermal gradient.

6.3.31 surface_vector_2d_variable

A **surface_vector_2d_variable** is a 2D vector field variable that is aggregated through the section of a surface element at a point on its surface.

beginecode

```
*)
TYPE surface_vector_2d_variable = ENUMERATION OF
     (surface_out_of_plane_shear_force,
```

```
      surface_out_of_plane_shear_strain);
END_TYPE;
(*
```

Enumerated item definitions:

surface_out_of_plane_shear_force: the aggregate out-of-plane shear force at a point on a curve element. The sign convention is defined by the following equation:

$$f = \mathbf{F}^t \begin{pmatrix} n_x \\ n_y \end{pmatrix}$$

where:

$\mathbf{F}$ is the aggregate out-of-plane shear force at a point on the surface with respect to the specified surface coordinate system;

f is the force in the direction of the **z** axis of the specified coordinate system, applied across a unit width of surface normal to the unit vector **n** in the plane of the surface. The force is applied by the surface on the side of the unit width in the direction of the normal **n** to the side away from the normal;

n_x and

n_y denote the components of unit vector **n** in the directions of the axes x and y of the specified surface coordinate system.

surface_out_of_plane_shear_strain: this is the out of plane shear strain vector for a surface, which is defined as:

$$\varepsilon = \begin{pmatrix} \dfrac{\partial w}{\partial x} + \dfrac{\partial u}{\partial z} \\ \dfrac{\partial w}{\partial y} + \dfrac{\partial v}{\partial z} \end{pmatrix}$$

where:

x is distance over the surface in the direction of the **x** axis of the specified surface coordinate system;

y is distance over the surface in the direction of the **y** axis of the specified surface coordinate system;

u is displacement in the direction of the **x** axis of the specified surface coordinate system.

v is displacement in the direction of the **y** axis of the specified surface coordinate system.

w is displacement in the direction of the **z** axis of the specified surface coordinate system.

6.3.32 user_defined_vector_2d_variable

A **user_defined_vector_2d_variable** is a 2D vector variable that is arbitrarily defined. The string shall be suitable for a tabular column heading.

beginecode

```
*)
TYPE user_defined_vector_2d_variable = STRING;
END_TYPE;
(*
```

6.3.33 curve_vector_3d_variable

A **curve_vector_3d_variable** is a 3D vector field variable that is aggregated over the section of a curve element at a point along its length.

beginecode

```
*)
TYPE curve_vector_3d_variable = ENUMERATION OF
     (applied_force_per_unit_length,
      applied_moment_per_unit_length);
END_TYPE;
(*
```

Enumerated item definitions:

applied_force_per_unit_length: the force per unit length of an element applied to the model.

applied_moment_per_unit_length: the moment per unit length of an element applied to the model.

6.3.34 surface_vector_3d_variable

A **surface_vector_3d_variable** is a 3D vector field variable that is aggregated through the section of a surface element at a point on its surface.

beginecode

```
*)
TYPE surface_vector_3d_variable = ENUMERATION OF
     (applied_force_per_unit_area,
      applied_moment_per_unit_area);
END_TYPE;
(*
```

Enumerated item definitions:

applied_force_per_unit_area: the force per unit area of an element applied to the model.

applied_moment_per_unit_area: the moment per unit area of an element applied to the model.

6.3.35 volume_vector_3d_variable

A **volume_vector_3d_variable** is a 3D vector field variable at a point within the volume of an element.

beginecode

```
*)
TYPE volume_vector_3d_variable = ENUMERATION OF
     (position,
      applied_force_per_unit_volume,
      applied_moment_per_unit_volume,
      displacement,
      infinitesimal_rotation,
      acceleration);
END_TYPE;
(*
```

Enumerated item definitions:

position: the position of a point within the volume of an element.

applied_force_per_unit_volume: the force per unit volume of an element applied to the model.

applied_moment_per_unit_volume: the moment per unit volume of an element applied to the model.

displacement: the displacement of a point within an element.

infinitesimal_rotation: the infinitesimal rotation of fibres embedded within the material of an element at a point. These rotations are assumed to be small for a linear analysis, so that they can be regarded as a vector quantity.

acceleration: a 3D vector that is defined over a volume of an element.

6.3.36 boundary_curve_vector_3d_variable

A **boundary_curve_vector_3d_variable** is a 3D vector field variable that is evaluated at a point along the element edge of a 3D surface element.

beginecode

```
*)
TYPE boundary_curve_vector_3d_variable = ENUMERATION OF
     (applied_force_per_unit_length,
```

```
     applied_moment_per_unit_length);
END_TYPE;
(*
```

Enumerated item definitions:

applied_force_per_unit_length: the force per unit length of element edge applied to the model.

applied_moment_per_unit_length: the moment per unit length of element edge applied to the model.

6.3.37 boundary_surface_vector_3d_variable

A **boundary_surface_vector_3d_variable** is a 3D vector field variable that is evaluated at a point on the element face of a volume element.

beginecode

```
*)
TYPE boundary_surface_vector_3d_variable = ENUMERATION OF
     (applied_force_per_unit_area,
      applied_moment_per_unit_area);
END_TYPE;
(*
```

Enumerated item definitions:

applied_force_per_unit_area: the force per unit area of element face applied to the model.

applied_moment_per_unit_length: the moment per unit area of element face applied to the model.

6.3.38 aggregated_vector_3d_variable

An **aggregated_vector_3d_variable** is a 3D vector variable that is obtained by aggregation of a field within an element or over an element boundary. The variable may be evaluated for an element face, a single element, an element group or an entire model.

beginecode

```
*)
TYPE aggregated_vector_3d_variable = ENUMERATION OF
     (total_applied_force,
      centre_of_mass_offset);
END_TYPE;
(*
```

Enumerated item definitions:

total_applied_force: the total applied force applied to a volume of the model.

centre_of_mass_offset: a 3D vector that is obtained by integration over an element, element group or whole model. The offset is the distance of the centre of mass from the origin of the master coordinate system for the model.

6.3.39 user_defined_vector_3d_variable

A **user_defined_vector_3d_variable** is a 3D vector variable that is arbitrarily defined. The string shall be suitable for a tabular column heading.

beginecode

```
*)
TYPE user_defined_vector_3d_variable = STRING;
END_TYPE;
(*
```

6.3.40 surface_tensor2_2d_variable

A **surface_tensor2_2d_variable** is a 2D second order tensor field variable, that is aggregated through the section of a surface element at a point on its surface.

beginecode

```
*)
TYPE surface_tensor2_2d_variable = ENUMERATION OF
     (surface_membrane_force,
      surface_membrane_strain,
      surface_bending_moment,
      surface_curvature);
END_TYPE;
(*
```

Enumerated item definitions:

surface_membrane_force: the aggregate membrane force at a point on a surface element. The sign convention is defined by the following equation:

$$\mathbf{f} = \mathcal{F} \begin{pmatrix} n_x \\ n_y \end{pmatrix}$$

where:

$\mathcal{F}$ is the aggregate membrane force tensor at a point on the surface with respect to the specified surface coordinate system;

$\mathbf{f}$ is the force in the plane of the surface with respect to the x and y axes of the specified coordinate system, applied across a unit width of surface normal to the unit vector $\mathbf{n}$ in the plane of the surface. The force is applied by the surface on the side of the unit width in the direction of the normal $\mathbf{n}$ to the side away from the normal;

n_x **and** n_y denote the components of unit vector **n** in the directions of the axes x and y of the specified surface coordinate system.

surface_membrane_strain: the aggregate membrane strain for the surface, which is defined as:

$$\varepsilon = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}$$

where:

u is displacement in the direction of the **x** axis of the specified surface coordinate system;

v is displacement in the direction of the **y** axis of the specified surface coordinate system.

This tensor is symmetrical and only three components are stored. The engineering convention, in which the off-diagonal components are summed, is not used.

This definition of strain is only valid for small deformations to a surface with a small initial strain.

surface_bending_moment: the aggregate bending moment at a point on a surface element. The sign convention is defined by the following equation:

$$\mathbf{m} = \mathcal{M} \begin{pmatrix} n_x \\ n_y \end{pmatrix}$$

where:

$\mathcal{M}$ is the aggregate membrane force tensor at a point on the surface with respect to the specified surface coordinate system;

m is the moment about the x and y axes of the specified coordinate system, applied across a unit width of surface normal to the unit vector **n** in the plane of the surface. The moment is applied by the surface on the side of the unit width in the direction of the normal **n** to the side away from the normal;

n_x **and** n_y denote the components of unit vector **n** in the directions of the axes x and y of the specified surface coordinate system.

surface_curvature: the curvature tensor for a surface which is defined as:

$$\varepsilon = \begin{pmatrix} \frac{\partial^2 w}{\partial x^2} & -\frac{\partial^2 w}{\partial x \partial y} \\ -\frac{\partial^2 w}{\partial x \partial y} & \frac{\partial^2 w}{\partial y^2} \end{pmatrix}$$

where:

x is distance over the surface in the direction of the **x** axis of the specified surface coordinate system;

y is distance over the surface in the direction of the **y** axis of the specified surface coordinate system;

w is displacement in the direction of the **z** axis of the specified surface coordinate system.

This tensor is symmetrical and only three components are stored. The engineering convention, in which the off-diagonal components are summed, is not used.

This definition of curvature is only valid for small deformations to a surface with a small initial curvature.

6.3.41 user_defined_tensor2_2d_variable

A **user_defined_tensor2_2d_variable** is 2D second order tensor variable that is arbitrarily defined. The string shall be suitable for a tabular column heading.

beginecode

```
*)
TYPE user_defined_tensor2_2d_variable = STRING;
END_TYPE;
(*
```

6.3.42 volume_tensor2_3d_variable

A **volume_tensor2_3d_variable** is a 3D second order tensor field variable, that is evaluated at a point within the volume of an element.

beginecode

```
*)
TYPE volume_tensor2_3d_variable = ENUMERATION OF
     (total_strain,
      stress);
END_TYPE;
(*
```

Enumerated item definitions:

total_strain: the elastic strain tensor ε that is the sum of the elastic strain ε^E and the thermal strain ε^T. All strains are expressed using the mathematical convention.

For a linear elastic analysis the elastic strains are given by:

$$\varepsilon_{ij}^E = 1/2 \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right)$$

where:

v is the translation at a point within the element;

x is the position at a point within the element.

For a linear elastic analysis the thermal strains are given by:

$$\varepsilon_{ij}^{T} = \alpha_{ij}(T - T_0)$$

where:

α is the tensor of secant thermal expansion coefficients at a point within the element;

T the temperature at a point within the element.

T_0 the reference temperature.

For a linear elastic analysis the total strains are given by:

$$\varepsilon_{ij} = \varepsilon_{ij}^{E} + \varepsilon_{ij}^{T}$$

This tensor is symmetrical and only six components are stored. The engineering convention, in which pairs of off-diagonal components are summed, is not used.

stress: the stress tensor σ that is defined so that for a linear elastic analysis, the total strain energy per unit volume is given by:

$$\Delta v = \sum_i \sum_j \sigma_{ij} \Delta \varepsilon_{ij}$$

where:

Δv is the change of the strain energy per unit volume at a point within the element;

$\Delta \varepsilon$ is the change of the strain tensor.

6.3.43 aggregated_tensor2_3d_variable

An **aggregated_tensor2_3d_variable** is an aggregated 3D second order tensor field variable that is evaluated at a point within the volume of an element.

beginecode

```
*)
TYPE aggregated_tensor2_3d_variable = ENUMERATION OF
     (rotational_inertia);
END_TYPE;
(*
```

Enumerated item definitions:

rotational_inertia: a 3D second order tensor that is obtained by integration over an element, element group or whole model. The rotational inertia is evaluated about the centre of mass for the domain of integration.

This tensor is symmetrical and only six components are stored.

6.3.44 user_defined_tensor2_3d_variable

A **user_defined_tensor2_3d_variable** is a 3d second order tensor variable that is arbitrarily defined. The string shall be suitable for a tabular column heading.

beginecode

```
*)
TYPE user_defined_tensor2_3d_variable = STRING;
END_TYPE;
(*
```

6.3.45 message_level

A **message_level** is the severity of the message.

beginecode

```
*)
TYPE message_level = ENUMERATION OF
     (error,
      warning,
      note);
END_TYPE;
(*
```

Enumerated item definitions:

error: the message is an error message.

warning: the message is an warning message.

note: the message is an informative note.

6.3.46 surface_3d_state_coordinate_system

A **surface_3d_state_coordinate_system** is the valid types of coordinate systems in a state definition for a surface 3D element.

beginecode

```
*)
TYPE surface_3d_state_coordinate_system = SELECT
     (fea_axis2_placement_3d,
      surface_3d_element_coordinate_system);
END_TYPE;
(*
```

6.3.47 surface_2d_state_coordinate_system

A **surface_2d_state_coordinate_system** is the valid types of coordinate systems in a state definition for a surface 2D element.

beginecode

```
*)
TYPE surface_2d_state_coordinate_system = SELECT
     (fea_axis2_placement_3d,
      surface_2d_element_coordinate_system);
END_TYPE;
(*
```

6.3.48 curve_3d_state_coordinate_system

A **curve_3d_state_coordinate_system** is the valid types of coordinate systems in a state definition for a curve 3D element.

beginecode

```
*)
TYPE curve_3d_state_coordinate_system = SELECT
     (fea_axis2_placement_3d,
      curve_3d_element_coordinate_system);
END_TYPE;
(*
```

6.3.49 curve_2d_state_coordinate_system

A **curve_2d_state_coordinate_system** is the valid types of coordinate systems in a state definition for a curve 2D element.

beginecode

```
*)
```

```
TYPE curve_2d_state_coordinate_system = SELECT
     (fea_axis2_placement_3d,
      curve_2d_element_coordinate_system);
END_TYPE;
(*
```

6.3.50 action_type

An **action_type** is the valid types of nodal actions, that are either forces or moments.

beginecode

```
*)
TYPE action_type = ENUMERATION OF
     (applied_loads,
      residual_loads);
END_TYPE;
(*
```

Enumerated item definitions:

applied_loads: the actions upon the node are applied loads.

residual_loads: the actions upon the node are residual loads resulting from a lack of equilibrium at the node.

6.3.51 volume_3d_element_output_reference

A **volume_3d_element_output_reference** is either an element representation, a group, or an element descriptor for output request or specification for volume 3D elements. If an element descriptor is referenced then all elements that reference that descriptor will have the same output request or specification.

beginecode

```
*)
TYPE volume_3d_element_output_reference = SELECT
     (volume_3d_element_representation,
      volume_3d_element_descriptor,
      volume_3d_element_group);
END_TYPE;
(*
```

6.3.52 volume_2d_element_output_reference

A **volume_2d_element_output_reference** is either an element representation, a group, or an element descriptor for output request or specification for volume 2D elements. If an element descriptor is referenced then all elements that reference that descriptor will have the same output request or specification.

beginecode

```
*)
TYPE volume_2d_element_output_reference = SELECT
     (volume_2d_element_representation,
      volume_2d_element_descriptor,
      volume_2d_element_group);
END_TYPE;
(*
```

6.3.53 surface_3d_element_output_reference

A **surface_3d_element_output_reference** is either an element representation, a group, or an element descriptor for output request or specification for surface 3D elements. If an element descriptor is referenced then all elements that reference that descriptor will have the same output request or specification.

beginecode

```
*)
TYPE surface_3d_element_output_reference = SELECT
     (surface_3d_element_representation,
      surface_3d_element_descriptor,
      surface_3d_element_group);
END_TYPE;
(*
```

6.3.54 surface_2d_element_output_reference

A **surface_2d_element_output_reference** is either an element representation, a group, or an element descriptor for output request or specification for surface 2D elements. If an element descriptor is referenced then all elements that reference that descriptor will have the same output request or specification.

beginecode

```
*)
TYPE surface_2d_element_output_reference = SELECT
     (surface_2d_element_representation,
      surface_2d_element_descriptor,
      surface_2d_element_group);
END_TYPE;
(*
```

6.3.55 curve_3d_element_output_reference

A **curve_3d_element_output_reference** is either an element representation, a group, or an element descriptor for output request or specification for curve 3D elements. If an element

descriptor is referenced then all elements that reference that descriptor will have the same output request or specification.

beginecode

```
*)
TYPE curve_3d_element_output_reference = SELECT
     (curve_3d_element_representation,
      curve_3d_element_descriptor,
      curve_3d_element_group);
END_TYPE;
(*
```

6.3.56 curve_2d_element_output_reference

A **curve_2d_element_output_reference** is either an element representation, a group, or an element descriptor for output request or specification for curve 2d elements. If an element descriptor is referenced then all elements that reference that descriptor will have the same output request or specification.

beginecode

```
*)
TYPE curve_2d_element_output_reference = SELECT
     (curve_2d_element_representation,
      curve_2d_element_descriptor,
      curve_2d_element_group);
END_TYPE;
(*
```

6.3.57 node_output_reference

A **node_output_reference** is either a node representation or a group for output request or specification for nodes.

beginecode

```
*)
TYPE node_output_reference = SELECT
     (node_representation,
      node_group);
END_TYPE;
(*
```

6.4 Finite element analysis control and result entity definitions: Analysis control

A control for a finite element analysis describes the operations carried out upon the model. A finite element analysis application calculates the response of the model to these operations.

6.4.1 control

A **control** is administrative information for a model, and associates analysis controls and results with a unique model.

EXPRESS specification:

```
*)
ENTITY control;
  model_ref                    : fea_model;
  control_id                   : identifier;
  creating_software            : text;
  description                  : text;
  user_defined_control         : SET [1:?] OF text;
UNIQUE
  UR1: model_ref, control_id;
END_ENTITY;
(*
```

Attribute definitions:

model_ref: the model to which the control applies.

control_id: the identifier for the control information.

creating_software: the software which created the control.

description: additional information about the purpose of this control.

user_defined_control: additional information about the analysis control parameters.

Formal propositions:

UR1: the control identifier shall be unique for the model.

6.4.2 analysis_step

An **analysis_step** is the result of a single step in an analysis.

EXPRESS specification:

```
*)
ENTITY analysis_step
  SUPERTYPE OF (ONEOF (control_analysis_step,
                       result_analysis_step));
  analysis_control              : control;
```

```
END_ENTITY;
(*
```

Attribute definitions:

analysis_control: the control which governed the creation of the analysis step.

6.4.3 control_analysis_step

A **control_analysis_step** is a single step in an analysis.

EXPRESS specification:

```
*)
ENTITY control_analysis_step
  SUPERTYPE OF (control_linear_static_analysis_step)
  SUBTYPE OF (analysis_step);
  step_id                      : identifier;
  sequence                     : integer;
  initial_state                : state;
  description                  : text;
UNIQUE
  UR1: analysis_control, sequence;
  UR2: analysis_control, step_id;
END_ENTITY;
(*
```

Attribute definitions:

step_id: the identifier for the step.

sequence: a sequence number that determines the order of processing for the control analysis step.

initial_state: the state of the model at the beginning of the step. This includes information such as reference temperatures and initial strains or stresses. For an initial equilibrium state of zero stress or strain it is only necessary to identify the initial state. The initial state will then have no referencing state definitions.

description: additional information about the purpose of this control.

Formal propositions:

UR1: the sequence number shall be unique for the control.

UR2: the step identifier shall be unique for the control.

6.4.4 symmetry_control

A **symmetry_control** is the type of analysis symmetry.

EXPRESS specification:

```
*)
ENTITY symmetry_control
  SUPERTYPE OF (ONEOF (no_symmetry_control,
                       cylindrical_symmetry_control));
END_ENTITY;
(*
```

6.4.5 no_symmetry_control

A **no_symmetry_control** is that no symmetry control will be used for this analysis.

EXPRESS specification:

```
*)
ENTITY no_symmetry_control
  SUBTYPE OF (symmetry_control);
END_ENTITY;
(*
```

6.4.6 cylindrical_symmetry_control

A **cylindrical_symmetry_control** is that there is cylindrical harmonic analysis symmetry.

