

A First-Order Logic Formalization of the Unified Foundational Ontology

Daniele Porelo, Claudenir M. Fonseca, João Paulo A. Almeida,
Giancarlo Guizzardi, Tiago Prince Sales

December 1, 2021

Abstract

This document presents a formalization of the Unified Foundational Ontology (UFO) in first-order logic. This formalization is documented by means of three complementary representations: (i) a representation in standard Common Logic using the CLIF syntax; (ii) a representation in natural language; and, when applicable, (iii) a UML-based diagrammatic representation. The presented formalization is supported by consistency and satisfiability checks performed through automated proofing tools.

1 Introduction

This document presents a formalization of the Unified Foundational Ontology (UFO) in first-order logic. This formalization is documented by means of three complementary representations: (i) a representation in standard Common Logic using the CLIF syntax; (ii) a representation in natural language; and, when applicable, (iii) a UML-based diagrammatic representation. The presented formalization is supported by consistency and satisfiability checks performed through automated proofing tools.

The remainder of this document is organized as a single formalization section (Section 2), which contains subsections for each submodule of the ontology.

2 Formalization

This section contains the formalization of the Unified Foundational Ontology (UFO) in first-order logics. This formalization is organized in several subsections where each presents the formalization of a portion of the whole ontology. The formalization is presented through different equivalent representations, designed to support the understanding of its contents: (i) a representation in standard Common Logic using the CLIF syntax; (ii) a representation in natural language; and, when applicable, (iii) a UML-based diagrammatic representation.

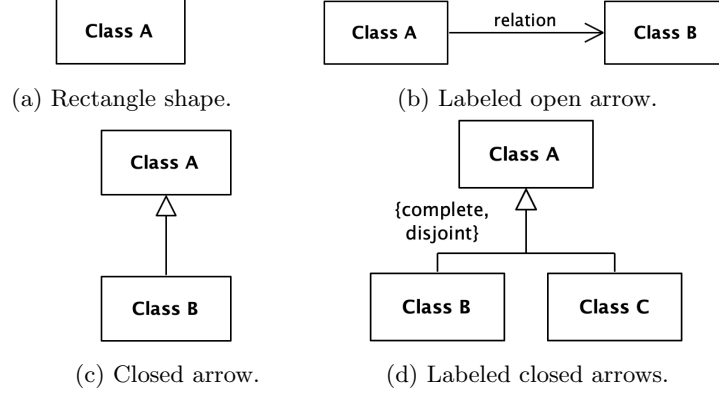


Figure 1: UML-based representation of first-order logic axioms.

The UML-based diagrammatic representation serves as a visual representation certain predicates and axioms, being each element in Figure 1 being translated as follows:

- Rectangle shape (Figure 1a): visual representation of unary predicates associated to types in the ontology; the associated predicate is shown in lower camel case with no spaces.

$classA(x)$

- Open arrow (Figure 1b): visual representation of binary predicates; the predicate associated to the arrows' label is shown in lower camel case with no spaces; the predicate can only be true for any x and y if it is also true predicates associated to the types of each end (keeping the order of the arrow in the binary predicate's positions); this representation may also be associated to ternary predicates iff its third position represents a time-index.

$\forall x, y (relation(x, y) \rightarrow (classA(x) \wedge classB(y)))$

$\forall x, y, w (relation(x, y, w) \rightarrow (classA(x) \wedge classB(y) \wedge world(w)))$

- Closed arrow (Figure 1c): visual representation of specializations between ontology's types, where the type in the tail of the arrow is a subtype of the type in the head of the arrow.

$\forall x (classB(x) \rightarrow classA(x))$

- Labeled closed arrow (Figure 1d): visual representation of disjoint and/or complete constraints over sets specializations between ontology's types.

$\forall x (classB(x) \rightarrow classA(x))$

$\forall x (classC(x) \rightarrow classA(x))$

$\forall x (classA(x) \rightarrow (classB(x) \vee classC(x)))$

$\neg \exists x (classB(x) \wedge classC(x))$

{complete}

{disjoint}

2.1 Partial Taxonomy of UFO: Thing

This subsection presents most general types of UFO's taxonomy specializing the type Thing (Figure 2).

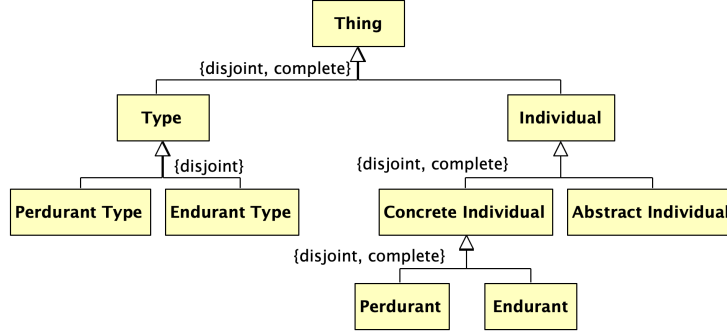


Figure 2: Visual representation of UFO's taxonomy of Thing.

- a1** For every x , x is a Thing iff x is either a Type or an Individual.

$$\forall x(\text{type_}(x) \vee \text{individual}(x) \leftrightarrow \text{thing}(x))$$

```

1 (cl-text ax_thing_taxonomy
2 (forall (x)
3   (iff (or (type_ x) (individual x))
4     (thing x))
5   )
6 )

```

- a2** There is no x such that it is a Type and an Individual.

$$\neg \exists x(\text{type_}(x) \wedge \text{individual}(x))$$

```

7 (cl-text ax_thing_partition
8 (not (exists (x)
9   (and (type_ x) (individual x)))
10 )
11 )

```

- a3** For every x , x is an Individual iff x is either a Concrete Individual or an Abstract Individual.

$$\forall x(\text{concreteIndividual}(x) \vee \text{abstractIndividual}(x) \leftrightarrow \text{individual}(x))$$

```

12 (cl-text ax_individual_taxonomy
13 (forall (x)
14   (iff (or (concreteIndividual x) (abstractIndividual x))
15     (individual x))
16   )
17 )

```

a4 There is no x such that it is a Concrete Individual and an Abstract Individual.

$$\neg \exists x(\text{concreteIndividual}(x) \wedge \text{abstractIndividual}(x))$$

```
18 (cl-text ax_individual_partition
19 (not (exists (x)
20   (and (concreteIndividual x) (abstractIndividual x)))
21 )
22 )
```

a5 For every x , x is a Concrete Individual iff x is either a Perdurant or an Endurant.

$$\forall x(\text{endurant}(x) \vee \text{perdurant}(x) \leftrightarrow \text{concreteIndividual}(x))$$

```
23 (cl-text ax_concreteIndividual_taxonomy
24 (forall (x)
25   (iff (or (endurant x) (perdurant x))
26     (concreteIndividual x))
27 )
28 )
```

a6 There is no x such that it is a Perdurant and an Endurant.

$$\neg \exists x(\text{endurant}(x) \wedge \text{perdurant}(x))$$

```
29 (cl-text ax_concreteIndividual_partition
30 (not (exists (x)
31   (and (endurant x) (perdurant x)))
32 )
33 )
```

a7 For every x , x is a Concrete Individual iff x is either a Perdurant or an Endurant.

$$\forall x(\text{endurantType}(x) \vee \text{perdurantType}(x) \rightarrow \text{type_}(x))$$

```
34 (cl-text ax_type_taxonomy
35 (forall (x)
36   (if (or (endurantType x) (perdurantType x))
37     (type_ x))
38 )
39 )
```

a8 There is no x such that it is a Perdurant Type and an Endurant Type.

$$\neg \exists x(\text{endurantType}(x) \wedge \text{perdurantType}(x))$$

```
40 (cl-text ax_type_partition
41 (not (exists (x)
42   (and (endurantType x) (perdurantType x)))
43 )
44 )
```

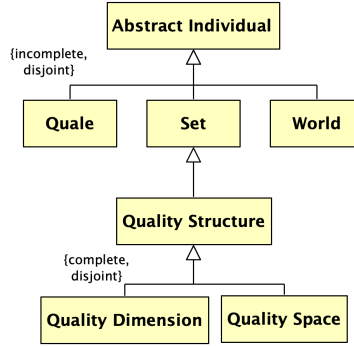


Figure 3: Visual representation of UFO's taxonomy of Abstract Individual.