EXPRESS specification:

```
*)
ENTITY cylindrical_symmetry_control
  SUBTYPE OF (symmetry_control);
  harmonic                     : cylindrical_harmonic_number;
  phase                        : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

harmonic: the cylindrical harmonic number for an axisymmetric analysis. The initial state may be the zero harmonic, whatever the harmonic of the final state.

phase: the angle θ measured from the i axis, which gives a factor $\cos\theta$ in the amplitude of all field variables for an axisymmetric analysis.

6.4.7 control_linear_static_analysis_step

A **control_linear_static_analysis_step** is that a step is linear static, and specifies the process.

EXPRESS specification:

```
*)
ENTITY control_linear_static_analysis_step
  SUBTYPE OF (control_analysis_step);
  process                      : control_static_load_increment_process;
END_ENTITY;
(*
```

Attribute definitions:

process: the load increment process for the linear static analysis.

Informal propositions:

IP1: the process identifier shall be unique for the control.

6.4.8 control_linear_static_analysis_step_with_harmonic

A **control_linear_static_analysis_step_with_harmonic** is that a step is linear static, and specifies the process.

EXPRESS specification:

```
*)
ENTITY control_linear_static_analysis_step_with_harmonic
  SUBTYPE OF (control_linear_static_analysis_step);
  symmetry                     : cylindrical_symmetry_control;
END_ENTITY;
(*
```

Attribute definitions:

symmetry: the type of symmetry for an analysis.

Informal propositions:

IP1: the harmonic for a linear static analysis step shall be the same as that for its final state.

IP2: a harmonic number shall be supplied for a state of an axisymmetric model.

6.4.9 constraint_element

A **constraint_element** is a constraint applied to a node or between many nodes.

EXPRESS specification:

```
*)
ENTITY constraint_element
  SUPERTYPE OF (ONEOF (single_point_constraint_element,
                       linear_constraint_equation_element,
                       point_constraint,
                       curve_constraint,
                       surface_constraint,
                       solid_constraint));
  element_id                   : identifier;
  steps                        : SET [1:?] OF control_analysis_step;
END_ENTITY;
(*
```

Attribute definitions:

element_id: the identifier for the constraint element.

steps: the analysis control steps to which the constraint element applies.

Informal propositions:

IP1: the constraint element identifier shall be unique for the model.

6.4.10 single_point_constraint_element

A **single_point_constraint_element** is a single point constraint which sets values for one or more degrees of freedom at a single node. This equation has the form:

$$a_i \cdot x_i = b_i$$

where:
x_i are the n nodal freedom values;
a_i are the n nodal freedom coefficients; and
b_i is the constraint equation coefficient.

The values of this constraint are set by a **single_point_constraint_element_values** entity.

EXPRESS specification:

```
*)
ENTITY single_point_constraint_element
  SUBTYPE OF (constraint_element);
  required_node                : node_output_reference;
  coordinate_system            : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_values          : SET [1:?] OF freedom_and_coefficient;
  description                  : text;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_values
                                          , coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_node: the node which is being constrained.

coordinate_system: the coordinate system with respect to which the constraint freedoms are defined.

freedoms_and_values: the freedom and value pair imposed by the constraint.

description: additional information about the single point constraint element.

Formal propositions:

WR1: the constraint coordinate system shall be specified if any of the specified freedoms are not scalar.

6.4.11 linear_constraint_equation_element

A **linear_constraint_equation_element** is a linear multi-point constraint specified as a constraint equation. This equation has the form:

$$\sum_{i=1,n} (a_i \cdot x_i) = b$$

where:
x_i are the n nodal freedom values;
a_i are the n nodal freedom coefficients; and
b is the constraint equation coefficient.

The value the constraint equation coefficient b above is set by a **linear_constraint_equation_element_value** entity.

EXPRESS specification:

```
*)
ENTITY linear_constraint_equation_element
  SUBTYPE OF (constraint_element);
  freedoms_and_coefficients    : SET [1 : ?] OF
                                 linear_constraint_equation_nodal_term;
  description                  : text;
END_ENTITY;
(*
```

Attribute definitions:

freedoms_and_coefficients: the freedoms and their coefficients for each term of the linear constraint equation.

description: additional information about the element.

6.4.12 linear_constraint_equation_nodal_term

A **linear_constraint_equation_nodal_term** is the freedoms and their coefficients at a node of the constraint.

EXPRESS specification:

```
*)
ENTITY linear_constraint_equation_nodal_term;
  node                         : node_representation;
  coordinate_system            : OPTIONAL fea_axis2_placement_3d;
  freedom_and_coefficient_term : freedom_and_coefficient;
  dependent                    : LOGICAL;
WHERE
  WR1: necessary_freedom_coordinate_system (freedom_and_coefficient_term
                                          , coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

node: the node which is associated with the degree of freedom.

coordinate_system: the coordinate system with respect to which the constraint freedom is being defined.

freedom_and_coefficient_term: the constrained freedom at the node, and the coefficient for the constrained freedom at the node (denoted a_i in the constraint equation shown above in **linear_constraint_equation_element**).

dependent: a flag which indicates whether or not the freedom is to be considered dependent by the analysis application, as follows: TRUE it is the dependent degree of freedom, FALSE it is not.

Formal propositions:

WR1: the constraint coordinate system shall be specified, if any of the specified freedoms are not scalar (eg. the warp degree of freedom).

6.4.13 freedom_and_coefficient

A **freedom_and_coefficient** is the degree of freedom and the corresponding coefficient value of a constraint equation.

EXPRESS specification:

```
*)
ENTITY freedom_and_coefficient;
  freedom                      : degree_of_freedom;
  a                            : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

freedom: the degree of freedom that the value is associated with.

a: the nodal freedom coefficient a associated with the degree of freedom, as defined in 6.4.11.

6.4.14 point_constraint

A **point_constraint** sets the values for one or more degree of freedom for all nodes that may be related to the specified point.

EXPRESS specification:

```
*)
ENTITY point_constraint
  SUBTYPE OF (constraint_element);
  required_point             : point;
  coordinate_system          : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_coefficients  : SET [1:?] OF freedom_and_coefficient;
  description                : text;
```

```
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_coefficients,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_point: the point whose associated nodes are being constrained.

coordinate_system: the coordinate system with respect to which the constraint freedoms are defined.

freedoms_and_coefficients: the freedom and coefficient pairs imposed by the constraint.

description: additional information about the point constraint.

Formal propositions:

WR1: the constraint coordinate system shall be specified if any of the specified freedoms are not scalar.

6.4.15 curve_constraint

A **curve_constraint** sets the values for one or more degree of freedom for all nodes that may be related to the specified curve.

EXPRESS specification:

```
*)
ENTITY curve_constraint
  SUBTYPE OF (constraint_element);
  required_curve            : curve;
  coordinate_system         : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_coefficients  : SET [1:?] OF freedom_and_coefficient;
  description               : text;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_coefficients,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_curve: the curve whose associated nodes are being constrained.

coordinate_system: the coordinate system with respect to which the constraint freedoms are defined.

freedoms_and_coefficients: the freedom and coefficient pairs imposed by the constraint.

description: additional information about the curve constraint.

Formal propositions:

WR1: the constraint coordinate system shall be specified if any of the specified freedoms are not scalar.

6.4.16 surface_constraint

A **surface_constraint** sets the values for one or more degree of freedom for all nodes that may be related to the specified surface.

EXPRESS specification:

```
*)
ENTITY surface_constraint
  SUBTYPE OF (constraint_element);
  required_surface            : surface;
  coordinate_system           : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_coefficients   : SET [1:?] OF freedom_and_coefficient;
  description                 : text;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_coefficients,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_surface: the surface whose associated nodes are being constrained.

coordinate_system: the coordinate system with respect to which the constraint freedoms are defined.

freedoms_and_coefficients: the freedom and coefficient pairs imposed by the constraint.

description: additional information about the surface constraint.

Formal propositions:

WR1: the constraint coordinate system shall be specified if any of the specified freedoms are not scalar.

6.4.17 solid_constraint

A **solid_constraint** sets the values for one or more degree of freedom for all nodes that may be related to the specified solid model.

EXPRESS specification:

```
*)
ENTITY solid_constraint
  SUBTYPE OF (constraint_element);
  required_solid              : solid_model;
  coordinate_system           : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_coefficients   : SET [1:?] OF freedom_and_coefficient;
  description                 : text;
```

```
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_coefficients,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_solid: the solid whose associated nodes are being constrained.

coordinate_system: the coordinate system with respect to which the constraint freedoms are defined.

freedoms_and_coefficients: the freedom and coefficient pairs imposed by the constraint.

description: additional information about the solid constraint.

Formal propositions:

WR1: the constraint coordinate system shall be specified if any of the specified freedoms are not scalar.

6.4.18 control_process

A **control_process** is the way in which the model is caused to depart from its initial state.

EXPRESS specification:

```
*)
ENTITY control_process
  SUPERTYPE OF (control_static_load_increment_process);
  process_id                  : identifier;
  description                 : text;
END_ENTITY;
(*
```

Attribute definitions:

process_id: the identifier for the process.

description: additional information about the process.

Informal propositions:

IP1: the process identifier shall be unique for the model.

6.4.19 control_static_load_increment_process

A **control_static_load_increment_process** is a process that is a static load increment, and specifies the final state. All loads other than those that specify the initial state (before the process of applying load for the step) are specified with this entity.

The load increment is the difference between the loads for the final state specified by this entity, and those for the initial state specified by entity **control_analysis_step**.

The load increment shall include any changes in the constraints.

EXPRESS specification:

```
*)
ENTITY control_static_load_increment_process
  SUBTYPE OF (control_process);
  final_input_state            : state;
END_ENTITY;
(*
```

Attribute definitions:

final_input_state: the final equilibrium state for the static load increment. The final state includes all field and node state definitions applied to the model. This state is a specified state without any analysis output information until an analysis has been carried out.

Informal propositions:

IP1: the state identifier shall be unique for the model.

6.4.20 element_sequence

An **element_sequence** is an ordered list of elements.

NOTE – Typical uses of the ordered list will be solution optimization or controlling the presentation of results.

EXPRESS specification:

```
*)
ENTITY element_sequence;
  order_id                : label;
  control                 : control;
  purpose                 : text;
  elements                : LIST [1:?] OF model_or_control_element;
UNIQUE
  UR1: order_id, control;
END_ENTITY;
(*
```

Attribute definitions:

order_id: the identifier for the ordered element list.

control: the control of which the ordered element list is a part.

purpose: additional information about the ordered element list specifying its use.

elements: the ordered list of elements.

Formal propositions:

UR1: the ordered list identifier shall be unique for the control reference in each of the steps.

6.4.21 node_sequence

A **node_sequence** is an ordered list of nodes.

NOTE – Typical uses of the ordered list will be solution optimization or controlling the presentation of results.

EXPRESS specification:

```
*)
ENTITY node_sequence;
  order_id                : label;
  control                 : control;
  purpose                 : text;
  nodes                   : LIST [1:?] OF node_representation;
UNIQUE
  UR1: order_id, control;
END_ENTITY;
(*
```

Attribute definitions:

order_id: the identifier for the ordered node list.

control: the control of which the ordered node list is a part.

purpose: additional information about the ordered node list specifying its use.

nodes: the ordered list of nodes.

Formal propositions:

UR1: the ordered list identifier shall be unique for the control reference in each of the steps.

6.5 Finite element analysis control and result entity definitions: Analysis results

Analysis results for a finite element analysis consists of the response of a model to a control as calculated by a finite element analysis application. All model states calculated by the analysis application shall be linked to a result.

6.5.1 result

A **result** is the administrative information for analysis results.

EXPRESS specification:

```
*)
ENTITY result;
  result_id                   : identifier;
  creating_software           : text;
  description                 : text;
END_ENTITY;
(*
```

Attribute definitions:

result_id: the identifier for the result.

creating_software: the software which created the result.

description: additional information about the result.

Informal propositions:

IP1: the result identifier shall be unique for the model.

6.5.2 result_analysis_step

A **result_analysis_step** is an analysis sub-step. As results are calculated, the analysis step specified by the control is divided into one or more analysis sub-steps. The resulting model state is stored for the end of each sub-step.

For a linear static analysis, analysis calculations produce a single analysis sub-step which is the final state.

EXPRESS specification:

```
*)
ENTITY result_analysis_step
  SUPERTYPE OF (result_linear_static_analysis_sub_step)
  SUBTYPE OF (analysis_step);
  analysis_result              : result;
END_ENTITY;
(*
```

Attribute definitions:

analysis_result: the result in which the step was calculated.

Informal propositions:

IP1: the result identifier shall be unique for the control.

6.5.3 result_linear_static_analysis_sub_step

A **result_linear_static_analysis_sub_step** is a sub-step that is calculated by a linear static analysis, and therefore is the final state of the step.

EXPRESS specification:

```
*)
ENTITY result_linear_static_analysis_sub_step
  SUBTYPE OF (result_analysis_step);
  state                        : calculated_state;
END_ENTITY;
(*
```

Attribute definitions:

state: the calculated state resulting from the linear static analysis sub-step.

6.5.4 control_result_relationship

A **control_result_relationship** is a control, and the result that that was produced from the analysis of the combination of the control and model.

EXPRESS specification:

```
*)
ENTITY control_result_relationship;
  control                      : control_analysis_step;
  result                       : result_analysis_step;
END_ENTITY;
(*
```

Attribute definitions:

control: the control for a given analysis step.

result: the result for a given analysis control.

6.6 Finite element analysis control and result entity definitions: Model state

At any instant, a model has a state. All information defining or requesting analysis variables within a model, such as stress, strain or deflections are associated with a state. Any field within the scope of the analysis, but which is not defined for a state has a value of zero.

6.6.1 state

A **state** is the state of a model as follows:

- the attributes of the entity provide administrative information for the state;
- details of the state are supplied by the **state_definition** entities which reference it.

EXPRESS specification:

```
*)
ENTITY state
  SUPERTYPE OF (ONEOF (specified_state,
                       calculated_state,
                       linearly_superimposed_state,
                       output_request_state));
  state_id                     : identifier;
  description                  : text;
END_ENTITY;
(*
```

Attribute definitions:

state_id: the identifier for the state.

description: additional information about the state.

Informal propositions:

IP1: the state identifier shall be unique for the model.

6.6.2 state_with_harmonic

A **state_with_harmonic** is the state of a model for an analysis that requires the specification of the type of symmetry and a corresponding harmonic.

EXPRESS specification:

```
*)
ENTITY state_with_harmonic
  SUBTYPE OF (state);
  symmetry                    : cylindrical_symmetry_control;
END_ENTITY;
(*
```

Attribute definitions:

symmetry: the type of symmetry for an analysis.

6.6.3 specified_state

A **specified_state** is all the information for the state that is specified, and that none of it is the result of a previous calculation represented within the information model.

EXPRESS specification:

```
*)
ENTITY specified_state
  SUBTYPE OF (state);
END_ENTITY;
(*
```

6.6.4 calculated_state

A **calculated_state** is some of the information for the state that has been calculated by a previous analysis step.

EXPRESS specification:

```
*)
ENTITY calculated_state
  SUBTYPE OF (state);
END_ENTITY;
(*
```

6.6.5 linearly_superimposed_state

A **linearly_superimposed_state** is some of the information for the state that has been calculated by a combination of previously calculated and/or specified steps.

EXPRESS specification:

```
*)
ENTITY linearly_superimposed_state
  SUBTYPE OF (state);
INVERSE
  components                    : SET[1:?] OF state_component
                                  FOR state;
END_ENTITY;
(*
```

Attribute definitions:

components: the states from which this state is created.

6.6.6 state_component

A **state_component** is a component of a linearly superimposed state.

EXPRESS specification:

```
*)
ENTITY state_component
  SUBTYPE OF (state);
  state                         : linearly_superimposed_state;
  factor                        : context_dependent_measure;
END_ENTITY;
(*
```

Attribute definitions:

state: the state that is to be combined.

factor: a weighting factor to scale information in the state.

6.6.7 output_request_state

An **output_request_state** is the computer readable output to be produced by an finite element analysis system as follows:

- the attributes of the entity provide administrative information for the state of output;
- details of the information to be produced by the finite element analysis system are supplied by the **state_definition** entities which reference it.

EXPRESS specification:

```
*)
ENTITY output_request_state
  SUBTYPE OF (state);
  steps                    : SET [1:?] OF control_analysis_step;
END_ENTITY;
(*
```

Attribute definitions:

steps: the analysis control steps to which the output requests apply.

Informal propositions:

IP1: the output state identifier shall be unique for the control reference in each of the steps.

6.7 Finite element analysis control and result entity definitions: State definition

A state definition has two roles:

- As a state definition with value it can reference a **defined_state**, **calculated_state**, or **linearly_superimposedstate** entity thereby defining the variables within the model for that state;

- As a state definition without value it can reference an **output_request_state** entity to define the information to be calculated by the finite element analysis application for one or more **control_anlaysis_state**.

  NOTE – An EXPRESS-G Partial Model illustrating the relationships of the state definitions is shown in figure 62

Figure 62 – State definition EXPRESS-G partial model

6.7.1 state_definition

A **state_definition** is both information known before and after an analysis (such as external loadings, and prescribed values of variables at a boundary), and therefore either requests or represents information calculated by the analysis (such as element stresses, and values of variables at unconstrained nodes). In this manner a **state_definition** output from one **step** may be used as input for a subsequent **state_definition**.

EXPRESS specification:

```
*)
ENTITY state_definition
  SUPERTYPE OF (ONEOF (field_variable_definition,
                       nodal_freedom_and_value_definition,
                       element_nodal_freedom_actions,
                       point_freedom_and_value_definition,
                       curve_freedom_and_value_definition,
                       surface_freedom_and_value_definition,
                       solid_freedom_and_value_definition,
                       linear_constraint_equation_element_value,
                       single_point_constraint_element_values,
                       analysis_message));
  defined_state              : state;
END_ENTITY;
(*
```

Attribute definitions:

defined_state: the model state that aggregated the state definitions.

6.7.2 field_variable_definition

A **field_variable_definition** is a state of the model that specifies values for a field variable pertaining to the elements of the model.

The field variable information may be:

- evaluated a point within the volume of an element;
- aggregated through the section of a surface or curve element at a point on the element;
- aggregated within an element, a group of elements, or an entire model;
- evaluated at a point on an element boundary (element face of a volume element or element edge of a surface element);
- aggregated over an element boundary.

EXPRESS specification:

```
*)
ENTITY field_variable_definition
  SUPERTYPE OF (ONEOF (field_variable_element_definition,
```

```
                      field_variable_element_group_value,
                      field_variable_whole_model_value,
                      field_variable_node_definition))
  SUBTYPE OF (state_definition);
END_ENTITY;
(*
```

6.7.3 field_variable_element_definition

A **field_variable_element_definition** defines the state of the model by specifying the values of a variable within an element.

EXPRESS specification:

```
*)
ENTITY field_variable_element_definition
  SUPERTYPE OF (ONEOF (volume_3d_element_field_variable_definition,
                      volume_2d_element_field_variable_definition,
                      surface_3d_element_field_variable_definition,
                      surface_2d_element_field_variable_definition,
                      curve_3d_element_field_variable_definition,
                      curve_2d_element_field_variable_definition))
  SUBTYPE OF (field_variable_definition);
END_ENTITY;
(*
```

6.7.4 volume_3d_element_field_variable_definition

A **volume_3d_element_field_variable_definition** is a state of the model that specifies the values of a variable within a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_field_variable_definition
  SUPERTYPE OF (ONEOF (
    volume_3d_element_location_point_variable_values,
    volume_3d_whole_element_variable_value,
    volume_3d_element_constant_specified_variable_value,
    volume_3d_element_nodal_specified_variable_values,

    volume_3d_element_boundary_location_point_variable_values,
    volume_3d_element_boundary_whole_face_variable_value,
    volume_3d_element_boundary_constant_specified_variable_value,
    volume_3d_element_boundary_nodal_specified_variable_values,

    volume_3d_element_boundary_edge_location_point_volume_variable_values,
    volume_3d_element_boundary_edge_whole_edge_variable_value,
    volume_3d_element_boundary_edge_constant_specified_volume_variable_value,
    volume_3d_element_boundary_edge_nodal_specified_variable_values))

  SUBTYPE OF (field_variable_element_definition);
```

```
  element                       :
               volume_3d_element_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified, or the element for which values are to be calculated.

6.7.5 volume_3d_element_location_point_variable_values

A **volume_3d_element_location_point_variable_values** is a state of the model that specifies the values of a variable at location points within a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_location_point_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  basis                         : BOOLEAN;
  values_and_locations          : SET[1:?] OF
                                  volume_3d_element_value_and_location;
  variable                      : volume_variable;
WHERE
  WR1: consistent_set_values (values_and_locations,variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.6 volume_3d_element_value_and_location

A **volume_3d_element_value_and_location** is the value of a variable at a location point within a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_value_and_location;
  simple_value                : field_value;
  location                    : volume_element_location;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.7 volume_3d_whole_element_variable_value

A **volume_3d_whole_element_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over a volume 3D element.

The value represented by this entity does not include a contribution from the aggregation of boundary variables.

EXPRESS specification:

```
*)
ENTITY volume_3d_whole_element_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value,variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.8 volume_3d_element_constant_specified_variable_value

A **volume_3d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_constant_specified_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_variable;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value,variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.9 volume_3d_element_nodal_specified_variable_values

A **volume_3d_element_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes of a volume 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

The values are supplied for the each of the nodes in the attribute **required_node_list** of entity **element_representation**. Further values are supplied for each node in attribute **additional_node_list** which is not a dummy.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_nodal_specified_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : volume_variable;
WHERE
  WR1: consistent_list_values (values,variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.10 volume_3d_element_boundary_location_point_variable_values

A **volume_3d_element_boundary_location_point_variable_values** is a state of the model that specifies the values of a variable at location points over the element face of a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_location_point_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  basis                         : BOOLEAN;
  values_and_locations          : SET[1:?] OF
                                  volume_3d_element_value_and_location;
  variable                      : boundary_variable;
  element_face                  : volume_3d_face;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element face.

6.7.11 volume_3d_element_boundary_whole_face_variable_value

A **volume_3d_element_boundary_whole_face_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over the element face of a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_whole_face_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_face                : volume_3d_face;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

coordinate_system: the coordinate system for the value.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.12 volume_3d_element_boundary_constant_specified_-variable_value

A **volume_3d_element_boundary_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over the whole of a volume 3D element face.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_constant_specified_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_variable;
  element_face                : volume_3d_face;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.13 volume_3d_element_boundary_nodal_specified_-variable_values

A **volume_3d_element_boundary_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes on an element face of a volume 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

Values are supplied for the nodes on the element face, in the order established graphically in 5.8.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_nodal_specified_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : boundary_variable;
  element_face                  : volume_3d_face;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.14 volume_3d_element_boundary_edge_location_point_-volume_variable_values

A **volume_3d_element_boundary_edge_location_point_volume_variable_values** is a state of the model that specifies the values of a variable at location points within the volume along an element edge of a volume 3D element.

EXPRESS specification:

```
*)
ENTITY
 volume_3d_element_boundary_edge_location_point_volume_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 volume_3d_element_value_and_location;
  variable                     : boundary_edge_variable;
  element_edge                 : volume_3d_edge;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element edge.

6.7.15 volume_3d_element_boundary_edge_whole_edge_-variable_value

A **volume_3d_element_boundary_edge_whole_edge_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation along an element edge of a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_edge_whole_edge_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_edge                : volume_3d_edge;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.16 volume_3d_element_boundary_edge_constant_-specified_volume_variable_value

A **volume_3d_element_boundary_edge_constant_specified_volume_variable_value** is a state of the model that is defined by specifying the value of a variable at a point on a volume, which is constant along a volume 3D element edge.

EXPRESS specification:

```
*)
ENTITY
 volume_3d_element_boundary_edge_constant_specified_volume_variable_value
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_edge_variable;
  element_edge                : volume_3d_edge;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.17 volume_3d_element_boundary_edge_nodal_specified_-variable_values

A **volume_3d_element_boundary_edge_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes along an element edge of a volume 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

Values are supplied for the nodes on the element edge, in the order established graphically in 5.8.

EXPRESS specification:

```
*)
ENTITY volume_3d_element_boundary_edge_nodal_specified_variable_values
  SUBTYPE OF (volume_3d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : boundary_edge_variable;
  element_edge                  : volume_3d_edge;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.18 volume_2d_element_field_variable_definition

A **volume_2d_element_field_variable_definition** is a state of the model that specifies the values of a variable within a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_field_variable_definition
  SUPERTYPE OF (ONEOF (
    volume_2d_element_location_point_variable_values,
    volume_2d_whole_element_variable_value,
    volume_2d_element_constant_specified_variable_value,
    volume_2d_element_nodal_specified_variable_values,

    volume_2d_element_boundary_location_point_variable_values,
    volume_2d_element_boundary_whole_face_variable_value,
    volume_2d_element_boundary_constant_specified_variable_value,
```

```
    volume_2d_element_boundary_nodal_specified_variable_values,

    volume_2d_element_boundary_edge_location_point_volume_variable_values,
    volume_2d_element_boundary_edge_whole_edge_variable_value,
    volume_2d_element_boundary_edge_constant_specified_volume_variable_value,
    volume_2d_element_boundary_edge_nodal_specified_variable_values))

  SUBTYPE OF (field_variable_element_definition);

  element                      :
     volume_2d_element_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified, or the element for which values are to be calculated.

6.7.19 volume_2d_element_location_point_variable_values

A **volume_2d_element_location_point_variable_values** is a state of the model that specifies the values of a variable at location points within a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_location_point_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 volume_2d_element_value_and_location;
  variable                     : volume_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.20 volume_2d_element_value_and_location

A **volume_2d_element_value_and_location** is a value of a variable at a location point within a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_value_and_location;
  simple_value                : field_value;
  location                    : volume_element_location;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.21 volume_2d_whole_element_variable_value

A **volume_2d_whole_element_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over a volume 2D element.

The value represented by this entity does not include a contribution from the aggregation of boundary variables.

EXPRESS specification:

```
*)
ENTITY volume_2d_whole_element_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.22 volume_2d_element_constant_specified_variable_value

A **volume_2d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over the whole of a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_constant_specified_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_variable;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.23 volume_2d_element_nodal_specified_variable_values

A **volume_2d_element_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes of a volume 2D element. The variable is interpolated within

the element using the appropriate shape functions specified for the variable.

The values are supplied for the each of the nodes in the attribute **required_node_list** of entity **element_representation**. Further values are supplied for each node in attribute **additional_node_list** which is not a dummy.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_nodal_specified_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  values                      : LIST [1:?] OF field_value;
  additional_node_values      : BOOLEAN;
  variable                    : volume_variable;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.24 volume_2d_element_boundary_location_point_variable_values

A **volume_2d_element_boundary_location_point_variable_values** is a state of the model that specifies the values of a variable at location points over the element face of a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_boundary_location_point_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  basis                       : BOOLEAN;
  values_and_locations        : SET[1:?] OF
                                volume_2d_element_value_and_location;
  variable                    : boundary_variable;
  element_face                : volume_2d_face;
WHERE
```

```
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element face.

6.7.25 volume_2d_element_boundary_whole_face_variable_-value

A **volume_2d_element_boundary_whole_face_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over the element face of a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_boundary_whole_face_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_face                : volume_2d_face;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.26 volume_2d_element_boundary_constant_specified_-variable_value

A **volume_2d_element_boundary_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over the whole of a volume 2D element face.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_boundary_constant_specified_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_variable;
  element_face                : volume_2d_face;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.27 volume_2d_element_boundary_nodal_specified_-variable_values

A **volume_2d_element_boundary_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes on an element face of a volume 2D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

Values are supplied for the nodes on the element face, in the order established graphically in 5.8.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_boundary_nodal_specified_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  values                       : LIST [1:?] OF
                                 field_value;
  variable                     : boundary_variable;
  additional_node_values       : BOOLEAN;
  element_face                 : volume_2d_face;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

variable: the type of field variable being specified.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

element_face: the element face for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.28 volume_2d_element_boundary_edge_location_point_-volume_variable_values

A **volume_2d_element_boundary_edge_location_point_volume_variable_values** is a state of the model that specifies the values of a variable at location points within the volume along an element edge of a volume 2d element.

EXPRESS specification:

```
*)
ENTITY
 volume_2d_element_boundary_edge_location_point_volume_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 volume_2d_element_value_and_location;
  variable                     : boundary_edge_variable;
  element_edge                 : volume_2d_edge;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element edge.

6.7.29 volume_2d_element_boundary_edge_whole_edge_-variable_value

A **volume_2d_element_boundary_edge_whole_edge_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation along an element edge of a volume 2d element.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_boundary_edge_whole_edge_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_edge                : volume_2d_edge;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.30 volume_2d_element_boundary_edge_constant_-specified_volume_variable_value

A **volume_2d_element_boundary_edge_constant_specified_volume_variable_value** is a state of the model that is defined by specifying the value of a variable at a point on a volume, which is constant along a volume 2d element edge.

EXPRESS specification:

```
*)
ENTITY
 volume_2d_element_boundary_edge_constant_specified_volume_variable_value
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_edge_variable;
  element_edge                : volume_2d_edge;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.31 volume_2d_element_boundary_edge_nodal_specified_-variable_values

A **volume_2d_element_boundary_edge_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes along an element edge of a volume 2d

element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

Values are supplied for the nodes on the element edge, in the order established graphically in 5.8.

EXPRESS specification:

```
*)
ENTITY volume_2d_element_boundary_edge_nodal_specified_variable_values
  SUBTYPE OF (volume_2d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : boundary_edge_variable;
  element_edge                  : volume_2d_edge;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_edge: the element edge of the volume for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.32 surface_3d_element_field_variable_definition

A **surface_3d_element_field_variable_definition** is a state of the model that specifies the values of a variable within a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_field_variable_definition
 SUPERTYPE OF (ONEOF (
 surface_3d_element_location_point_volume_variable_values,
```

```
 surface_3d_element_location_point_variable_values,
 surface_3d_whole_element_variable_value,
 surface_3d_element_constant_specified_variable_value,
 surface_3d_element_constant_specified_volume_variable_value,
 surface_3d_element_nodal_specified_variable_values,

 surface_3d_element_boundary_location_point_surface_variable_values,
 surface_3d_element_boundary_whole_face_variable_value,
 surface_3d_element_boundary_constant_specified_variable_value,
 surface_3d_element_boundary_constant_specified_surface_variable_value,
 surface_3d_element_boundary_nodal_specified_variable_values,

 surface_3d_element_boundary_edge_location_point_surface_variable_values,
 surface_3d_element_boundary_edge_location_point_variable_values,
 surface_3d_element_boundary_edge_whole_edge_variable_value,
 surface_3d_element_boundary_edge_constant_specified_variable_value,
 surface_3d_element_boundary_edge_constant_specified_surface_variable_value,
 surface_3d_element_boundary_edge_nodal_specified_variable_values))

  SUBTYPE OF (field_variable_element_definition);

  element                      :
    surface_3d_element_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified, or the element for which values are to be calculated.

6.7.33 surface_3d_element_location_point_volume_variable_values

A **surface_3d_element_location_point_volume_variable_values** is a state of the model that specifies the values of a variable at location points within the volume of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_location_point_volume_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                       surface_3d_element_value_and_volume_location;
  variable                     : volume_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.34 surface_3d_element_value_and_location

A **surface_3d_element_value_and_location** is a value of a variable at a location point within the 2D field of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_value_and_location;
  simple_value                 : field_value;
  location                     : surface_element_location;
  coordinate_system            : OPTIONAL surface_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.35 surface_3d_element_value_and_volume_location

A **surface_3d_element_value_and_volume_location** is a value of a variable at a location point within the volume of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_value_and_volume_location;
  simple_value                 : field_value;
```

```
  location                      : surface_volume_element_location;
  coordinate_system             : OPTIONAL surface_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.36 surface_3d_element_location_point_variable_values

A **surface_3d_element_location_point_variable_values** is a state of the model that specifies the values of a variable at location points over the surface of a surface 3D element. The variables are obtained by aggregation through the section at each surface point.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_location_point_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  basis                         : BOOLEAN;
  values_and_locations          : SET[1:?] OF
                                  surface_3d_element_value_and_location;
  variable                      : surface_element_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.37 surface_3d_whole_element_variable_value

A **surface_3d_whole_element_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over a surface 3D element.

The value represented by this entity does not include a contribution from the aggregation of boundary variables.

EXPRESS specification:

```
*)
ENTITY surface_3d_whole_element_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.38 surface_3d_element_constant_specified_variable_value

A **surface_3d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over the surface of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_constant_specified_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : surface_element_variable;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
```

```
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.39 surface_3d_element_constant_specified_volume_-variable_value

A **surface_3d_element_constant_specified_volume_variable_value** is a state of the model that specifies the value of a variable which is constant within the volume of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_constant_specified_volume_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                 : field_value;
  variable                     : volume_variable;
  coordinate_system            : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.40 surface_3d_element_nodal_specified_variable_values

A **surface_3d_element_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes of a surface 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

The values are supplied for the each of the nodes in the attribute **required_node_list** of entity **element_representation**. Further values are supplied for each node in attribute **additional_node_list** which is not a dummy.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_nodal_specified_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  values                       : LIST [1:?] OF
                                 field_value;
  additional_node_values       : BOOLEAN;
  variable                     : surface_element_variable;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.41 surface_3d_element_boundary_location_point_surface_variable_values

A **surface_3d_element_boundary_location_point_surface_variable_values** is a state of the model that specifies the values of a variable at location points on the surface of a surface 3D

element.

EXPRESS specification:

```
*)
ENTITY
 surface_3d_element_boundary_location_point_surface_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
               surface_3d_element_value_and_location;
  variable                     : boundary_variable;
  element_face                 : surface_3d_face;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.42 surface_3d_element_boundary_whole_face_variable_-value

A **surface_3d_element_boundary_whole_face_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation on the surface of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_boundary_whole_face_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                 : field_value;
```

```
  variable                    : boundary_aggregated_variable;
  element_face                : surface_3d_face;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being output.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.43 surface_3d_element_boundary_constant_specified_-surface_variable_value

A **surface_3d_element_boundary_constant_specified_surface_variable_value** is a state of the model that specifies the value of a variable at a point on a surface, which is constant over the surface of the surface 3D element.

EXPRESS specification:

```
*)
ENTITY
 surface_3d_element_boundary_constant_specified_surface_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_variable;
  element_face                : surface_3d_face;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
```

```
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically shown in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.44 surface_3d_element_boundary_constant_specified_-variable_value

A **surface_3d_element_boundary_constant_specified_variable_value** is a state of the model that specifies the value of a variable aggregated through the section, which is constant on the surface of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_boundary_constant_specified_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_face                : surface_3d_face;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.45 surface_3d_element_boundary_nodal_specified_-variable_values

A **surface_3d_element_boundary_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes on the surface of a surface 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_boundary_nodal_specified_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  values                       : LIST [1:?] OF
                                 field_value;
  additional_node_values       : BOOLEAN;
  variable                     : boundary_variable;
  element_face                 : surface_3d_face;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.46 surface_3d_element_boundary_edge_location_point_-surface_variable_values

A **surface_3d_element_boundary_edge_location_point_surface_variable_values** is a state of the model that specifies the values of a variable at location points within the surface along an element edge of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY
 surface_3d_element_boundary_edge_location_point_surface_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 surface_3d_element_value_and_volume_location;
  variable                     : boundary_edge_variable;
  element_edge                 : surface_3d_edge;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element edge.

6.7.47 surface_3d_element_boundary_edge_location_point_-variable_values

A **surface_3d_element_boundary_edge_location_point_variable_values** is a state of the model that specifies the values of a variable aggregated through the section at location points along an element edge of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY
 surface_3d_element_boundary_edge_location_point_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 surface_3d_element_value_and_location;
  variable                     : boundary_edge_variable;
  element_edge                 : surface_3d_edge;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element edge.

6.7.48 surface_3d_element_boundary_edge_whole_edge_-variable_value

A **surface_3d_element_boundary_edge_whole_edge_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation along an element edge of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_boundary_edge_whole_edge_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_edge                : surface_3d_edge;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.49 surface_3d_element_boundary_edge_constant_specified_-surface_variable_value

A **surface_3d_element_boundary_edge_constant_specified_surface_variable_value** is a state of the model that is defined by specifying the value of a variable at a point on a surface, which is constant along a surface 3D element edge.

EXPRESS specification:

```
*)
ENTITY
 surface_3d_element_boundary_edge_constant_specified_surface_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_edge_variable;
  element_edge                : surface_3d_edge;
  coordinate_system           : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.50 surface_3d_element_boundary_edge_constant_specified_-variable_value

A **surface_3d_element_boundary_edge_constant_specified_variable_value** is a state of the model that specifies the value of a variable aggregated through the section, which is constant along a surface 3D element edge.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_boundary_edge_constant_specified_variable_value
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  simple_value                : field_value;
```

```
  variable                      : boundary_edge_variable;
  element_edge                  : surface_3d_edge;
  coordinate_system             : OPTIONAL surface_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.51 surface_3d_element_boundary_edge_nodal_specified_variable_values

A **surface_3d_element_boundary_edge_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes along an element edge of a surface 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

Values are supplied for the nodes on the element edge, in the order established graphically in 5.8.

EXPRESS specification:

```
*)
ENTITY surface_3d_element_boundary_edge_nodal_specified_variable_values
  SUBTYPE OF (surface_3d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : boundary_edge_variable;
```

```
  element_edge                : surface_3d_edge;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.52 surface_2d_element_field_variable_definition

A **surface_2d_element_field_variable_definition** is a state of the model that specifies the values of a variable within a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_field_variable_definition
  SUPERTYPE OF (ONEOF (
 surface_2d_element_location_point_volume_variable_values,
 surface_2d_element_location_point_variable_values,
 surface_2d_whole_element_variable_value,
 surface_2d_element_constant_specified_variable_value,
 surface_2d_element_constant_specified_volume_variable_value,
 surface_2d_element_nodal_specified_variable_values,

 surface_2d_element_boundary_location_point_surface_variable_values,
 surface_2d_element_boundary_whole_face_variable_value,
 surface_2d_element_boundary_constant_specified_variable_value,
 surface_2d_element_boundary_constant_specified_surface_variable_value,
 surface_2d_element_boundary_nodal_specified_variable_values,

 surface_2d_element_boundary_edge_location_point_surface_variable_values,
 surface_2d_element_boundary_edge_location_point_variable_values,
```

```
 surface_2d_element_boundary_edge_whole_edge_variable_value,
 surface_2d_element_boundary_edge_constant_specified_variable_value,
 surface_2d_element_boundary_edge_constant_specified_surface_variable_value,
 surface_2d_element_boundary_edge_nodal_specified_variable_values))

  SUBTYPE OF (field_variable_element_definition);
  element                      :
    surface_2d_element_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified, or the element for which values are to be calculated.

6.7.53 surface_2d_element_location_point_volume_variable_values

A **surface_2d_element_location_point_volume_variable_values** is a state of the model that specifies the values of a variable at location points within the volume of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_location_point_volume_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 surface_2d_element_value_and_volume_location;
  variable                     : volume_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.54 surface_2d_element_value_and_location

A **surface_2d_element_value_and_location** is a value of a variable at a location point within the 2D field of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_value_and_location;
  simple_value                : field_value;
  location                    : surface_element_location;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.55 surface_2d_element_value_and_volume_location

A **surface_2d_element_value_and_volume_location** is a value of a variable at a location point within the volume of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_value_and_volume_location;
  simple_value                : field_value;
  location                    : surface_volume_element_location;
  coordinate_system           : OPTIONAL surface_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.56 surface_2d_element_location_point_variable_values

A **surface_2d_element_location_point_variable_values** is a state of the model that specifies the values of a variable at location points over the surface of a surface 2D element. The variables are obtained by aggregation through the section at each surface point.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_location_point_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 surface_2d_element_value_and_location;
  variable                     : surface_element_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.57 surface_2d_whole_element_variable_value

A **surface_2d_whole_element_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over a surface 2D element.

The value represented by this entity does not include a contribution from the aggregation of boundary variables.

EXPRESS specification:

```
*)
ENTITY surface_2d_whole_element_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                 : field_value;
  variable                     : volume_aggregated_variable;
  coordinate_system            : OPTIONAL surface_2d_element_coordinate_system;
WHERE
```

```
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified. item[coordinate_system] The coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.58 surface_2d_element_constant_specified_variable_value

A **surface_2d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant over the surface of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_constant_specified_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : surface_element_variable;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.59 surface_2d_element_constant_specified_volume_-variable_value

A **surface_2d_element_constant_specified_volume_variable_value** is a state of the model that specifies the value of a variable which is a constant within the volume of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_constant_specified_volume_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                 : field_value;
  variable                     : volume_variable;
  coordinate_system            : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.60 surface_2d_element_nodal_specified_variable_values

A **surface_2d_element_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes of a surface 2D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

The values are supplied for the each of the nodes in the attribute **required_node_list** of entity **element_representation**. Further values are supplied for each node in attribute **additional_-node_list** which is not a dummy.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_nodal_specified_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : surface_element_variable;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.61 surface_2d_element_boundary_location_point_surface_-variable_values

A **surface_2d_element_boundary_location_point_surface_variable_values** is a state of the model that specifies the values of a variable at location points on the surface of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY
 surface_2d_element_boundary_location_point_surface_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  basis                         : BOOLEAN;
  values_and_locations          : SET[1:?] OF
                                  surface_2d_element_value_and_location;
  variable                      : boundary_variable;
  element_face                  : surface_2d_face;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
```

```
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.62 surface_2d_element_boundary_whole_face_variable_value

A **surface_2d_element_boundary_whole_face_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation on the surface of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_boundary_whole_face_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_face                : surface_2d_face;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.63 surface_2d_element_boundary_constant_specified_-surface_variable_value

A **surface_2d_element_boundary_constant_specified_surface_variable_value** is a state of the model that specifies the value of a variable at a point on a surface, which is constant over the surface of the surface 2D element.