2.2 Partial Taxonomy of UFO: Abstract Individual

This subsection presents a portion of UFO's taxonomy specializing the type Abstract Individual (Figure 3).

a9 Every x that is a Quale is also an Abstract Individual.

$$\forall x(\text{quale}(x) \rightarrow \text{abstractIndividual}(x))$$

```

1 (cl-text ax_abstractIndividual_taxonomy_quale
2 (forall (x)
3   (if (quale x)
4     (abstractIndividual x))
5   )
6 )

```

a10 Every x that is a Set is also an Abstract Individual.

$$\forall x(\text{set_}(x) \rightarrow \text{abstractIndividual}(x))$$

```

7 (cl-text ax_abstractIndividual_taxonomy_set
8 (forall (x)
9   (if (set_ x)
10     (abstractIndividual x))
11   )
12 )

```

a11 Every x that is a World is also an Abstract Individual.

$$\forall x(\text{world}(x) \rightarrow \text{abstractIndividual}(x))$$

```

13 (cl-text ax_abstractIndividual_taxonomy_world
14 (forall (x)
15   (if (world x)
16     (abstractIndividual x))
17   )
18 )

```

a12 There is no x such that it is a Quale, a Set, and a World (pairwise disjoint).

$$\neg \exists x ((\text{quale}(x) \wedge \text{set_}(x)) \vee (\text{quale}(x) \wedge \text{world}(x)) \vee (\text{set_}(x) \wedge \text{world}(x)))$$

```

19 (cl-text ax_abstractIndividual_pairwiseDisjoint
20 (not (exists (x)
21   (or (and (quale x) (set_ x)) (and (quale x) (world x)) (and
22     (set_ x) (world x))))
23 )

```

a13 Every x that is a Quality Structure is also a Set.

$$\forall x (\text{qualityStructure}(x) \rightarrow \text{set_}(x))$$

```

24 (cl-text ax_set_taxonomy_qualityStructure
25 (forall (x)
26   (if (qualityStructure x)
27     (set_ x))
28 )
29 )

```

a14 For every x , x is a Quality Structure iff x is either a Quality Dimension or a Quality Space.

$$\forall x (\text{qualityDimension}(x) \vee \text{qualitySpace}(x) \leftrightarrow \text{qualityStructure}(x))$$

```

30 (cl-text ax_qualityStructure_taxonomy
31 (forall (x)
32   (iff (or (qualityDimension x) (qualitySpace x))
33     (qualityStructure x))
34 )
35 )

```

a15 There is no x such that it is a Quality Dimension and a Quality Space.

$$\neg \exists x (\text{qualityDimension}(x) \wedge \text{qualitySpace}(x))$$

```

36 (cl-text ax_qualityStructure_partition
37 (not (exists (x)
38   (and (qualityDimension x) (qualitySpace x)))
39 )
40 )

```

2.3 UFO Taxonomy

2.3.1 Partial Taxonomy of Abstract Individual

```

51 % Abstract Individual
52
53 fof(ax_abstractIndividual_taxonomy_quale, axiom, (
54   ![X]: (quale(X) => (abstractIndividual(X)))
55 )).
56
57 fof(ax_abstractIndividual_taxonomy_set, axiom, (
58   ![X]: (set_(X) => (abstractIndividual(X)))

```

```

59)).
60
61 fof(ax_abstractIndividual_taxonomy_world, axiom, (
62   ![X]: (world(X) => (abstractIndividual(X)))
63)).
64
65 fof(ax_abstractIndividual_pairwiseDisjoint, axiom, (
66   ~?[X]: ((quale(X) & set_(X)) | (quale(X) & world(X)) | (set_(X) &
67   world(X)))
68)).
69 % Set
70
71 fof(ax_set_taxonomy_qualityStructure, axiom, (
72   ![X]: (qualityStructure(X) => (set_(X)))
73)).
74
75 % Quality Structure
76
77 fof(ax_qualityStructure_taxonomy, axiom, (
78   ![X]: ((qualityDimension(X) | qualitySpace(X)) <=> (
79   qualityStructure(X)))
80)).
81
82 fof(ax_qualityStructure_partition, axiom, (
83   ~?[X]: (qualityDimension(X) & qualitySpace(X))
84)).
85
86 % Abstract Individual partial taxonomy instances
87 % (tested to rule out trivial models)

```

2.3.2 Partial Taxonomy of Endurant

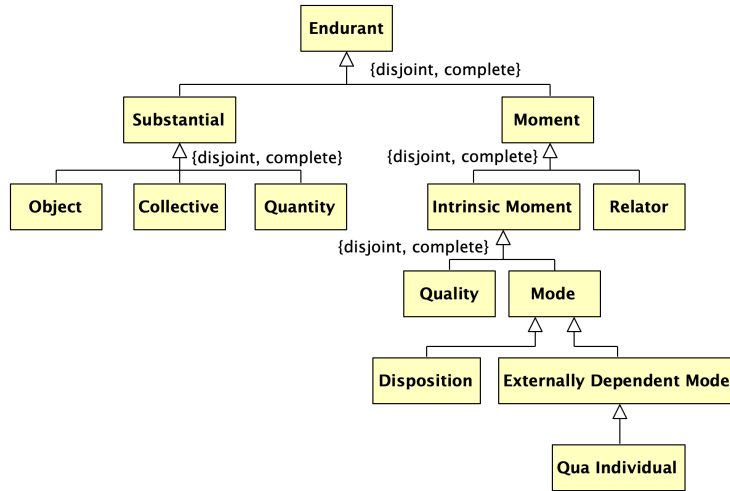


Figure 4: Partial Taxonomy of UFO – Endurant.

```

88 % fof(ax_abstractIndividual_instances, axiom, (
89 %   set_(set1) & quale(quale1) & qualityStructure(qualityStructure1
    ) & qualityDimension(qualityDimension1) & qualitySpace(
    qualitySpace1) & world(world1)
90 % )).
91
92 % Endurant
93
94 fof(ax_endurant_taxonomy, axiom, (
95   ![X]: ((substantial(X) | moment(X)) <=> (endurant(X)))
96 )).
97
98 fof(ax_endurant_partition, axiom, (
99   ~?[X]: (substantial(X) & moment(X))
100 )).
101
102 % Substantial
103
104 fof(ax_substantial_taxonomy, axiom, (
105   ![X]: ((object(X) | collective(X) | quantity(X)) <=> (substantial
    (X)))
106 )).
107
108 fof(ax_substantial_partition, axiom, (
109   ~?[X]: ((object(X) & collective(X)) | (object(X) & quantity(X)) |
    (collective(X) & quantity(X)))
110 )).
111
112 % Moment
113
114 fof(ax_moment_taxonomy, axiom, (
115   ![X]: ((intrinsicMoment(X) | relator(X)) <=> (moment(X)))
116 )).
117
118 fof(ax_moment_partition, axiom, (
119   ~?[X]: (intrinsicMoment(X) & relator(X))
120 )).
121
122 % Intrinsic Moment
123
124 fof(ax_intrinsicMoment_taxonomy, axiom, (
125   ![X]: ((quality(X) | mode(X)) <=> (intrinsicMoment(X)))
126 )).
127
128 fof(ax_intrinsicMoment_partition, axiom, (
129   ~?[X]: (quality(X) & mode(X))
130 )).
131
132 % Mode
133
134 fof(ax_mode_taxonomy_externallyDependentMode, axiom, (
135   ![X]: (externallyDependentMode(X) => (mode(X)))
136 )).
137
138 % Externally Dependent Mode
139
140 fof(ax_externallyDependentMode_taxonomy_quaIndividual, axiom, (