EXPRESS specification:

```
*)
ENTITY
 surface_2d_element_boundary_constant_specified_surface_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_variable;
  element_face                : surface_2d_face;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.64 surface_2d_element_boundary_constant_specified_-variable_value

A **surface_2d_element_boundary_constant_specified_variable_value** is a state of the model that specifies the value of a variable aggregated through the section, which is constant on the surface of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_boundary_constant_specified_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_aggregated_variable;
  element_face                : surface_2d_face;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.65 surface_2d_element_boundary_nodal_specified_-variable_values

A **surface_2d_element_boundary_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes on the surface of a surface 2D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_boundary_nodal_specified_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : boundary_variable;
  element_face                  : surface_2d_face;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_face: the element face of the surface for which the values are specified.

The relationship between the element face number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.66 surface_2d_element_boundary_edge_location_point_-surface_variable_values

A **surface_2d_element_boundary_edge_location_point_surface_variable_values** is a state of the model that specifies the values of a variable at location points along an element edge of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY
 surface_2d_element_boundary_edge_location_point_surface_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 surface_2d_element_value_and_volume_location;
  variable                     : boundary_edge_variable;
  element_edge                 : surface_2d_edge;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.67 surface_2d_element_boundary_edge_location_point_-variable_values

A **surface_2d_element_boundary_edge_location_point_variable_values** is a state of the model that specifies the values of a variable aggregated through the section at location points along an element edge of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY
 surface_2d_element_boundary_edge_location_point_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
```

```
  values_and_locations         : SET[1:?] OF
                                 surface_2d_element_value_and_location;
  variable                     : boundary_edge_variable;
  element_edge                 : surface_2d_edge;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

Informal propositions:

IP1: each location in the list shall lie on the specified element edge.

6.7.68 surface_2d_element_boundary_edge_whole_edge_-variable_value

A **surface_2d_element_boundary_edge_whole_edge_variable_value** that specifies a state of the model that specifies the value of a variable which is obtained by aggregation along an element edge of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_boundary_edge_whole_edge_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                 : field_value;
  variable                     : boundary_aggregated_variable;
  element_edge                 : surface_2d_edge;
  coordinate_system            : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
```

```
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.69 surface_2d_element_boundary_edge_constant_specified_surface_variable_value

A **surface_2d_element_boundary_edge_constant_specified_surface_variable_value** is a state of the model that specifies the value of a variable at a point on a surface, which is a constant along a surface 2D element edge.

EXPRESS specification:

```
*)
ENTITY
 surface_2d_element_boundary_edge_constant_specified_surface_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_edge_variable;
  element_edge                : surface_2d_edge;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.70 surface_2d_element_boundary_edge_constant_specified_variable_value

A **surface_2d_element_boundary_edge_constant_specified_variable_value** is a state of the model that specifies the value of a variable aggregated through the section, which is constant along a surface 2D element edge.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_boundary_edge_constant_specified_variable_value
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : boundary_edge_variable;
  element_edge                : surface_2d_edge;
  coordinate_system           : OPTIONAL surface_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.71 surface_2d_element_boundary_edge_nodal_specified_variable_values

A **surface_2d_element_boundary_edge_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes along an element edge of a surface 2D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

Values are supplied for the nodes on the element edge, in the order established graphically in 5.8.

EXPRESS specification:

```
*)
ENTITY surface_2d_element_boundary_edge_nodal_specified_variable_values
  SUBTYPE OF (surface_2d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : boundary_edge_variable;
  element_edge                  : surface_2d_edge;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

element_edge: the element edge of the surface for which the values are specified.

The relationship between the element edge number and the element node sequence numbers is established graphically in 5.8.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.72 curve_3d_element_field_variable_definition

A **curve_3d_element_field_variable_definition** is a state of the model that specifies the values of a variable within a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_field_variable_definition
 SUPERTYPE OF (ONEOF (
            curve_3d_element_location_point_volume_variable_values,
            curve_3d_element_location_point_variable_values,
            curve_3d_whole_element_variable_value,
            curve_3d_element_constant_specified_variable_value,
            curve_3d_element_constant_specified_volume_variable_value,
            curve_3d_element_nodal_specified_variable_values))
  SUBTYPE OF (field_variable_element_definition);
  element                      :
      curve_3d_element_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified, or the element for which values are to be calculated.

6.7.73 curve_3d_element_location_point_volume_variable_values

A **curve_3d_element_location_point_volume_variable_values** is a state of the model that specifies the values of a variable at location points within the volume of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_location_point_volume_variable_values
  SUBTYPE OF (curve_3d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 curve_3d_element_value_and_volume_location;
  variable                     : volume_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
```

```
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.74 curve_3d_element_value_and_location

A **curve_3d_element_value_and_location** is a value of a variable along the 1D field of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_value_and_location;
  simple_value                : field_value;
  location                    : curve_element_location;
  coordinate_system           : OPTIONAL curve_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.75 curve_3d_element_value_and_volume_location

A **curve_3d_element_value_and_volume_location** is an aggregated value of a variable at a location point in the volume of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_value_and_volume_location;
  simple_value                : field_value;
  location                    : curve_volume_element_location;
  coordinate_system           : OPTIONAL curve_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.76 curve_3d_element_location_point_variable_values

A **curve_3d_element_location_point_variable_values** is a state of the model that specifies the values of a variable at location points over the curve of a curve 3D element. The variables are obtained by aggregation through the section at each curve point.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_location_point_variable_values
  SUBTYPE OF (curve_3d_element_field_variable_definition);
  basis                       : BOOLEAN;
  values_and_locations        : SET[1:?] OF
                                curve_3d_element_value_and_location;
  variable                    : curve_element_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.77 curve_3d_whole_element_variable_value

A **curve_3d_whole_element_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over a curve 3D element. The value represented by this entity does not include a contribution from the aggregation of boundary variables.

EXPRESS specification:

```
*)
ENTITY curve_3d_whole_element_variable_value
  SUBTYPE OF (curve_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL curve_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.78 curve_3d_element_constant_specified_variable_value

A **curve_3d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant along the length of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_constant_specified_variable_value
  SUBTYPE OF (curve_3d_element_field_variable_definition);
  simple_value                : field_value;
```

```
  variable                    : curve_element_variable;
  coordinate_system           : OPTIONAL curve_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.79 curve_3d_element_constant_specified_volume_variable_-value

A **curve_3d_element_constant_specified_volume_variable_value** is a state of the model that specifies the value of a variable at a point within a volume which is constant within the volume of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_constant_specified_volume_variable_value
  SUBTYPE OF (curve_3d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_variable;
  coordinate_system           : OPTIONAL curve_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.80 curve_3d_element_nodal_specified_variable_values

A **curve_3d_element_nodal_specified_variable_values** is a state of the model that specifies the value of a variable for the nodes of a curve 3D element. The variable is interpolated within the element using the appropriate shape functions specified for the variable.

The values are supplied for the each of the nodes in the attribute **required_node_list** of entity **element_representation**. Further values are supplied for each node in attribute **additional_node_list** which is not a dummy.

EXPRESS specification:

```
*)
ENTITY curve_3d_element_nodal_specified_variable_values
  SUBTYPE OF (curve_3d_element_field_variable_definition);
  values                        : LIST [1:?] OF
                                  field_value;
  additional_node_values        : BOOLEAN;
  variable                      : curve_element_variable;
WHERE
  WR1: consistent_list_values (values, variable);
  WR2: appropriate_list_value_existence (values,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

values: the values of the field variable.

additional_node_values: indicates whether values are given for all nodes of the element (TRUE), or for just the required nodes (FALSE).

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.81 curve_2d_element_field_variable_definition

A **curve_2d_element_field_variable_definition** is a state of the model that specifies the values of a variable within a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_field_variable_definition
 SUPERTYPE OF (ONEOF (
            curve_2d_element_location_point_volume_variable_values,
            curve_2d_element_location_point_variable_value,
            curve_2d_whole_element_variable_value,
            curve_2d_element_constant_specified_variable_value,
            curve_2d_element_constant_specified_volume_variable_value))
  SUBTYPE OF (field_variable_element_definition);
  element                      :
      curve_2d_element_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified, or the element for which values are to be calculated.

6.7.82 curve_2d_element_location_point_volume_variable_values

A **curve_2d_element_location_point_volume_variable_values** is a state of the model that specifies the values of a variable at location points within the volume of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_location_point_volume_variable_values
  SUBTYPE OF (curve_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 curve_2d_element_value_and_volume_location;
  variable                     : curve_element_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.83 curve_2d_element_value_and_location

A **curve_2d_element_value_and_location** is a value of a variable along the 1D field of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_value_and_location;
  simple_value                : field_value;
  location                    : curve_section_element_location;
  coordinate_system           : OPTIONAL curve_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.84 curve_2d_element_value_and_volume_location

A **curve_2d_element_value_and_volume_location** is an aggregated value of a variable at a location point in the volume of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_value_and_volume_location;
  simple_value                : field_value;
  location                    : curve_volume_element_location;
  coordinate_system           : OPTIONAL curve_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

simple_values: the value of the field variable.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

6.7.85 curve_2d_element_location_point_variable_value

A **curve_2d_element_location_point_variable_value** is a state of the model that specifies the value of a variable aggregated through the section of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_location_point_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
  basis                        : BOOLEAN;
  values_and_locations         : SET[1:?] OF
                                 curve_2d_element_value_and_location;
  variable                     : curve_element_variable;
WHERE
  WR1: consistent_set_values (values_and_locations, variable);
  WR2: appropriate_set_value_existence (values_and_locations,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

basis: if **basis** is TRUE then the value set contains only basis locations for the field variable within the element, and no others.

values_and_locations: the values and locations of the field variable.

variable: the type of field variable being specified.

Formal propositions:

WR1: each variable shall have a corresponding value of the proper type.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.86 curve_2d_whole_element_variable_value

A **curve_2d_whole_element_variable_value** is a state of the model that specifies the value of a variable which is obtained by aggregation over a curve 2D element. The value represented by this entity does not include a contribution from the aggregation of boundary variables.

EXPRESS specification:

```
*)
ENTITY curve_2d_whole_element_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL curve_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.87 curve_2d_element_constant_specified_variable_value

A **curve_2d_element_constant_specified_variable_value** is a state of the model that specifies the value of a variable which is constant along the length of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_constant_specified_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : surface_element_variable;
  coordinate_system           : OPTIONAL curve_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.88 curve_2d_element_constant_specified_volume_variable_-value

A **curve_2d_element_constant_specified_volume_variable_value** is a state of the model that specifies the value of a variable at a point within a volume, which is constant throughout the volume of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_element_constant_specified_volume_variable_value
  SUBTYPE OF (curve_2d_element_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_variable;
  coordinate_system           : OPTIONAL curve_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.89 field_variable_element_group_value

A **field_variable_element_group_value** is a state of the model that specifies the value of a variable aggregated over a group of elements.

The value represented by this entity includes a contribution from variables aggregated within the elements and over their boundaries.

EXPRESS specification:

```
*)
ENTITY field_variable_element_group_value
  SUBTYPE OF (field_variable_definition);
  group                        : element_group;
  simple_value                 : field_value;
  variable                     : volume_aggregated_variable;
  coordinate_system            : OPTIONAL fea_axis2_placement_3d;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

group: the element group for which the values are specified, or the element groups for which values are to be calculated.

simple_value: the value of the field variable at the aggregate centroid of the element group.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.90 field_variable_whole_model_value

A **field_variable_whole_model_value** is a state of the model that specifies the value of a variable aggregated over all the elements in the model.

The value represented by this entity includes a contribution from variables aggregated within the elements and over their boundaries.

EXPRESS specification:

```
*)
ENTITY field_variable_whole_model_value
  SUBTYPE OF (field_variable_definition);
  simple_value                 : field_value;
  variable                     : volume_aggregated_variable;
```

```
  coordinate_system            : OPTIONAL fea_axis2_placement_3d;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable at the centroid of the model.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.91 field_variable_node_definition

A **field_variable_node_definition** is a state of the model that specifies the values of a field variable at nodes to represent nodal averaged values.

EXPRESS specification:

```
*)
ENTITY field_variable_node_definition
  SUPERTYPE OF (ONEOF (volume_3d_node_field_variable_definition,
                       volume_2d_node_field_variable_definition,
                       surface_3d_node_field_variable_definition,
                       surface_2d_node_field_variable_definition,
                       curve_3d_node_field_variable_definition,
                       curve_2d_node_field_variable_definition))
  SUBTYPE OF (field_variable_definition);
  node                         : node_output_reference;
  group                        : OPTIONAL element_group;
END_ENTITY;
(*
```

Attribute definitions:

node: the node for which the value is specified, or the node for which the value is to be calculated.

group: the elements connected to the node for which the field variable value is valid. If this attribute is omitted then the variable is valid for all elements connected to the node.

6.7.92 volume_3d_node_field_variable_definition

A **volume_3d_node_field_variable_definition** is a state of the model that specifies the values of a variable at the nodes of a volume 3D element.

EXPRESS specification:

```
*)
ENTITY volume_3d_node_field_variable_definition
  SUBTYPE OF (field_variable_node_definition);
  simple_value                : field_value;
  variable                    : volume_variable;
  coordinate_system           : OPTIONAL volume_3d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.93 volume_2d_node_field_variable_definition

A **volume_2d_node_field_variable_definition** is a state of the model that specifies the values of a variable at the nodes of a volume 2D element.

EXPRESS specification:

```
*)
ENTITY volume_2d_node_field_variable_definition
  SUBTYPE OF (field_variable_node_definition);
  simple_value                : field_value;
  variable                    : volume_variable;
  coordinate_system           : OPTIONAL volume_2d_element_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
```

```
      TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.94 surface_3d_node_field_variable_definition

A **surface_3d_node_field_variable_definition** is a state of the model that specifies the values of a variable at the nodes of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (
            surface_3d_node_field_section_variable_values,
            surface_3d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY;
(*
```

6.7.95 surface_3d_node_field_section_variable_values

A **surface_3d_node_field_section_variable_values** is a state of the model that specifies the values of a variable through the section at the nodes of a surface 3D element.

EXPRESS specification:

```
*)
ENTITY surface_3d_node_field_section_variable_values
  SUBTYPE OF (surface_3d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : surface_element_variable;
  location                    : surface_section_element_location;
  coordinate_system           : OPTIONAL surface_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
```

```
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.96 surface_3d_node_field_aggregated_variable_values

A **surface_3d_node_field_aggregated_variable_values** is a state of the model that specifies the values of a variable at the nodes of a surface 3D element. The variables are obtained by aggregation through the section at each node.

EXPRESS specification:

```
*)
ENTITY surface_3d_node_field_aggregated_variable_values
  SUBTYPE OF (surface_3d_node_field_variable_definition);
  simple_value                 : field_value;
  variable                     : volume_aggregated_variable;
  coordinate_system            : OPTIONAL surface_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.97 surface_2d_node_field_variable_definition

A **surface_2d_node_field_variable_definition** is a state of the model that specifies the values of a variable at the nodes of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (
            surface_2d_node_field_section_variable_values,
            surface_2d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY;
(*
```

6.7.98 surface_2d_node_field_section_variable_values

A **surface_2d_node_field_section_variable_values** is a state of the model that specifies the values of a variable through the section at the nodes of a surface 2D element.

EXPRESS specification:

```
*)
ENTITY surface_2d_node_field_section_variable_values
  SUBTYPE OF (surface_2d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : surface_element_variable;
  location                    : surface_section_element_location;
  coordinate_system           : OPTIONAL surface_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.99 surface_2d_node_field_aggregated_variable_values

A **surface_2d_node_field_aggregated_variable_values** is a state of the model that specifies the values of a variable at the nodes of a surface 2D element. The variables are obtained by aggregation through the section at each node.

EXPRESS specification:

```
*)
ENTITY surface_2d_node_field_aggregated_variable_values
  SUBTYPE OF (surface_2d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL surface_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.100 curve_3d_node_field_variable_definition

A **curve_3d_node_field_variable_definition** is a state of the model that specifies the values of a variable at the nodes of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (
            curve_3d_node_field_section_variable_values,
            curve_3d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY;
(*
```

6.7.101 curve_3d_node_field_section_variable_values

A **curve_3d_node_field_section_variable_values** is a state of the model that specifies the values of a variable through the section at the nodes of a curve 3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_node_field_section_variable_values
  SUBTYPE OF (curve_3d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : curve_element_variable;
  location                    : curve_section_element_location;
  coordinate_system           : OPTIONAL curve_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.102 curve_3d_node_field_aggregated_variable_values

A **curve_3d_node_field_aggregated_variable_values** is a state of the model that specifies the values of a variable aggregated through the cross-section of the curve at the nodes of a curve

3D element.

EXPRESS specification:

```
*)
ENTITY curve_3d_node_field_aggregated_variable_values
  SUBTYPE OF (curve_3d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL curve_3d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.103 curve_2d_node_field_variable_definition

A **curve_2d_node_field_variable_definition** is a state of the model that specifies the values of a variable at the nodes of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_node_field_variable_definition
  SUPERTYPE OF (ONEOF (
            curve_2d_node_field_section_variable_values,
            curve_2d_node_field_aggregated_variable_values))
  SUBTYPE OF (field_variable_node_definition);
END_ENTITY;
(*
```

6.7.104 curve_2d_node_field_section_variable_values

A **curve_2d_node_field_section_variable_values** is a state of the model that specifies the values of a variable across the section at the nodes of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_node_field_section_variable_values
  SUBTYPE OF (curve_2d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : curve_element_variable;
  location                    : curve_section_element_location;
  coordinate_system           : OPTIONAL curve_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

location: the location at which the value of the field variable is specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.105 curve_2d_node_field_aggregated_variable_values

A **curve_2d_node_field_aggregated_variable_values** is a state of the model that specifies the values of a variable aggregated through the cross-section at the nodes of a curve 2D element.

EXPRESS specification:

```
*)
ENTITY curve_2d_node_field_aggregated_variable_values
  SUBTYPE OF (curve_2d_node_field_variable_definition);
  simple_value                : field_value;
  variable                    : volume_aggregated_variable;
  coordinate_system           : OPTIONAL curve_2d_state_coordinate_system;
WHERE
  WR1: necessary_value_coordinate_system (simple_value, coordinate_system);
  WR2: consistent_value (simple_value, variable);
  WR3: appropriate_value_existence (simple_value,
       TYPEOF (SELF\state_definition.defined_state));
```

```
END_ENTITY;
(*
```

Attribute definitions:

simple_value: the value of the field variable.

variable: the type of field variable being specified.

coordinate_system: the coordinate system for the value.

Formal propositions:

WR1: the coordinate system shall be specified if any of the specified values are not scalar.

WR2: each variable shall have a corresponding value of the proper type.

WR3: the value shall exist only when the **state_definition** is a **state_definition_with_value**.

6.7.106 nodal_freedom_and_value_definition

A **nodal_freedom_and_value_definition** is a state of the model that specifies information at a node of the model with respect to the solution degrees of freedom that are not a part of an element field discretization. The information may be:

- Values for the solution degrees of freedom, such as x translation, or z rotation;

- Loads applied in the direction of the solutions degrees of freedom such as values for x force, or z moment. These may be loads applied to a node by elements of the model, or loads applied to a node from outside the model.

  NOTE – The purpose of this entity is NOT to represent an element displacement field that has been discretized at the model nodes. The **field_variable_node_definition** should be used for that purpose.

  The purpose of this entity IS to represent information such as loads and displacements at nodes.

EXPRESS specification:

```
*)
ENTITY nodal_freedom_and_value_definition
SUPERTYPE OF (ONEOF (nodal_freedom_values,
                     nodal_freedom_action_definition))
SUBTYPE OF (state_definition);
  node                        : node_output_reference;
  coordinate_system           : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_values         : SET [1:?] OF freedom_and_value;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_values,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

node: the node for which the values are specified, or the node for which the values are to be calculated.

coordinate_system: the coordinate system in which the nodal degrees of freedom are defined.

freedoms_and_values: the degree of freedom and the matching value.

Formal propositions:

WR1: a coordinate system shall be specified if any of the freedoms are not scalar.

6.7.107 nodal_freedom_values

A **nodal_freedom_values** is a state of the model that specifies the values for degrees of freedom at a node.

EXPRESS specification:

```
*)
ENTITY nodal_freedom_values
  SUBTYPE OF (nodal_freedom_and_value_definition);
END_ENTITY;
(*
```

6.7.108 nodal_freedom_action_definition

A **nodal_freedom_action_definition** is a state of the model that specifies the values for the actions applied directly to a node of the model. The type of action is indicated with the **action** attribute.

EXPRESS specification:

```
*)
ENTITY nodal_freedom_action_definition
  SUBTYPE OF (nodal_freedom_and_value_definition);
  action                        : action_type;
END_ENTITY;
(*
```

Attribute definitions:

action: what type of values (loads and moments) the action represents (such as external applied loads, or residual loads at the node).

6.7.109 element_nodal_freedom_actions

An **element_nodal_freedom_actions** is the values of nodal action for an element. This action is applied by the element to the node.

NOTE – This entity is used to represent the reactions of a single or multi-point constraint

EXPRESS specification:

```
*)
ENTITY element_nodal_freedom_actions
  SUBTYPE OF (state_definition);
  element                      : model_or_control_element;
  nodal_action                 : LIST [1:?] OF
                                 element_nodal_freedom_terms;
END_ENTITY;
(*
```

Attribute definitions:

element: the element for which the values are specified or calculated. This can be either an element or a constraint.

nodal_action: the values and associated degrees of freedom at the nodes of an element. The terms in the LIST correspond to the nodes in the **required_node_list** and the **additional_node_list** for the element.

6.7.110 element_nodal_freedom_terms

An **element_nodal_freedom_terms** is a list of degrees of freedom and corresponding values at a node of an element, and a reference coordinate system in which the values are defined.

EXPRESS specification:

```
*)
ENTITY element_nodal_freedom_terms;
  coordinate_system            : fea_axis2_placement_3d;
  freedoms_and_values          : SET [1:?] OF freedom_and_value;
END_ENTITY;
(*
```

Attribute definitions:

coordinate_system: the coordinate system which the values are defined with respect to.

freedoms_and_values: the freedom and value at the nodes of the element.

6.7.111 curve_freedom_and_value_definition

A **curve_freedom_and_value_definition** is a state of the model that specifies information at all nodes that may be associated with a curve with respect to the solution degrees of freedom that are not a part of an element field discretization. The intent of this entity is identical to a **nodal_freedom_and_value_definition**, , however it is applied to all nodes associated to a curve, rather than to a single node.

EXPRESS specification:

```
*)
ENTITY curve_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (curve_freedom_values,
                       curve_freedom_action_definition))
  SUBTYPE OF (state_definition);
```

```
  required_curve       : curve;
  coordinate_system    : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_values  : SET [1:?] OF freedom_and_value;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_values,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_curve: the curve whose associated nodes are for which the values are specified, or whose associated nodes for which the values are to be calculated.

coordinate_system: the coordinate system with respect to which the nodal degrees of freedom are defined.

freedoms_and_values: the freedom and value pairs.

Formal propositions:

WR1: a coordinate system shall be specified if any of the freedoms are not scalar.

6.7.112 curve_freedom_values

A **curve_freedom_values** is a state of the model that specifies the values for degrees of freedom at a node that is associated with a curve.

EXPRESS specification:

```
*)
ENTITY curve_freedom_values
  SUBTYPE OF (curve_freedom_and_value_definition);
END_ENTITY;
(*
```

6.7.113 curve_freedom_action_definition

A **curve_freedom_action_definition** is a state of the model that specifies the values for the actions applied directly to nodes associated with a curve. The type of action is indicated with the **action** attribute.

EXPRESS specification:

```
*)
ENTITY curve_freedom_action_definition
  SUBTYPE OF (curve_freedom_and_value_definition);
  action                       : action_type;
END_ENTITY;
(*
```

Attribute definitions:

action: what type of values (loads and moments) the action represents (such as external applied loads, or residual loads at the nodes associated with the curve).

6.7.114 point_freedom_and_value_definition

A **point_freedom_and_value_definition** is a state of the model that specifies information at all nodes that may be associated with a point with respect to the solution degrees of freedom that are not a part of an element field discretization. The intent of this entity is identical to a **nodal_freedom_and_value_definition**, however it is applied to all nodes associated to a point, rather than to a single node.

EXPRESS specification:

```
*)
ENTITY point_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (point_freedom_values,
                       point_freedom_action_definition))
  SUBTYPE OF (state_definition);
  required_point       : point;
  coordinate_system    : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_values  : SET [1:?] OF freedom_and_value;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_values,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_point: the point whose associated nodes are for which the values are specified, or whose associated nodes for which the values are to be calculated.

coordinate_system: the coordinate system with respect to which the nodal degrees of freedom are defined.

freedoms_and_values: the freedom and value pairs.

Formal propositions:

WR1: a coordinate system shall be specified if any of the freedoms are not scalar.

6.7.115 point_freedom_values

A **point_freedom_values** is a state of the model that specifies the values for degrees of freedom at a node that is associated with a point.

EXPRESS specification:

```
*)
ENTITY point_freedom_values
  SUBTYPE OF (point_freedom_and_value_definition);
END_ENTITY;
(*
```

6.7.116 point_freedom_action_definition

A **point_freedom_action_definition** is a state of the model that specifies the values for the actions applied directly to nodes associated with a point. The type of action is indicated with the **action** attribute.

EXPRESS specification:

```
*)
ENTITY point_freedom_action_definition
  SUBTYPE OF (point_freedom_and_value_definition);
  action                        : action_type;
END_ENTITY;
(*
```

Attribute definitions:

action: what type of values (loads and moments) the action represents (such as external applied loads, or residual loads at the nodes associated with the point).

6.7.117 surface_freedom_and_value_definition

A **surface_freedom_and_value_definition** is a state of the model that specifies information at all nodes that may be associated with a surface with respect to the solution degrees of freedom that are not a part of an element field discretization. The intent of this entity is identical to a **nodal_freedom_and_value_definition**, however it is applied to all nodes associated to a surface, rather than to a single node.

EXPRESS specification:

```
*)
ENTITY surface_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (surface_freedom_values,
                       surface_freedom_action_definition))
  SUBTYPE OF (state_definition);
  required_surface     : surface;
  coordinate_system    : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_values  : SET [1:?] OF freedom_and_value;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_values,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_surface: the surface whose associated nodes are for which the values are specified, or whose associated nodes for which the values are to be calculated.

coordinate_system: the coordinate system with respect to which the nodal degrees of freedom are defined.

freedoms_and_values: the freedom and value pairs.

Formal propositions:

WR1: a coordinate system shall be specified if any of the freedoms are not scalar.

6.7.118 surface_freedom_values

A **surface_freedom_values** is a state of the model that specifies the values for degrees of freedom at a node that is associated with a surface.

EXPRESS specification:

```
*)
ENTITY surface_freedom_values
  SUBTYPE OF (surface_freedom_and_value_definition);
END_ENTITY;
(*
```

6.7.119 surface_freedom_action_definition

A **surface_freedom_action_definition** is a state of the model that specifies the values for the actions applied directly to nodes associated with a surface. The type of action is indicated with the **action** attribute.

EXPRESS specification:

```
*)
ENTITY surface_freedom_action_definition
  SUBTYPE OF (surface_freedom_and_value_definition);
  action                        : action_type;
END_ENTITY;
(*
```

Attribute definitions:

action: what type of values (loads and moments) the action represents (such as external applied loads, or residual loads at the nodes associated with the surface).

6.7.120 solid_freedom_and_value_definition

A **solid_freedom_and_value_definition** is a state of the model that specifies information at all nodes that may be associated with a solid with respect to the solution degrees of freedom that are not a part of an element field discretization. The intent of this entity is identical to a **nodal_freedom_and_value_definition**, , however it is applied to all nodes associated to a solid, rather than to a single node.

EXPRESS specification:

```
*)
ENTITY solid_freedom_and_value_definition
  SUPERTYPE OF (ONEOF (solid_freedom_values,
                       solid_freedom_action_definition))
  SUBTYPE OF (state_definition);
  required_solid       : solid_model;
```

```
  coordinate_system    : OPTIONAL fea_axis2_placement_3d;
  freedoms_and_values  : SET [1:?] OF freedom_and_value;
WHERE
  WR1: necessary_freedom_coordinate_system_set (freedoms_and_values,
                                                coordinate_system);
END_ENTITY;
(*
```

Attribute definitions:

required_solid: the solid whose associated nodes are for which the values are specified, or whose associated nodes for which the values are to be calculated.

coordinate_system: the coordinate system with respect to which the nodal degrees of freedom are defined.

freedoms_and_values: the freedom and value pairs.

Formal propositions:

WR1: a coordinate system shall be specified if any of the freedoms are not scalar.

6.7.121 solid_freedom_values

A **solid_freedom_values** is a state of the model that specifies the values for degrees of freedom at a node that is associated with a solid.

EXPRESS specification:

```
*)
ENTITY solid_freedom_values
  SUBTYPE OF (solid_freedom_and_value_definition);
END_ENTITY;
(*
```

6.7.122 solid_freedom_action_definition

A **solid_freedom_action_definition** is a state of the model that specifies the values for the actions applied directly to nodes associated with a solid. The type of action is indicated with the **action** attribute.

EXPRESS specification:

```
*)
ENTITY solid_freedom_action_definition
  SUBTYPE OF (solid_freedom_and_value_definition);
  action                       : action_type;
END_ENTITY;
(*
```

Attribute definitions:

action: what type of values (loads and moments) the action represents (such as external applied loads, or residual loads at the nodes associated with the solid).

6.7.123 freedom_and_value

A **freedom_and_value** is the degree of freedom and the corresponding value of element nodal action at a node of an element.

The type of action is indicated by the degree of freedom. Hence for freedom **x_translation** the x force value is specified.

EXPRESS specification:

```
*)
ENTITY freedom_and_value;
  freedom                      : degree_of_freedom;
  simple_value                 : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

freedom: the degree of freedom associated with the value.

simple_value: the value of the nodal action for the degree of freedom.

6.7.124 linear_constraint_equation_element_value

A **linear_constraint_equation_element_value** is the specified constraint equation coefficient b associated with the freedoms (x_i) and coefficients (a_i) as shown in the equation in 6.4.11.

EXPRESS specification:

```
*)
ENTITY linear_constraint_equation_element_value
  SUBTYPE OF (state_definition);
  element                      : linear_constraint_equation_element;
  b                            : measure_or_unspecified_value;
END_ENTITY;
(*
```

Attribute definitions:

element: the linear constraint equation element to which the coefficient is applied.

b: the constraint equation coefficient b as above.

6.7.125 single_point_constraint_element_values

A **single_point_constraint_element_values** is the set of specified constraint equation coefficients and corresponding freedoms for a single point constraint.

NOTE – The freedoms are specified with the values to allow the correlation of the constraint coefficient with the nodal freedom coefficient in a **single_point_constraint_element** entity.

EXPRESS specification:

```
*)
ENTITY single_point_constraint_element_values
  SUBTYPE OF (state_definition);
  element                      : single_point_constraint_element;
  b                            : SET[1:?] OF freedom_and_value;
END_ENTITY;
(*
```

Attribute definitions:

element: the single point constraint element to which the coefficients and freedoms are applied.

b: the constraint equation coefficients and freedoms.

6.7.126 analysis_message

An **analysis_message** is message information that may be attached to an element, a node, or an entire model.

EXPRESS specification:

```
*)
ENTITY analysis_message
  SUPERTYPE OF (ONEOF (whole_model_analysis_message,
                       element_analysis_message,
                       node_analysis_message,
                       element_group_analysis_message))
  SUBTYPE OF (state_definition);
  message_level                : message_level;
  message_text                 : text;
END_ENTITY;
(*
```

Attribute definitions:

message_level: the severity of analysis message.

message_text: the message text.

6.7.127 whole_model_analysis_message

A **whole_model_analysis_message** is message information that may be attached to the entire **fea_model**.

EXPRESS specification:

```
*)
ENTITY whole_model_analysis_message
  SUBTYPE OF (analysis_message);
END_ENTITY;
(*
```

6.7.128 element_analysis_message

An **element_analysis_message** is message information that may be attached to an element.

EXPRESS specification:

```
*)
ENTITY element_analysis_message
  SUBTYPE OF (analysis_message);
  element                      : element_representation;
END_ENTITY;
(*
```

Attribute definitions:

element: the element that the message concerns.

6.7.129 node_analysis_message

A **node_analysis_message** is message information that may be attached to a node.

EXPRESS specification:

```
*)
ENTITY node_analysis_message
  SUBTYPE OF (analysis_message);
  node                         : node_output_reference;
END_ENTITY;
(*
```

Attribute definitions:

node: the node that the message concerns.

6.7.130 element_group_analysis_message

An **element_group_analysis_message** is message information that may be attached to an element group.

EXPRESS specification:

```
*)
ENTITY element_group_analysis_message
  SUBTYPE OF (analysis_message);
  group                        : element_group;
END_ENTITY;
(*
```

Attribute definitions:

group: the group of elements that the message concerns.

6.8 Finite element analysis control and result schema function definitions

These definitions are used to ensure the correctness of information in many of the finite element analysis control and result schema information model entities.

6.8.1 necessary_freedom_coordinate_system

A **necessary_freedom_coordinate_system** ensures that a coordinate system is specified for a freedom which contains non-scalar quantity.

EXPRESS specification:

```
*)
FUNCTION necessary_freedom_coordinate_system
    (freedoms          : GENERIC;
     coordinate_system : GENERIC): BOOLEAN;

    CONSTANT
     feacr : STRING := 'FINITE_ELEMENT_ANALYSIS_CONTROL_AND_RESULT_SCHEMA.';
    END_CONSTANT;

    LOCAL
     required          : BOOLEAN;
     status            : BOOLEAN;
     i                 : INTEGER;
    END_LOCAL;

    IF (NOT (SIZEOF (TYPEOF(freedoms) *
                    [feacr+'FREEDOMS_AND_VALUES',
                     feacr+'FREEDOMS_AND_COEFFICIENTS']) = 0)) THEN
      status := FALSE;
      RETURN (status);
    END_IF;

    required := FALSE;

    CASE freedoms.freedom OF
      'X_TRANSLATION',
      'Y_TRANSLATION',
      'Z_TRANSLATION',
      'X_ROTATION',
      'Y_ROTATION',
      'Z_ROTATION',
      'WARP' :
        required := TRUE;
    END_CASE;

    IF (required) THEN
        IF (NOT EXISTS (coordinate_system) ) THEN
            status := FALSE;
          ELSE
```

```
            status := TRUE;
        END_IF;
    END_IF;

    RETURN (status);

END_FUNCTION;
(*
```

Argument definitions:

freedoms: input argument that is the degrees of freedom being checked.

coordinate_system: input argument that is the coordinate system that is being checked for existence.

6.8.2 necessary_freedom_coordinate_system_set

A **necessary_freedom_coordinate_system_set** ensures that a coordinate system is specified for a set of freedoms which contains non-scalar quantities.

EXPRESS specification:

```
*)
FUNCTION necessary_freedom_coordinate_system_set
    (freedoms          : SET [1 : ?] OF GENERIC;
     coordinate_system : GENERIC): BOOLEAN;

    CONSTANT
     feacr : STRING := 'FINITE_ELEMENT_ANALYSIS_CONTROL_AND_RESULT_SCHEMA.';
    END_CONSTANT;

    LOCAL
     required          : BOOLEAN;
     status            : BOOLEAN;
     i                 : INTEGER;
    END_LOCAL;

    IF (NOT (SIZEOF (TYPEOF(freedoms) *
                     [feacr+'FREEDOMS_AND_VALUES',
                      feacr+'FREEDOMS_AND_COEFFICIENTS']) = 0)) THEN
      status := FALSE;
      RETURN (status);
    END_IF;

    required := FALSE;
    REPEAT i := 1 TO SIZEOF (freedoms);
        CASE freedoms[i].freedom OF
          'X_TRANSLATION',
  'Y_TRANSLATION',
  'Z_TRANSLATION',
  'X_ROTATION',
  'Y_ROTATION',
```

```
  'Z_ROTATION',
  'WARP' :
             required := TRUE;
        END_CASE;
    END_REPEAT;

    IF (required) THEN
        IF (NOT EXISTS (coordinate_system) ) THEN
            status := FALSE;
          ELSE
            status := TRUE;
        END_IF;
    END_IF;

    RETURN (status);

END_FUNCTION;
(*
```

Argument definitions:

freedoms: input argument that is the degrees of freedom being checked.

coordinate_system: input argument that is the coordinate system that is being checked for existence.

6.8.3 necessary_value_coordinate_system

A **necessary_value_coordinate_system** ensures that a coordinate system is supplied for the value of a non-scalar variable.

EXPRESS specification:

```
*)
FUNCTION necessary_value_coordinate_system
    (cs_value          : field_value;
     coordinate_system : GENERIC): BOOLEAN;

    LOCAL
     required          : BOOLEAN;
     status            : BOOLEAN;
    END_LOCAL;

    required := FALSE;

    IF 'FEA_SCALAR_VECTOR_TENSOR_SCHEMA.VECTOR_2D' IN
    TYPEOF (cs_value) THEN
        required := TRUE;
    END_IF;
    IF 'FEA_SCALAR_VECTOR_TENSOR_SCHEMA.VECTOR_3D' IN
    TYPEOF (cs_value) THEN
        required := TRUE;
    END_IF;
```

```
    IF 'FEA_SCALAR_VECTOR_TENSOR_SCHEMA.TENSOR2_2D' IN
    TYPEOF (cs_value) THEN
        required := TRUE;
    END_IF;
    IF 'FEA_SCALAR_VECTOR_TENSOR_SCHEMA.TENSOR2_3D' IN
    TYPEOF (cs_value) THEN
        required := TRUE;
    END_IF;

    IF (required) THEN
        IF (NOT EXISTS (coordinate_system) ) THEN
            status := FALSE;
        ELSE
          status := TRUE;
        END_IF;
    END_IF;

    RETURN (status);

END_FUNCTION;
(*
```

Argument definitions:

value: input argument that is the value being checked.

coordinate_system: input argument that is the coordinate system that is being checked for existence.

6.8.4 consistent_set_values

A **consistent_set_values** ensures the SET of variables and values are consistent.