```



```

141 ![X]: (quaIndividual(X) => (externallyDependentMode(X)))
142 )).
143
144 % Endurant partial taxonomy instances
145 % (tested to rule out trivial models)

```

2.3.3 Partial Taxonomy of Endurant Type (on ontological natures)

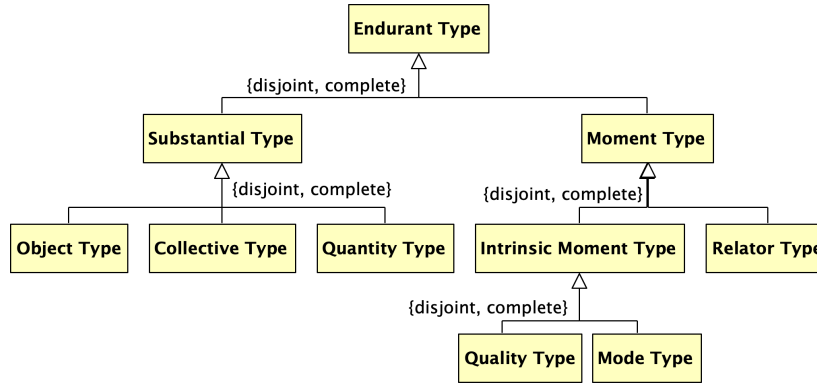


Figure 5: Partial Taxonomy of UFO – Endurant Types (by ontological nature).

```

147 % fof(ax_endurant_instances, axiom, (
148 %   substantial(substantial1) & moment(moment1) & object(object1) &
149 %   collective(collective1) & quantity(quantity1) &
150 %   intrinsicMoment(intrinsicMoment1) & relator(relator1) & quality
151 %   (quality1) & mode(mode1) & disposition(disposition1) &
152 %   externallyDependentMode(externallyDependentMode1) &
153 %   quaIndividual(quaIndividual1)
154 % ))).
155
156 % Endurant Type (by ontological nature)
157
158 fof(ax_endurantType_taxonomy_nature, axiom, (
159   ![X]: ((substantialType(X) | momentType(X)) <=> (endurantType(X))
160   )
161 ))).
162
163 fof(ax_endurantType_partition_nature, axiom, (
164   ~?[X]: (substantialType(X) & momentType(X))
165 ))).
166
167 % Substantial Type
168
169 fof(ax_substantialType_taxonomy, axiom, (
170   ![X]: ((objectType(X) | collectiveType(X) | quantityType(X)) <=>
171   (substantialType(X)))
172 ))).
173
174 fof(ax_substantialType_partition, axiom, (

```

```

168 ~?[X]: ((objectType(X) & collectiveType(X)) | (objectType(X) &
169 quantityType(X)) | (collectiveType(X) & quantityType(X)))
170 ))).
171 % Moment Type
172
173 fof(ax_momentType_taxonomy, axiom, (
174 ! [X]: ((intrinsicMomentType(X) | relatorType(X)) <=> (momentType(
175 X)))
176 ))).
177 fof(ax_momentType_partition, axiom, (
178 ~?[X]: (intrinsicMomentType(X) & relatorType(X))
179 ))).
180
181 % Intrinsic Moment Type
182
183 fof(ax_intrinsicMomentType_taxonomy, axiom, (
184 ! [X]: ((qualityType(X) | modeType(X)) <=> (intrinsicMomentType(X)
185 ))
186 ))).
187 fof(ax_intrinsicMomentType_partition, axiom, (
188 ~?[X]: (qualityType(X) & modeType(X))
189 ))).
190
191 % Endurant Type (by ontological nature) partial taxonomy instances
192 % (tested to rule out trivial models)

```

2.3.4 Partial Taxonomy of Endurant Type (on modal properties of types)

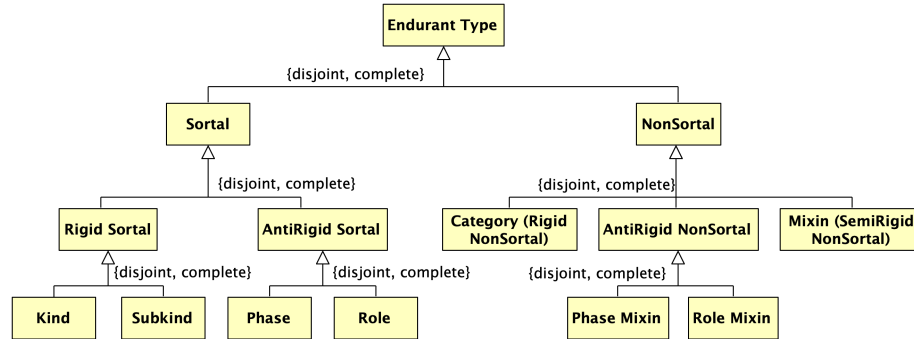


Figure 6: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```

194 % fof(ax_endurantType_instances_natures, axiom, (
195 %   substantialType(substantialType1) & momentType(momentType1) &
196 %   objectType(objectType1) & collectiveType(collectiveType1) &
197 %   quantityType(quantityType1) & intrinsicMomentType(

```

```

    intrinsicMomentType1) & relatorType(relatorType1) & qualityType
    (qualityType1) & modeType(modeType1)
196 % )).
197
198 % Endurant Type (by modal properties of types)
199
200 fof(ax_endurantType_taxonomy_properties, axiom, (
201   ![X]: ((sortal(X) | nonSortal(X)) <=> (endurantType(X)))
202 )).
203
204 fof(ax_endurantType_partition_properties, axiom, (
205   ~?[X]: (sortal(X) & nonSortal(X))
206 )).
207
208 % Sortal
209
210 fof(ax_sortal_taxonomy, axiom, (
211   ![X]: ((rigidSortal(X) | antiRigidSortal(X)) <=> (sortal(X)))
212 )).
213
214 fof(ax_sortal_partition, axiom, (
215   ~?[X]: (rigidSortal(X) & antiRigidSortal(X))
216 )).
217
218 % Rigid Sortal
219
220 fof(ax_rigidSortal_taxonomy, axiom, (
221   ![X]: ((kind(X) | subkind(X)) <=> (rigidSortal(X)))
222 )).
223
224 fof(ax_rigidSortal_partition, axiom, (
225   ~?[X]: (kind(X) & subkind(X))
226 )).
227
228 % Anti-Rigid Sortal
229
230 fof(ax_antiRigidSortal_taxonomy, axiom, (
231   ![X]: ((phase(X) | role(X)) <=> (antiRigidSortal(X)))
232 )).
233
234 fof(ax_antiRigidSortal_partition, axiom, (
235   ~?[X]: (phase(X) & role(X))
236 )).
237
238 % Non-Sortal
239
240 fof(ax_nonSortal_taxonomy, axiom, (
241   ![X]: ((rigidNonSortal(X) | semiRigidNonSortal(X) |
242     antiRigidNonSortal(X)) <=> (nonSortal(X)))
243 )).
244
245 fof(ax_nonSortal_partition, axiom, (
246   ~?[X]: ((rigidNonSortal(X) & semiRigidNonSortal(X)) | (
247     rigidNonSortal(X) & antiRigidNonSortal(X)) | (
248       semiRigidNonSortal(X) & antiRigidNonSortal(X)))
249 )).

```