EXPRESS specification:

```
*)
FUNCTION consistent_set_values
    (values_and_locations  : SET [1:?] OF GENERIC;
     variable              : GENERIC) : BOOLEAN;

     LOCAL
     vv_type : STRING;
     i    : INTEGER;
    END_LOCAL;

    vv_type := variable_value_type (variable);

    REPEAT i := 1 TO HIINDEX (values_and_locations);
       IF NOT (vv_type IN TYPEOF (values_and_locations[i].simple_value)) THEN
          RETURN (FALSE);
        END_IF;
    END_REPEAT;
```

```
    RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

values_and_locations: input argument that is the SET of values to be compared with the variable to ensure consistency.

variable: input argument that is the variable to be compared with the SET of values to ensure a consistency.

6.8.5 consistent_list_values

A **consistent_list_values** ensures the LIST of variables and values are consistent.

EXPRESS specification:

```
*)
FUNCTION consistent_list_values
    (values_and_locations  : LIST [1:?] OF GENERIC;
     variable              : GENERIC) : BOOLEAN;

     LOCAL
     vv_type : STRING;
     i    : INTEGER;
    END_LOCAL;

    vv_type := variable_value_type (variable);

    REPEAT i := 1 TO HIINDEX (values_and_locations);
       IF NOT (vv_type IN TYPEOF (values_and_locations[i].simple_value)) THEN
          RETURN (FALSE);
        END_IF;
    END_REPEAT;

    RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

values_and_locations: input argument that is the LIST of values to be compared with the variable to ensure consistency.

variable: input argument that is the variable to be compared with the LIST of values to ensure a consistency.

6.8.6 consistent_value

A **consistent_value** ensures the variable and value are consistent.

EXPRESS specification:

```
*)
FUNCTION consistent_value
    (c_value              : GENERIC;
     variable             : GENERIC) : BOOLEAN;

    LOCAL
     vv_type : STRING;
    END_LOCAL;

    vv_type := variable_value_type (variable);

    IF vv_type IN TYPEOF (c_value) THEN
       RETURN (FALSE);
    END_IF;

    RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

value: input argument that is the value to be compared with the variable to ensure consistency.

variable: input argument that is the variable to be compared with the value to ensure a consistency.

6.8.7 variable_value_type

A **variable_value_type** calculates the corresponding value for a supplied variable.

EXPRESS specification:

```
*)
FUNCTION variable_value_type

    (variable             : GENERIC) : STRING;

    CONSTANT
     svt   : STRING := 'FEA_SCALAR_VECTOR_TENSOR_SCHEMA.';
     feacr : STRING := 'FINITE_ELEMENT_ANALYSIS_CONTROL_AND_RESULT_SCHEMA.';
    END_CONSTANT;

    LOCAL
     vv_type : STRING;
     variable_typeof : SET[1:?] OF STRING;
    END_LOCAL;

    vv_type:='NO_MATCH';
    variable_typeof := TYPEOF (variable);
```

```
IF (feacr+'CURVE_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'SURFACE_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'VOLUME_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'BOUNDARY_CURVE_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'BOUNDARY_SURFACE_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'AGGREGATED_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'VOLUME_ANGULAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'AGGREGATED_ANGULAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'USER_DEFINED_SCALAR_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.SCALAR';
END_IF;

IF (feacr+'CURVE_VECTOR_2D_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.VECTOR_2D';
END_IF;

IF (feacr+'SURFACE_VECTOR_2D_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.VECTOR_2D';
END_IF;

IF (feacr+'USER_DEFINED_VECTOR_2D_VARIABLE' IN variable_typeof)
THEN
      vv_type := svt+'.VECTOR_2D';
END_IF;

IF (feacr+'CURVE_VECTOR_3D_VARIABLE' IN variable_typeof) THEN
      vv_type := svt+'.VECTOR_3D';
END_IF;
```

```
    IF (feacr+'SURFACE_VECTOR_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.VECTOR_3D';
    END_IF;

    IF (feacr+'VOLUME_VECTOR_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.VECTOR_3D';
    END_IF;

    IF (feacr+'BOUNDARY_CURVE_VECTOR_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.VECTOR_3D';
    END_IF;

    IF (feacr+'BOUNDARY_SURFACE_VECTOR_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.VECTOR_3D';
    END_IF;

    IF (feacr+'AGGREGATED_VECTOR_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.VECTOR_3D';
    END_IF;

    IF (feacr+'USER_DEFINED_VECTOR_3D_VARIABLE' IN variable_typeof)
    THEN
          vv_type := svt+'.VECTOR_3D';
    END_IF;

    IF (feacr+'SURFACE_TENSOR2_2D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.TENSOR2_2D';
    END_IF;

    IF (feacr+'USER_DEFINED_TENSOR2_2D_VARIABLE' IN variable_typeof)
    THEN
          vv_type := svt+'.TENSOR2_2D';
    END_IF;

    IF (feacr+'VOLUME_TENSOR2_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.TENSOR2_3D';
    END_IF;

    IF (feacr+'AGGREGATED_TENSOR2_3D_VARIABLE' IN variable_typeof) THEN
          vv_type := svt+'.TENSOR2_3D';
    END_IF;

    IF (feacr+'USER_DEFINED_TENSOR2_3D_VARIABLE' IN variable_typeof)
    THEN
          vv_type := svt+'.TENSOR2_3D';
    END_IF;

    RETURN (vv_type);

END_FUNCTION;
(*
```

Argument definitions:

variable: input argument that is used to calculate the appropriately corresponding type of value.

6.8.8 appropriate_set_value_existence

A **appropriate_set_value_existence** ensures the value is UNSPECIFIED only when the **state_definition** is a **state_definition_without_value** for a SET of **values_and_locations**.

EXPRESS specification:

```
*)
FUNCTION appropriate_set_value_existence
    (values_and_locations  : SET [1:?] OF GENERIC;
     type_self             : SET [1:?] OF STRING) : BOOLEAN;

    LOCAL
     i    : INTEGER;
    END_LOCAL;

    REPEAT i := 1 TO HIINDEX (values_and_locations);
       IF (appropriate_value_existence (values_and_locations[i].simple_value,
          type_self)) THEN
          RETURN (FALSE);
       END_IF;
    END_REPEAT;

    RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

values_and_locations: input argument that is the SET of values to be checked for existence.

type_self: input argument that is the **state_definition**.

6.8.9 appropriate_list_value_existence

A **appropriate_set_value_existence** ensures the value is UNSPECIFIED only when the **state_definition** is a **state_definition_without_value** for a LIST of **values_and_locations**.

EXPRESS specification:

```
*)
FUNCTION appropriate_list_value_existence
    (values                : LIST [1:?] OF GENERIC;
     type_self             : SET [1:?] OF STRING) : BOOLEAN;

    LOCAL
     i    : INTEGER;
    END_LOCAL;
```

```
    REPEAT i := 1 TO HIINDEX (values);
       IF (appropriate_value_existence (values[i], type_self)) THEN
          RETURN (FALSE);
       END_IF;
    END_REPEAT;

    RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

values_and_locations: input argument that is the LIST of values to be checked for existence.

type_self: input argument that is the **state_definition**.

6.8.10 appropriate_value_existence

An **appropriate_value_existence** ensures the value is UNSPECIFIED only when the **state_definition** is a **state_definition_with_value**.

EXPRESS specification:

```
*)
FUNCTION appropriate_value_existence
    (a_value_e             : GENERIC;
     type_self             : SET [1:?] OF STRING) : BOOLEAN;

    LOCAL
     feacr : STRING;
    END_LOCAL;

    feacr := 'FINITE_ELEMENT_ANALYSIS_CONTROL_AND_RESULT_SCHEMA.';

    IF (feacr+'OUTPUT_REQUEST_STATE' IN (type_self)) THEN
       RETURN (FALSE);
    END_IF;

    IF (feacr+'OUTPUT_REQUEST_STATE' IN (type_self)) AND
       ('UNSPECIFIED' IN TYPEOF (a_value_e)) THEN
       RETURN (FALSE);
    END_IF;

    RETURN (TRUE);

END_FUNCTION;
(*
```

Argument definitions:

values: input argument that is the value to be checked for proper type.

type_self: input argument that is the **state_definition**.

EXPRESS specification:

```
*)
END_SCHEMA; -- finite_element_analysis_control_and_result_schema
(*
```

7 FEA scalar vector tensor schema

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

EXPRESS specification:

```
*)
SCHEMA fea_scalar_vector_tensor_schema;

  REFERENCE FROM measure_schema
      (context_dependent_measure);
(*
```

NOTE – The schema referenced above can be found in the following Part of ISO 10303:

measure_schema	ISO 10303-41

7.1 Introduction

The subject of the **fea_scalar_vector_tensor_schema** is the definition of the scalars, vectors (first order tensors) and tensors that are necessary to represent the input and output of finite element analyses.

7.2 Fundamental concepts and assumptions

The fundamental concepts and assumptions include the concept that units are assigned in a context, and the value of the measure is assigned directly.

7.3 Scalar vector tensor schema type definitions: The FEA representation of scalars, vectors, and tensors

Finite element analysis requires scalars, two and three dimensional vectors, and two and three dimensional second and fourth order tensors to represent the input and output of finite element analyses. The three dimensional fourth order tensors are used for material response, and are documented in 7.4.

7.3.1 angular_value

A **angular_value** is a zero order tensor representing the magnitude of an angular value about an axis. The axis shall be the Z axis of the founding placement of the referencing entity.

EXPRESS specification:

```
*)
TYPE angular_value =
     context_dependent_measure;
```

```
END_TYPE;
(*
```

7.3.2 scalar

A **scalar** is a zero order tensor.

EXPRESS specification:

```
*)
TYPE scalar =
     context_dependent_measure;
END_TYPE;
(*
```

7.3.3 tensor1

A **tensor1** is a first order tensor, or vector.

EXPRESS specification:

```
*)
TYPE tensor1 = SELECT
  (tensor1_2d,
   tensor1_3d);
END_TYPE;
(*
```

7.3.4 tensor1_2d

A **tensor1_2d** is a first order tensor, or vector in two dimensional space.

EXPRESS specification:

```
*)
TYPE tensor1_2d =
     ARRAY [1:2] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the 1 component of the tensor.

ARRAY[2]: the 2 component of the tensor.

7.3.5 tensor1_3d

A **tensor1_3d** is a first order tensor, or vector, in three dimensional space.

EXPRESS specification:

```
*)
TYPE tensor1_3d =
  ARRAY [1:3] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the 1 component of the tensor.

ARRAY[2]: the 2 component of the tensor.

ARRAY[3]: the 3 component of the tensor.

7.3.6 symmetric_tensor2_2d

A **symmetric_tensor2_2d** is a symmetric second order tensor in two dimensional space.

A 2D second order tensor s_{ij} is the relationship between two 2D vectors f_i and n_i, as shown below:

$$f_i = \sum_{j=1}^{2} s_{ij} n_j \quad \text{for} \quad i = 1, 2$$

EXPRESS specification:

```
*)
TYPE symmetric_tensor2_2d = SELECT
  (anisotropic_symmetric_tensor2_2d);
END_TYPE;
(*
```

7.3.7 anisotropic_symmetric_tensor2_2d

An **anisotropic_symmetric_tensor2_2d** is an anisotropic symmetric second order tensor in two dimensional space.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor2_2d =
  ARRAY [1:3] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the 11 component of the tensor.

ARRAY[2]: the 22 component of the tensor.

ARRAY[3]: the 12 component of the tensor.

7.3.8 symmetric_tensor2_3d

A **symmetric_tensor2_3d** is a symmetric second order tensor in three dimensional space.

A 3D second order tensor s_{ij} is the relationship between two 3D vectors f_i and n_i, as shown below:

$$f_i = \sum_{j=1}^{3} s_{ij} n_j \quad \text{for} \quad i = 1, 3$$

EXPRESS specification:

```
*)
TYPE symmetric_tensor2_3d = SELECT
  (isotropic_symmetric_tensor2_3d,
   orthotropic_symmetric_tensor2_3d,
   anisotropic_symmetric_tensor2_3d);
END_TYPE;
(*
```

7.3.9 isotropic_symmetric_tensor2_3d

An **isotropic_symmetric_tensor2_3d** is an isotropic symmetric second order tensor in three dimensional space, which is characterized by one independent value.

The value s gives terms in the tensor as follows:

$$\begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & s \end{pmatrix}$$

EXPRESS specification:

```
*)
TYPE isotropic_symmetric_tensor2_3d =
     context_dependent_measure;
END_TYPE;
(*
```

7.3.10 orthotropic_symmetric_tensor2_3d

An **orthotropic_symmetric_tensor2_3d** is an orthotropic symmetric second order tensor in three dimensional space, which is characterized by three independent values.

The values s_{11}, s_{22}, and s_{33} give terms in the tensor as follows:

$$\begin{pmatrix} s_{11} & 0 & 0 \\ 0 & s_{22} & 0 \\ 0 & 0 & s_{33} \end{pmatrix}$$

EXPRESS specification:

```
*)
TYPE orthotropic_symmetric_tensor2_3d =
     ARRAY [1:3] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the tensor components s_{11} defined above.

ARRAY[2]: the tensor components s_{22} defined above.

ARRAY[3]: the tensor components s_{33} defined above.

7.3.11 anisotropic_symmetric_tensor2_3d

An **anisotropic_symmetric_tensor2_3d** is an anisotropic symmetric second order tensor in three dimensional space.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor2_3d =
     ARRAY [1:6] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the 11 component of the tensor.

ARRAY[2]: the 12 component of the tensor.

ARRAY[3]: the 13 component of the tensor.

ARRAY[4]: the 22 component of the tensor.

ARRAY[5]: the 23 component of the tensor.

ARRAY[6]: the 33 component of the tensor.

7.3.12 symmetric_tensor4_2d

A **symmetric_tensor4_2d** is a symmetric fourth order tensor in two dimensional space.

A 2D fourth order tensor d_{ijkl} is the relationship between two 2D second order tensors s_{ij} and e_{ij}, as shown below:

$$s_{ij} = \sum_{k=1}^{2} \sum_{l=1}^{2} d_{ijkl} e_{kl} \text{ for } i = 1,2 \text{ and } j = 1,2$$

EXPRESS specification:

```
*)
TYPE symmetric_tensor4_2d = SELECT
  (anisotropic_symmetric_tensor4_2d);
END_TYPE;

(*
```

7.3.13 anisotropic_symmetric_tensor4_2d

An **anisotropic_symmetric_tensor4_2d** is an anisotropic symmetric fourth order tensor in two dimensional space.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor4_2d =
     ARRAY [1:6] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the 1111 component of the tensor.

ARRAY[2]: the 1122 component of the tensor.

ARRAY[3]: the 1112 component of the tensor.

ARRAY[4]: the 2222 component of the tensor.

ARRAY[5]: the 2212 component of the tensor.

ARRAY[6]: the 1212 component of the tensor.

7.4 Scalar vector tensor schema type definitions: The FEA representation of a fourth order material response tensor

In finite element analysis it is common to represent a symmetric 3D second order tensor s_{ij} as a (6) array S_i, with terms:

$$\begin{pmatrix} s_{11} \\ s_{22} \\ s_{33} \\ s_{12} \\ s_{23} \\ s_{31} \end{pmatrix}$$

Correspondingly, a symmetric 3D fourth order tensor d_{ijkl} is represented as a symmetric (6×6)

matrix D_{ij} with terms:

$$\begin{pmatrix} d_{1111} & d_{1122} & d_{1133} & d_{1112} & d_{1123} & d_{1131} \\ & d_{2222} & d_{2233} & d_{2212} & d_{2223} & d_{2231} \\ & & d_{3333} & d_{3312} & d_{3323} & d_{3331} \\ & & & d_{1212} & d_{1223} & d_{1231} \\ & symmetric & & & d_{2323} & d_{2331} \\ & & & & & d_{3131} \end{pmatrix}$$

Note that if:

- σ_{ij} is stress, with (6) array representation S_i;
- ε_{ij} is strain;
- d_{ijkl} is elasticity, with (6×6) matrix representation D_{ij}.

then:

$$\sigma_{ij} = \sum_{k=1}^{3} \sum_{l=1}^{3} d_{ijkl}(\varepsilon_{kl} - \alpha_{kl}\Delta T - \beta_{ij}\Delta M) \text{ for } i = 1,3 \text{ and } j = 1,3$$

and in contracted notation:

$$S_i = \sum_{j=1}^{6} D_{ij}(E_j - {E_j}^T - {E_j}^M) \text{ for } i = 1,6$$

where E_j is the **engineering** representation of the strain with components:

$$\begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{12} \\ 2\varepsilon_{23} \\ 2\varepsilon_{31} \end{pmatrix}$$

A second order tensor shall be represented by its true **mathematical** components, and not by the components of its **engineering** representation.

7.4.1 symmetric_tensor4_3d

A **symmetric_tensor4_3d** is a symmetric fourth order material response tensor in two dimensional space. A 3D fourth order tensor d_{ijkl} represents the relationship between two 3D second order tensors s_{ij} and e_{ij}, as shown below:

$$s_{ij} = \sum_{k=1}^{3} \sum_{l=1}^{3} d_{ijkl} e_{kl} \text{ for } i = 1,3 \text{ and } j = 1,3$$

EXPRESS specification:

```
*)

TYPE symmetric_tensor4_3d = SELECT
 (anisotropic_symmetric_tensor4_3d,
  fea_isotropic_symmetric_tensor4_3d,
  fea_iso_orthotropic_symmetric_tensor4_3d,
  fea_transverse_isotropic_symmetric_tensor4_3d,
  fea_column_normalised_orthotropic_symmetric_tensor4_3d,
  fea_column_normalised_monoclinic_symmetric_tensor4_3d);
END_TYPE;
(*
```

7.4.2 anisotropic_symmetric_tensor4_3d

An **anisotropic_symmetric_tensor4_3d** is an anisotropic symmetric fourth order tensor in two dimensional space.

EXPRESS specification:

```
*)
TYPE anisotropic_symmetric_tensor4_3d =
     ARRAY [1:21] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the 1111 component of the tensor.

ARRAY[2]: the 1122 component of the tensor.

ARRAY[3]: the 1133 component of the tensor.

ARRAY[4]: the 1112 component of the tensor.

ARRAY[5]: the 1123 component of the tensor.

ARRAY[6]: the 1131 component of the tensor.

ARRAY[7]: the 2222 component of the tensor.

ARRAY[8]: the 2233 component of the tensor.

ARRAY[9]: the 2212 component of the tensor.

ARRAY[10]: the 2223 component of the tensor.

ARRAY[11]: the 2231 component of the tensor.

ARRAY[12]: the 3333 component of the tensor.

ARRAY[13]: the 3312 component of the tensor.

ARRAY[14]: the 3323 component of the tensor.

ARRAY[15]: the 3331 component of the tensor.

ARRAY[16]: the 1212 component of the tensor.

ARRAY[17]: the 1223 component of the tensor.

ARRAY[18]: the 1231 component of the tensor.

ARRAY[19]: the 2323 component of the tensor.

ARRAY[20]: the 2331 component of the tensor.

ARRAY[21]: the 3131 component of the tensor.

7.4.3 fea_isotropic_symmetric_tensor4_3d

An **fea_isotropic_symmetric_tensor4_3d** is an isotropic symmetric fourth order tensor in three dimensional space, which is characterized by two independent values.

The values E and ν give terms in the (6×6) array representation of the **inverse** of the tensor as follows:

$$\begin{pmatrix} \frac{1}{E} & -\frac{\nu}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ & \frac{1}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ & & \frac{1}{E} & 0 & 0 & 0 \\ & & & \frac{1}{G} & 0 & 0 \\ & symmetric & & & \frac{1}{G} & 0 \\ & & & & & \frac{1}{G} \end{pmatrix}$$

where

$$G = \frac{E}{2(1+\nu)}$$

.

For an elasticity tensor:

- E is the Young's modulus;
- ν is the Poisson's ratio;
- G is the shear modulus;

– the terms are shown for the inverse of an elasticity tensor, which is a compliance tensor.

EXPRESS specification:

```
*)
TYPE fea_isotropic_symmetric_tensor4_3d =
     ARRAY [1:2] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the term E defined above, which is the Young's modulus for an elasticity tensor.

ARRAY[2]: the term ν defined above, which is the Poisson's ratio for an elasticity tensor.

7.4.4 fea_iso_orthotropic_symmetric_tensor4_3d

An **fea_iso_orthotropic_symmetric_tensor4_3d** is an iso-orthotropic symmetric fourth order tensor in three dimensional space, which is characterized by three independent values.

The values E, ν, and G give terms in the (6×6) array representation of the **inverse** of the tensor as follows:

$$\begin{pmatrix} \frac{1}{E} & -\frac{\nu}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ & \frac{1}{E} & -\frac{\nu}{E} & 0 & 0 & 0 \\ & & \frac{1}{E} & 0 & 0 & 0 \\ & & & \frac{1}{G} & 0 & 0 \\ & symmetric & & & \frac{1}{G} & 0 \\ & & & & & \frac{1}{G} \end{pmatrix}$$

For an elasticity tensor:

– E is the Young's modulus;

– ν is the Poisson's ratio;

– G is the shear modulus;

– the terms are shown for the inverse of an elasticity tensor, which is a compliance tensor.

EXPRESS specification:

```
*)
TYPE fea_iso_orthotropic_symmetric_tensor4_3d =
     ARRAY [1:3] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the term E defined above, which is the Young's modulus for an elasticity tensor.

ARRAY[2]: the term ν defined above, which is the Poisson's ratio for an elasticity tensor.

ARRAY[3]: the term G defined above, which is the shear modulus for an elasticity tensor.

7.4.5 fea_transverse_isotropic_symmetric_tensor4_3d

An **fea_transverse_isotropic_symmetric_tensor4_3d** is a transverse isotropic symmetric fourth order tensor in three dimensional space, which is characterized by five independent values.

The values E_{ll}, E_{tt}, ν_{tt}, ν_{tl}, and G_{lt} give terms in the (6×6) array representation of the **inverse** of the tensor as follows:

$$\begin{pmatrix} \frac{1}{E_{ll}} & -\frac{\nu_{tl}}{E_{tt}} & -\frac{\nu_{tl}}{E_{tt}} & 0 & 0 & 0 \\ & \frac{1}{E_{tt}} & -\frac{\nu_{tt}}{E_{tt}} & 0 & 0 & 0 \\ & & \frac{1}{E_{tt}} & 0 & 0 & 0 \\ & & & \frac{1}{G_{lt}} & 0 & 0 \\ & symmetric & & & \frac{1}{G_{tt}} & 0 \\ & & & & & \frac{1}{G_{lt}} \end{pmatrix}$$

where

$$G_{tt} = \frac{E_{tt}}{2(1+\nu_{tt})}$$

.

For an elasticity tensor:

- E_{ll} is the longitudinal Young's modulus;
- ν_{tt} is the Poisson's ratio for strains in the transverse direction when uniaxially stressed in a transverse direction;
- E_{tt} is the transverse Young's modulus;
- ν_{tl} is the ratio between strain in a transverse direction to strain in the longitudinal direction when uniaxially strained in the longitudinal direction;
- G_{lt} is the modulus for shear in the longitudinal - transverse plane;
- The 1 axis is longitudinal;
- The 2 and 3 axes are transverse;

– the terms are shown for the inverse of an elasticity tensor, which is a compliance tensor.

EXPRESS specification:

```
*)
TYPE fea_transverse_isotropic_symmetric_tensor4_3d =
     ARRAY [1:5] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the term E_{ll} defined above, which is the Young's modulus in the longitudinal direction for an elasticity tensor.

ARRAY[2]: the term ν_{tt} defined above, which is the Poisson's ratio in the transverse plane for an elasticity tensor.

ARRAY[2]: the term E_{tt} defined above, which is the Young's modulus in a transverse direction for an elasticity tensor.

ARRAY[4]: the term ν_{tl} defined above, which is the Poisson's ratio between transverse and longitudinal directions for an elasticity tensor.

ARRAY[5]: the term G_{lt} defined above, which is the modulus for shear between longitudinal and transverse directions for an elasticity tensor.

7.4.6 fea_column_normalised_orthotropic_symmetric_tensor4_-3d

An **fea_column_normalised_orthotropic_symmetric_tensor4_3d** is a column normalised orthotropic symmetric fourth order tensor in three dimensional space, which is characterized by nine independent values.

The values E_{11}, E_{22}, E_{33}, ν_{12}, ν_{23}, ν_{31}, G_{12}, G_{23}, and G_{31} give terms in the (6×6) array representation of the **inverse** of the tensor as follows:

$$\begin{pmatrix} \frac{1}{E_{11}} & -\frac{\nu_{12}}{E_{22}} & -\frac{\nu_{13}}{E_{33}} & 0 & 0 & 0 \\ & \frac{1}{E_{22}} & -\frac{\nu_{23}}{E_{33}} & 0 & 0 & 0 \\ & & \frac{1}{E_{33}} & 0 & 0 & 0 \\ & & & \frac{1}{G_{12}} & 0 & 0 \\ & symmetric & & & \frac{1}{G_{23}} & 0 \\ & & & & & \frac{1}{G_{31}} \end{pmatrix}$$

For an elasticity tensor:

- E_{11} is the Young's modulus in the 1 axis direction;
- E_{22} is the Young's modulus in the 2 axis direction;
- E_{33} is the Young's modulus in the 3 axis direction;
- ν_{12} is the Poisson's ratio for transverse strain in the 2 direction when uniaxially stressed in the 1 direction, $\nu_{12} = -\varepsilon_2/\varepsilon_1$ when $s_{ij} = 0$ for $i, j \neq 1$;
- ν_{13} is the Poisson's ratio for transverse strain in the 3 direction when uniaxially stressed in the 1 direction, $\nu_{13} = -\varepsilon_3/\varepsilon_1$ when $s_{ij} = 0$ for $i, j \neq 1$;
- ν_{23} is the Poisson's ratio for transverse strain in the 3 direction when uniaxially stressed in the 2 direction; $\nu_{23} = -\varepsilon_3/\varepsilon_2$ when $s_{ij} = 0$ for $i, j \neq 2$;
- G_{12} is the modulus for shear in the 1/2 plane;
- G_{23} is the modulus for shear in the 2/3 plane;
- G_{31} is the modulus for shear in the 3/1 plane;
- the terms are shown for the inverse of an elasticity tensor, which is a compliance tensor.

NOTE – The Poisson's ratio convention used here and symmetry results in $E_{ii}\nu_{ji} = E_{jj}\nu_{ij}$.

EXPRESS specification:

```
*)
TYPE fea_column_normalised_orthotropic_symmetric_tensor4_3d =
     ARRAY [1:9] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the term E_{11} defined above, which is the Young's modulus in the 1 direction for an elasticity tensor.

ARRAY[2]: the term E_{22} defined above, which is the Young's modulus in the 2 direction for an elasticity tensor.

ARRAY[3]: the term E_{33} defined above, which is the Young's modulus in the 3 direction for an elasticity tensor.

ARRAY[4]: the term ν_{12} defined above, which is the Poisson's ratio between the 1 and 2 directions for an elasticity tensor.

ARRAY[5]: the term ν_{13} defined above, which is the Poisson's ratio between the 1 and 3 directions for an elasticity tensor.

ARRAY[6]: the term ν_{23} defined above, which is the Poisson's ratio between the 2 and 3 directions for an elasticity tensor.

ARRAY[7]: the term G_{12} defined above, which is the modulus between the 2 and 3 directions for an elasticity tensor.

ARRAY[8]: the term G_{23} defined above, which is the modulus between the 3 and 1 directions for an elasticity tensor.

ARRAY[9]: the term G_{31} defined above, which is the modulus between the 1 and 2 directions for an elasticity tensor.

7.4.7 fea_column_normalised_monoclinic_symmetric_tensor4_-3d

An **fea_column_normalised_monoclinic_symmetric_tensor4_3d** is a column normalised monoclinic symmetric fourth order tensor in three dimensional space, which is characterized by thirteen independent values.

The values E_{11}, E_{22}, E_{33}, ν_{12}, ν_{13}, ν_{23}, G_{23}, G_{31}, G_{12}, $\nu_{1,12}$, $\nu_{2,12}$, $\nu_{3,12}$, and $\nu_{23,31}$ give terms in the (6×6) array representation of the **inverse** of the tensor as follows:

$$\begin{pmatrix} \frac{1}{E_{11}} & -\frac{\nu_{12}}{E_{22}} & -\frac{\nu_{13}}{E_{33}} & \frac{\nu_{1,12}}{G_{12}} & 0 & 0 \\ & \frac{1}{E_{22}} & -\frac{\nu_{23}}{E_{22}} & \frac{\nu_{2,12}}{G_{12}} & 0 & 0 \\ & & \frac{1}{E_{33}} & \frac{\nu_{3,12}}{G_{12}} & 0 & 0 \\ & & & \frac{1}{G_{12}} & 0 & 0 \\ & symmetric & & & \frac{1}{G_{23}} & \frac{\nu_{23,31}}{G_{31}} \\ & & & & & \frac{1}{G_{31}} \end{pmatrix}$$

For an elasticity tensor:

- E_{11} is the Young's modulus in the 1 axis direction;
- E_{22} is the Young's modulus in the 2 axis direction;
- E_{33} is the Young's modulus in the 3 axis direction;
- ν_{12} is the Poisson's ratio for transverse strain in the 2 direction when uniaxially stressed in the 1 direction, $E_{11}\nu_{21} = E_{22}\nu_{12}$;
- ν_{13} is the Poisson's ratio for transverse strain in the 3 direction when uniaxially stressed in the 1 direction, $E_{11}\nu_{31} = E_{33}\nu_{13}$;

- ν_{23} is the Poisson's ratio for transverse strain in the 3 direction when uniaxially stressed in the 2 direction, $E_{22}\nu_{32} = E_{33}\nu_{23}$;

- G_{23} is the modulus for shear in the 2/3 plane;

- G_{31} is the modulus for shear in the 3/1 plane;

- G_{12} is the modulus for shear in the 1/2 plane;

- $\nu_{1,12}$ is the coupling between the 1 axis extension and shear in the 1/2 plane, $E_{11}\nu_{1,12} = G_{12}\nu_{12,1}$;

- $\nu_{2,12}$ is the coupling between the 2 axis extension and shear in the 1/2 plane, $E_{22}\nu_{2,12} = G_{12}\nu_{12,3}$;

- $\nu_{3,12}$ is the coupling between the 3 axis extension and shear in the 1/2 plane, $E_{33}\nu_{3,12} = G_{12}\nu_{12,2}$;

- $\nu_{23,31}$ is the coupling between shears in the 2/3 and the 3/1 planes, $G_{23}\nu_{23,31} = G_{13}\nu_{31,23}$;

- the terms are shown for the inverse of an elasticity tensor, which is a compliance tensor.

EXPRESS specification:

```
*)
TYPE fea_column_normalised_monoclinic_symmetric_tensor4_3d =
     ARRAY [1:13] OF context_dependent_measure;
END_TYPE;
(*
```

Enumerated item definitions:

ARRAY[1]: the term E_{11} defined above, which is the Young's modulus in the 1 direction for an elasticity tensor.

ARRAY[2]: the term E_{22} defined above, which is the Young's modulus in the 2 direction for an elasticity tensor.

ARRAY[3]: the term E_{33} defined above, which is the Young's modulus in the 3 direction for an elasticity tensor.

ARRAY[4]: the term ν_{12} defined above, which is the Poisson's ratio between the 1 and 2 directions for an elasticity tensor.

ARRAY[5]: the term ν_{13} defined above, which is the Poisson's ratio between the 1 and 3 directions for an elasticity tensor.

ARRAY[6]: the term ν_{23} defined above, which is the Poisson's ratio between the 2 and 3 directions for an elasticity tensor.

ARRAY[7]: the term G_{23} defined above, which is the modulus between the 2 and 3 directions for an elasticity tensor.

ARRAY[8]: the term G_{31} defined above, which is the modulus between the 3 and 1 directions for an elasticity tensor.

ARRAY[9]: the term G_{12} defined above, which is the modulus between the 1 and 2 directions for an elasticity tensor.

ARRAY[10]: the term $\nu_{1,12}$ defined above, which is the coupling between 1 extension and 1/2 plane shear for an elasticity tensor.

ARRAY[11]: the term $\nu_{2,12}$ defined above, which is the coupling between 2 extension and 1/2 plane shear for an elasticity tensor.

ARRAY[12]: the term $\nu_{3,12}$ defined above, which is the coupling between 3 extension and 1/2 plane shear for an elasticity tensor.

ARRAY[13]: the term $\nu_{23,31}$ defined above, which is the coupling between 2/3 and 3/1 plane shear for an elasticity tensor.

EXPRESS specification:

```
*)
END_SCHEMA; -- scalar_vector_tensor_schema
(*
```

Annex A
(normative)

Short names of entities

Table A.1 provides the short names of entities specified in this Part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303. This annex is also provided in digital form.

Annex B
(informative)

EXPRESS listing

This annex provides a listing of the EXPRESS constructs specified in this Part of ISO 10303. No text or annotation is included. This annex is provided only in digital form.

Annex C
(informative)

Finite element multi-disciplinary and nonlinear analysis extensions

The purpose of this annex is to provide a top-level description of how the FEA information model will eventually support an integrated analysis environment. The scope of the FEA information model is multi-disciplinary, extending across all analysis disciplines necessary for design verification as illustrated in figure C.1. In addition this annex will describe how the FEA information model was designed to be easily adapted to nonliear analyses.

Figure C.1 – Multi-disciplinary analysis environment

Each of the disciplines has existing or potential FEA application, either uncoupled or coupled to other disciplines, with bounding interface information requirements as shown in figure 1. While Version 1.0 of the STEP FEA information model emphasises structural analysis applications, the architecture is sufficiently generic and open to accommodate other disciplines as required. The FEA information model will cover all the informational needs of the analyst and include

all the information pertaining to definition of the finite element models, controls and results. The model will accommodate multi-level substructuring within an analysis discipline as well as across disciplines.

Multi-levelled substructuring was considered in the design of the FEA information model by implementing an approach that "wraps around" the existing information model. This approach was selected for minimum impact on the existing model so that a usable portion of the FEA information model could be published without the prospect of a large scale revision to support substructuring

Within each analysis discipline a hierarchy of analysis types is planned. The goal is to provide as common an analysis architecture as practical. The results state definition packet concept forms the inter-disciplinary interface. Note that the interpolation of information to this common information structure is out of scope. As multi-disciplinary analysis is addressed this architecture will be expanded. Candidate structural analysis types to be supported by the FEA information model include:

- Analysis - Structural
    - Linear
        - Static Thermo-structural
        - Bifurcation Buckling
        - Transient Response
        - Frequency Response
        - Modes and Frequencies
    - Nonlinear
        - Static Thermo-structural
        - Buckling
        - Transient Response
        - Frequency Response
        - Creep

Other analysis disciplines and interactions to be considered include:

- Analysis - Heat Transfer
- Analysis - Fluid

- Analysis - Electromagnetic
- Analysis - Flexible Multi-body Kinematics
- Analysis - Optic
- Analysis - Others

The FEA information model has been intentionally designed to accomodate nonlinear analyses. The areas to be detailed to complete this expansion include the definition of nonlinear element types, material property representationa, and element output scalars, vectors, and tensors. The basic structure to represent the input and output of a nonlinear analysis, however, has been accounted for in the information model design. Each of these areas has been designed to be easily expanded without impacting the current information model.

One of the main aspects of nonlinear analyses is the use of output information from a previous iteration to serve as input for the next. This information representation capability has been included in the specification of analysis control and state definition. The extensions for nonlinear analyses to this area would need only be the delineation of specific types of nonlinear analysis steps and states. Because of this approach the majority of the information needed to exchange a nonlinear analysis is currently in place.

Annex D
(informative)

EXPRESS-G Diagrams

Figure D.1 – Structural response definition schema EXPRESS-G diagram, 1 of 85.

Figure D.2 – Structural response representation schema EXPRESS-G diagram, 2 of 85.

Figure D.3 – Structural response representation schema EXPRESS-G diagram, 3 of 85.

Figure D.4 – Structural response representation schema EXPRESS-G diagram, 4 of 85.

Figure D.5 – Structural response representation schema EXPRESS-G diagram, 5 of 85.

Figure D.6 – Structural response representation schema EXPRESS-G diagram, 6 of 85.

Figure D.7 – Structural response representation schema EXPRESS-G diagram, 7 of 85.

Figure D.8 – Structural response representation schema EXPRESS-G diagram, 8 of 85.

Figure D.9 – Structural response representation schema EXPRESS-G diagram, 9 of 85.

Figure D.10 – Structural response representation schema EXPRESS-G diagram, 10 of 85.

Figure D.11 – Structural response representation schema EXPRESS-G diagram, 11 of 85.

Figure D.12 – Structural response representation schema EXPRESS-G diagram, 12 of 85.

Figure D.13 – Structural response representation schema EXPRESS-G diagram, 13 of 85.

Figure D.14 – Structural response representation schema EXPRESS-G diagram, 14 of 85.

Figure D.15 – Structural response representation schema EXPRESS-G diagram, 15 of 85.

Figure D.16 – Structural response representation schema EXPRESS-G diagram, 16 of 85.

Figure D.17 – Structural response representation schema EXPRESS-G diagram, 17 of 85.

Figure D.18 – Structural response representation schema EXPRESS-G diagram, 18 of 85.

Figure D.19 – Structural response representation schema EXPRESS-G diagram, 19 of 85.

Figure D.20 – Structural response representation schema EXPRESS-G diagram, 20 of 85.

Figure D.21 – Structural response representation schema EXPRESS-G diagram, 21 of 85.

Figure D.22 – Structural response representation schema EXPRESS-G diagram, 22 of 85.

Figure D.23 – Structural response representation schema EXPRESS-G diagram, 23 of 85.

Figure D.24 – Structural response representation schema EXPRESS-G diagram, 24 of 85.

Figure D.25 – Structural response representation schema EXPRESS-G diagram, 25 of 85.

Figure D.26 – Finite element analysis and control schema EXPRESS-G diagram, 26 of 85.

Figure D.27 – Finite element analysis and control schema EXPRESS-G diagram, 27 of 85.

Figure D.28 – Finite element analysis and control schema EXPRESS-G diagram, 28 of 85.

Figure D.29 – Finite element analysis and control schema EXPRESS-G diagram, 29 of 85.

Figure D.30 – Finite element analysis and control schema EXPRESS-G diagram, 30 of 85.

Figure D.31 – Finite element analysis and control schema EXPRESS-G diagram, 31 of 85.

Figure D.32 – Finite element analysis and control schema EXPRESS-G diagram, 32 of 85.

Figure D.33 – Finite element analysis and control schema EXPRESS-G diagram, 33 of 85.

Figure D.34 – Finite element analysis and control schema EXPRESS-G diagram, 34 of 85.

Figure D.35 – Finite element analysis and control schema EXPRESS-G diagram, 35 of 85.

Figure D.36 – Finite element analysis and control schema EXPRESS-G diagram, 36 of 85.

Figure D.37 – Finite element analysis and control schema EXPRESS-G diagram, 37 of 85.

Figure D.38 – Finite element analysis and control schema EXPRESS-G diagram, 38 of 85.

Figure D.39 – Finite element analysis and control schema EXPRESS-G diagram, 39 of 85.

Figure D.40 – Finite element analysis and control schema EXPRESS-G diagram, 40 of 85.

Figure D.41 – Finite element analysis and control schema EXPRESS-G diagram, 41 of 85.

Figure D.42 – Finite element analysis and control schema EXPRESS-G diagram, 42 of 85.

Figure D.43 – Finite element analysis and control schema EXPRESS-G diagram, 43 of 85.

Figure D.44 – Finite element analysis and control schema EXPRESS-G diagram, 44 of 85.

Figure D.45 – Finite element analysis and control schema EXPRESS-G diagram, 45 of 85.

Figure D.46 – Finite element analysis and control schema EXPRESS-G diagram, 46 of 85.

Figure D.47 – Finite element analysis and control schema EXPRESS-G diagram, 47 of 85.

Figure D.48 – Finite element analysis and control schema EXPRESS-G diagram, 48 of 85.

Figure D.49 – Finite element analysis and control schema EXPRESS-G diagram, 49 of 85.

Figure D.50 – Finite element analysis and control schema EXPRESS-G diagram, 50 of 85.

Figure D.51 – Finite element analysis and control schema EXPRESS-G diagram, 51 of 85.

Figure D.52 – Finite element analysis and control schema EXPRESS-G diagram, 52 of 85.

Figure D.53 – Finite element analysis and control schema EXPRESS-G diagram, 53 of 85.

Figure D.54 – Finite element analysis and control schema EXPRESS-G diagram, 54 of 85.

Figure D.55 – Finite element analysis and control schema EXPRESS-G diagram, 55 of 85.

Figure D.56 – Finite element analysis and control schema EXPRESS-G diagram, 56 of 85.

Figure D.57 – Finite element analysis and control schema EXPRESS-G diagram, 57 of 85.

Figure D.58 – Finite element analysis and control schema EXPRESS-G diagram, 58 of 85.

Figure D.59 – Finite element analysis and control schema EXPRESS-G diagram, 59 of 85.

Figure D.60 – Finite element analysis and control schema EXPRESS-G diagram, 60 of 85.

Figure D.61 – Finite element analysis and control schema EXPRESS-G diagram, 61 of 85.

Figure D.62 – Finite element analysis and control schema EXPRESS-G diagram, 62 of 85.

Figure D.63 – Finite element analysis and control schema EXPRESS-G diagram, 63 of 85.

Figure D.64 – Finite element analysis and control schema EXPRESS-G diagram, 64 of 85.

Figure D.65 – Finite element analysis and control schema EXPRESS-G diagram, 65 of 85.

Figure D.66 – Finite element analysis and control schema EXPRESS-G diagram, 66 of 85.

Figure D.67 – Finite element analysis and control schema EXPRESS-G diagram, 67 of 85.

Figure D.68 – Finite element analysis and control schema EXPRESS-G diagram, 68 of 85.

Figure D.69 – Finite element analysis and control schema EXPRESS-G diagram, 69 of 85.

Figure D.70 – Structural response representation schema EXPRESS-G diagram, 70 of 85.

Figure D.71 – Structural response representation schema EXPRESS-G diagram, 71 of 85.

Figure D.72 – Structural response representation schema EXPRESS-G diagram, 72 of 85.

Figure D.73 – Structural response representation schema EXPRESS-G diagram, 73 of 85.

Figure D.74 – Structural response representation schema EXPRESS-G diagram, 74 of 85.

Figure D.75 – Structural response representation schema EXPRESS-G diagram, 75 of 85.

Figure D.76 – Structural response representation schema EXPRESS-G diagram, 76 of 85.

Figure D.77 – Finite element analysis and control schema EXPRESS-G diagram, 77 of 85.

Figure D.78 – Finite element analysis and control schema EXPRESS-G diagram, 78 of 85.

Figure D.79 – Finite element analysis and control schema EXPRESS-G diagram, 79 of 85.

Figure D.80 – Finite element analysis and control schema EXPRESS-G diagram, 80 of 85.

Figure D.81 – Finite element analysis and control schema EXPRESS-G diagram, 81 of 85.

Figure D.82 – Finite element analysis and control schema EXPRESS-G diagram, 82 of 85.

Figure D.83 – Finite element analysis and control schema EXPRESS-G diagram, 83 of 85.

Figure D.84 – Finite element analysis and control schema EXPRESS-G diagram, 84 of 85.

Figure D.85 – Scalar vector tensor schema EXPRESS-G diagram, 85 of 85.

Index

action_type 243
additional_0d_nodes 203
additional_1d_nodes 203
additional_2d_nodes 204
additional_3d_nodes 205
additional_node 30
aggregated_angular_variable 230
aggregated_scalar_variable 229
aggregated_tensor2_3d_variable 240
aggregated_vector_3d_variable 236
aligned_axis_tolerance 118
aligned_curve_3d_element_coordinate_system 132
aligned_surface_2d_element_coordinate_system 129
aligned_surface_3d_element_coordinate_system 125
analysis_message 349
analysis_step 246
angular_value 362
anisotropic_symmetric_tensor2_2d 364
anisotropic_symmetric_tensor2_3d 366
anisotropic_symmetric_tensor4_2d 367
anisotropic_symmetric_tensor4_3d 369
appropriate_list_value_existence 359
appropriate_set_value_existence 359
appropriate_value_existence 360
arbitrary_volume_2d_element_coordinate_system 122
arbitrary_volume_3d_element_coordinate_system 119
axi_or_plane 17
axisymmetric_2d_element_property 195
axisymmetric_curve_2d_element_descriptor 83
axisymmetric_curve_2d_element_representation 74
axisymmetric_surface_2d_element_descriptor 81
axisymmetric_surface_2d_element_representation 66
axisymmetric_volume_2d_element_descriptor 78
axisymmetric_volume_2d_element_representation 58

boundary_aggregated_variable 224
boundary_curve_scalar_variable 228
boundary_curve_vector_3d_variable 235
boundary_edge_variable 225
boundary_surface_scalar_variable 228
boundary_surface_vector_3d_variable 236
boundary_variable 223
build_direction_node 208

calculated_state 260
consistent_element_reference 212
consistent_geometric_reference 209
consistent_list_values 355
consistent_set_values 354
consistent_value 355
constant_surface_3d_element_coordinate_system 129
constraint_element 249
control_analysis_step 247
control_linear_static_analysis_step_with_harmonic 249
control_linear_static_analysis_step 248
control_process 255
control_result_relationship 259
control_static_load_increment_process 255
control 246
coordinate_system_type 17
curve_2d_element_basis 85
curve_2d_element_constant_specified_variable_value 328
curve_2d_element_constant_specified_volume_variable_value 329
curve_2d_element_coordinate_system 135
curve_2d_element_descriptor 32
curve_2d_element_field_variable_definition 325
curve_2d_element_group 199
curve_2d_element_integrated_matrix_with_definition 159
curve_2d_element_integrated_matrix 159
curve_2d_element_integration 159
curve_2d_element_location_point_variable_value 327
curve_2d_element_location_point_volume_variable_values 325
curve_2d_element_output_reference 245
curve_2d_element_property 191
curve_2d_element_representation 31
curve_2d_element_value_and_location 326
curve_2d_element_value_and_volume_location 326
curve_2d_node_field_aggregated_variable_values 339
curve_2d_node_field_section_variable_values 338
curve_2d_node_field_variable_definition 338
curve_2d_state_coordinate_system 242
curve_2d_whole_element_variable_value 327
curve_3d_element_basis 84
curve_3d_element_constant_specified_variable_value 322
curve_3d_element_constant_specified_volume_variable_value 323
curve_3d_element_coordinate_system 34
curve_3d_element_descriptor 83
curve_3d_element_field_variable_definition 319
curve_3d_element_group 199
curve_3d_element_integrated_matrix_with_definition 156

curve_3d_element_integrated_matrix 156
curve_3d_element_integration 156
curve_3d_element_length_integration_explicit 158
curve_3d_element_length_integration_rule 157
curve_3d_element_length_integration 157
curve_3d_element_location_point_variable_values 321
curve_3d_element_location_point_volume_variable_values 319
curve_3d_element_nodal_specified_variable_values 324
curve_3d_element_output_reference 244
curve_3d_element_position_weight 158
curve_3d_element_property 188
curve_3d_element_representation 71
curve_3d_element_value_and_location 320
curve_3d_element_value_and_volume_location 320
curve_3d_node_field_aggregated_variable_values 337
curve_3d_node_field_section_variable_values 337
curve_3d_node_field_variable_definition 336
curve_3d_state_coordinate_system 242
curve_3d_whole_element_variable_value 322
curve_constraint 253
curve_edge 35
curve_element_end_coordinate_system 34
curve_element_end_offset 193
curve_element_end_release_packet 194
curve_element_end_release 194
curve_element_freedom 29
curve_element_interval_constant 190
curve_element_interval_linearly_varying 190
curve_element_interval 189
curve_element_location 163
curve_element_purpose 23
curve_element_section_definition 191
curve_element_section_derived_definitions 192
curve_element_variable 225
curve_freedom_action_definition 343
curve_freedom_and_value_definition 342
curve_freedom_values 343
curve_matrix_property_type 26
curve_scalar_variable 226
curve_section_element_location 164
curve_section_integration_explicit 158
curve_vector_2d_variable 230
curve_vector_3d_variable 234
curve_volume_element_location 163
cylindrical_harmonic_number 219
cylindrical_point 43

cylindrical_symmetry_control 248

degree_of_freedom 28
direction_node 41
directionally_explicit_element_coefficient 91
directionally_explicit_element_coordinate_system_aligned 137
directionally_explicit_element_coordinate_system_arbitrary 137
directionally_explicit_element_coordinate_system 137
directionally_explicit_element_representation 89
dummy_node 31

element_2d_shape 24
element_analysis_message 350
element_aspect 34
element_definition 11
element_descriptor 77
element_geometric_relationship 54
element_group_analysis_message 350
element_group 196
element_integration_algebraic 146
element_material 166
element_nodal_freedom_actions 341
element_nodal_freedom_terms 342
element_order 21
element_representation 53
element_sequence 256
element_volume 35
euler_angles 138
explicit_element_matrix 93
explicit_element_representation 91

fea_area_density 169
fea_axis2_placement_2d 41
fea_axis2_placement_3d 42
fea_column_normalised_monoclinic_symmetric_tensor4_3d 375
fea_column_normalised_orthotropic_symmetric_tensor4_3d 373
fea_group_relation 197
fea_group 196
fea_iso_orthotropic_symmetric_tensor4_3d 371
fea_isotropic_symmetric_tensor4_3d 370
fea_linear_elasticity 168
fea_mass_density 168
fea_material_property_representation_item 167
fea_material_property_representation 166
fea_model_2d 37
fea_model_3d 36
fea_model_definition 10

fea_model 36
fea_moisture_absorption 171
fea_parametric_point 160
fea_representation_item 38
fea_representation_relationship 42
fea_secant_coefficient_of_linear_thermal_expansion 170
fea_shell_bending_stiffness 172
fea_shell_membrane_bending_coupling_stiffness 173
fea_shell_membrane_stiffness 171
fea_shell_shear_stiffness 174
fea_tangential_coefficient_of_linear_thermal_expansion 169
fea_transverse_isotropic_symmetric_tensor4_3d 372
field_value 223
field_variable_definition 264
field_variable_element_definition 265
field_variable_element_group_value 329
field_variable_node_definition 331
field_variable_whole_model_value 330
freedom_and_coefficient 252
freedom_and_value 348

geometric_node 47
grounded_damper 88
grounded_spring 88

higher_order_element_representation 54

integration_rule 30
isotropic_symmetric_tensor2_3d 365

linear_constraint_equation_element_value 348
linear_constraint_equation_element 251
linear_constraint_equation_nodal_term 251
linearly_superimposed_state 260

map_to_common_cartesian 209
matrix_property_type 24
matrix_symmetry 27
measure_or_unspecified_value 223
message_level 241
model_or_control_element 219

necessary_freedom_coordinate_system_set 352
necessary_freedom_coordinate_system 351
necessary_value_coordinate_system 353
no_symmetry_control 248
nodal_freedom_action_definition 341
nodal_freedom_and_value_definition 340

nodal_freedom_values 341
node_analysis_message 350
node_definition 10
node_geometric_relationship 45
node_group 196
node_output_reference 245
node_representation 44
node_sequence 257
node_set 47
node_with_solution_coordinate_system 46
node_with_vector 46
node 45
number_of_terms 206

orthotropic_symmetric_tensor2_3d 365
output_request_state 261

parametric_curve_3d_element_coordinate_direction 135
parametric_curve_3d_element_coordinate_system 132
parametric_surface_2d_element_coordinate_system 131
parametric_surface_3d_element_coordinate_system 127
parametric_volume_2d_element_coordinate_system 122
parametric_volume_3d_element_coordinate_system 119
plane_2d_element_assumption 21
plane_2d_element_property 195
plane_curve_2d_element_descriptor 84
plane_curve_2d_element_representation 75
plane_surface_2d_element_descriptor 81
plane_surface_2d_element_representation 68
plane_volume_2d_element_descriptor 79
plane_volume_2d_element_representation 60
point_constraint 252
point_element_matrix 86
point_element_representation 85
point_freedom_action_definition 345
point_freedom_and_value_definition 344
point_freedom_values 344

required_0d_nodes 200
required_1d_nodes 200
required_2d_nodes 201
required_3d_nodes 202
result_analysis_step 258
result_linear_static_analysis_sub_step 258
result 257

scalar 363

shape_function 30
simple_plane_2d_element_property 195
single_point_constraint_element_values 348
single_point_constraint_element 250
solid_constraint 254
solid_freedom_action_definition 347
solid_freedom_and_value_definition 346
solid_freedom_values 347
specified_state 260
spherical_point 43
state_component 261
state_definition 264
state_with_harmonic 260
state 259
stationary_mass 87
structural_response_property_definition_representation 37
structural_response_property 10
surface_2d_edge 222
surface_2d_element_basis 82
surface_2d_element_boundary_constant_specified_surface_variable_value 311
surface_2d_element_boundary_constant_specified_variable_value 312
surface_2d_element_boundary_edge_constant_specified_surface_variable_value 316
surface_2d_element_boundary_edge_constant_specified_variable_value 317
surface_2d_element_boundary_edge_location_point_surface_variable_values 313
surface_2d_element_boundary_edge_location_point_variable_values 314
surface_2d_element_boundary_edge_nodal_specified_variable_values 318
surface_2d_element_boundary_edge_whole_edge_variable_value 315
surface_2d_element_boundary_location_point_surface_variable_values 309
surface_2d_element_boundary_nodal_specified_variable_values 313
surface_2d_element_boundary_whole_face_variable_value 310
surface_2d_element_constant_specified_variable_value 307
surface_2d_element_constant_specified_volume_variable_value 308
surface_2d_element_coordinate_system 33
surface_2d_element_descriptor 32
surface_2d_element_field_variable_definition 303
surface_2d_element_group 198
surface_2d_element_integrated_matrix_with_definition 154
surface_2d_element_integrated_matrix 154
surface_2d_element_integration 154
surface_2d_element_length_integration_explicit 155
surface_2d_element_length_integration_rule 155
surface_2d_element_length_integration 155
surface_2d_element_location_point_variable_values 306
surface_2d_element_location_point_volume_variable_values 304
surface_2d_element_nodal_specified_variable_values 308
surface_2d_element_output_reference 244

surface_2d_element_representation 31
surface_2d_element_value_and_location 305
surface_2d_element_value_and_volume_location 305
surface_2d_face 222
surface_2d_node_field_aggregated_variable_values 336
surface_2d_node_field_section_variable_values 335
surface_2d_node_field_variable_definition 335
surface_2d_state_coordinate_system 242
surface_2d_whole_element_variable_value 306
surface_3d_edge 221
surface_3d_element_basis 82
surface_3d_element_boundary_constant_specified_surface_variable_value 295
surface_3d_element_boundary_constant_specified_variable_value 296
surface_3d_element_boundary_edge_constant_specified_surface_variable_value 300
surface_3d_element_boundary_edge_constant_specified_variable_value 301
surface_3d_element_boundary_edge_location_point_surface_variable_values 298
surface_3d_element_boundary_edge_location_point_variable_values 299
surface_3d_element_boundary_edge_nodal_specified_variable_values 302
surface_3d_element_boundary_edge_whole_edge_variable_value 300
surface_3d_element_boundary_location_point_surface_variable_values 293
surface_3d_element_boundary_nodal_specified_variable_values 297
surface_3d_element_boundary_whole_face_variable_value 294
surface_3d_element_constant_specified_variable_value 291
surface_3d_element_constant_specified_volume_variable_value 292
surface_3d_element_coordinate_system 33
surface_3d_element_descriptor 80
surface_3d_element_field_integration_explicit 152
surface_3d_element_field_integration_rule 151
surface_3d_element_field_integration 151
surface_3d_element_field_variable_definition 287
surface_3d_element_group 198
surface_3d_element_integrated_matrix_with_definition 150
surface_3d_element_integrated_matrix 150
surface_3d_element_integration 150
surface_3d_element_location_point_variable_values 290
surface_3d_element_location_point_volume_variable_values 288
surface_3d_element_nodal_specified_variable_values 293
surface_3d_element_output_reference 244
surface_3d_element_representation 63
surface_3d_element_value_and_location 289
surface_3d_element_value_and_volume_location 289
surface_3d_face 221
surface_3d_node_field_aggregated_variable_values 334
surface_3d_node_field_section_variable_values 333
surface_3d_node_field_variable_definition 333
surface_3d_state_coordinate_system 242

surface_3d_whole_element_variable_value 291
surface_constraint 254
surface_element_location 161
surface_element_property 183
surface_element_purpose 22
surface_element_variable 225
surface_freedom_action_definition 346
surface_freedom_and_value_definition 345
surface_freedom_values 346
surface_matrix_property_type 25
surface_position_weight 152
surface_scalar_variable 227
surface_section_element_location_absolute 162
surface_section_element_location_dimensionless 163
surface_section_element_location 162
surface_section_field_constant 186
surface_section_field_varying 186
surface_section_field 186
surface_section_integration_explicit 153
surface_section_integration_rule 152
surface_section_integration 151
surface_section_position_weight 153
surface_section 187
surface_tensor2_2d_variable 237
surface_vector_2d_variable 232
surface_vector_3d_variable 234
surface_volume_element_location 161
symmetric_tensor2_2d 364
symmetric_tensor2_3d 365
symmetric_tensor4_2d 366
symmetric_tensor4_3d 368
symmetry_control 247
system_and_freedom 91

tensor1_2d 363
tensor1_3d 363
tensor1 363

uniform_surface_section_layered 188
uniform_surface_section 187
unspecified_value 223
user_defined_scalar_variable 230
user_defined_tensor2_2d_variable 239
user_defined_tensor2_3d_variable 241
user_defined_vector_2d_variable 234
user_defined_vector_3d_variable 237

valid_parametric_coordinate 207
variable_value_type 356
volume_2d_edge 221
volume_2d_element_basis 80
volume_2d_element_boundary_constant_specified_variable_value 282
volume_2d_element_boundary_edge_constant_specified_volume_variable_value 286
volume_2d_element_boundary_edge_location_point_volume_variable_values 284
volume_2d_element_boundary_edge_nodal_specified_variable_values 286
volume_2d_element_boundary_edge_whole_edge_variable_value 285
volume_2d_element_boundary_location_point_variable_values 280
volume_2d_element_boundary_nodal_specified_variable_values 283
volume_2d_element_boundary_whole_face_variable_value 281
volume_2d_element_constant_specified_variable_value 279
volume_2d_element_coordinate_system 33
volume_2d_element_descriptor 32
volume_2d_element_field_integration_explicit 149
volume_2d_element_field_integration_rule 149
volume_2d_element_field_integration 149
volume_2d_element_field_variable_definition 276
volume_2d_element_group 198
volume_2d_element_integrated_matrix_with_definition 148
volume_2d_element_integrated_matrix 148
volume_2d_element_location_point_variable_values 277
volume_2d_element_nodal_specified_variable_values 279
volume_2d_element_output_reference 243
volume_2d_element_representation 31
volume_2d_element_value_and_location 278
volume_2d_face 220
volume_2d_node_field_variable_definition 332
volume_2d_whole_element_variable_value 278
volume_3d_edge 220
volume_3d_element_basis 79
volume_3d_element_boundary_constant_specified_variable_value 271
volume_3d_element_boundary_edge_constant_specified_volume_variable_value 274
volume_3d_element_boundary_edge_location_point_volume_variable_values 273
volume_3d_element_boundary_edge_nodal_specified_variable_values 275
volume_3d_element_boundary_edge_whole_edge_variable_value 274
volume_3d_element_boundary_location_point_variable_values 269
volume_3d_element_boundary_nodal_specified_variable_values 272
volume_3d_element_boundary_whole_face_variable_value 270
volume_3d_element_constant_specified_variable_value 268
volume_3d_element_coordinate_system 33
volume_3d_element_descriptor 78
volume_3d_element_field_integration_explicit 147
volume_3d_element_field_integration_rule 147
volume_3d_element_field_integration 146

volume_3d_element_field_variable_definition 265
volume_3d_element_group 197
volume_3d_element_integrated_matrix_with_definition 146
volume_3d_element_integrated_matrix 145
volume_3d_element_location_point_variable_values 266
volume_3d_element_nodal_specified_variable_values 268
volume_3d_element_output_reference 243
volume_3d_element_representation 55
volume_3d_element_shape 24
volume_3d_element_value_and_location 266
volume_3d_face 219
volume_3d_node_field_variable_definition 332
volume_3d_whole_element_variable_value 267
volume_aggregated_variable 224
volume_angular_variable 229
volume_element_location 161
volume_element_purpose 22
volume_position_weight 147
volume_scalar_variable 228
volume_tensor2_3d_variable 239
volume_variable 224
volume_vector_3d_variable 235

whole_model_analysis_message 349

æ