```

248 % Category
249
250 fof(ax_rigidNonSortal_taxonomy, axiom, (
251   ![X]: (rigidNonSortal(X) <=> (category(X)))
252 )).
253
254 % Mixin
255
256 fof(ax_semiRigidNonSortal_taxonomy, axiom, (
257   ![X]: (semiRigidNonSortal(X) <=> (mixin(X)))
258 )).
259
260 % Anti-Rigid Non-Sortal
261
262 fof(ax_antiRigidNonSortal_taxonomy, axiom, (
263   ![X]: ((phaseMixin(X) | roleMixin(X)) <=> (antiRigidNonSortal(X))
264 )).
265
266 fof(ax_antiRigidNonSortal_partition, axiom, (
267   ~?[X]: (phaseMixin(X) & roleMixin(X))
268 )).
269
270 % Endurant Type (by modal properties of types) partial taxonomy
271 % instances
272 % (tested to rule out trivial models)

```

2.3.5 Defining Types, Individuals, and Specialization

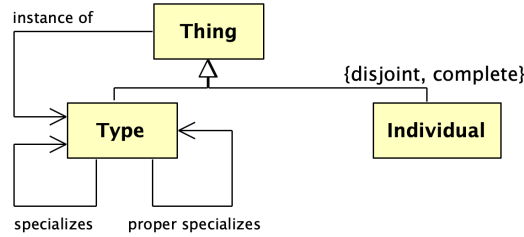


Figure 7: Types, individuals, instantiation, and specialization.

```

273 % fof(ax_endurantType_instances_properties, axiom, (
274 %   sortal(sortal1) & nonSortal(nonSortal1) & rigidSortal(
275 %     rigidSortal1) & antiRigidSortal(antiRigidSortal1) & kind(kind1)
276 %     & subkind(subkind1) & phase(phase1) & role(role1) & category(
277 %       category1) & mixin(mixin1) & antiRigidNonSortal(
278 %         antiRigidNonSortal1) & phaseMixin(phaseMixin1) & roleMixin(
279 %           roleMixin1)
280 %   ))
281 % %%%%%%%%% Instance of, Types, and Individuals %%%%%%%%%
282
283 fof(ax_dIof, axiom, (
284   ![X,Y,W]: (iof(X,Y,W) => (type_(Y) & world(W)))
285 )).

```

```

281)).
282
283 fof(ax_dType_a1, axiom, (
284   ![X]: (type_(X) <=> (?[Y,W]: iof(Y,X,W)))
285)).
286
287 fof(ax_dIndividual_a2, axiom, (
288   ![X]: (individual(X) <=> (~?[Y,W]: iof(Y,X,W)))
289)).
290
291 fof(ax_multiLevel_a3, axiom, (
292   ![X,Y,W]: (iof(X,Y,W) => (type_(X) | individual(X)))
293)).
294
295 fof(ax_twoLevelConstrained_a4, axiom, (
296   ~?[X,Y,Z,W]: (type_(X) & iof(X,Y,W) & iof(Y,Z,W))
297)).
298
299 % Instantiation relations
300 % (tested to rule out trivial models)
301
302 % fof(ax_iofInUse, axiom, (
303 %   type_(t2) & individual(i2) & world(w2) & iof(i2,t2,w2)
304 %)).
305
306 % Ax |= "th_everythingIsAThing_t1"; conjecture commented for
    convenience
307
308 % fof(th_everythingIsAThing_t1, conjecture, (
309 %   ![X]: (type_(X) | individual(X))
310 %)).
311
312 % Ax |= "th_thingPartition_t2"; conjecture commented for
    convenience
313
314 % fof(th_thingPartition_t2, conjecture, (
315 %   ~?[X]: (type_(X) & individual(X))
316 %)).
317
318 %%%%%%%%% Specialization and Proper Specialization %%%%%%%%%
319
320 fof(ax_dSpecializes, axiom, (
321   ![X,Y]: (specializes(X,Y) => (type_(X) & type_(Y)))
322)).
323
324 fof(ax_specialization_a5, axiom, (
325   ![T1,T2]: (specializes(T1,T2) <=> (
326     type_(T1) & type_(T2) & ![W]: (world(W) => ![E]: (iof(E,T1,W)
327       => iof(E,T2,W)))
328   ))).
329
330 fof(ax_properSpecializes_d1, axiom, (
331   ![X,Y]: (properSpecializes(X,Y) <=> (specializes(X,Y) & ~
    specializes(Y,X)))
332)).
333

```

```

334 % Ax |= "th_cyclicSpecializations_t3"; conjecture commented for
      convenience
335
336 % fof(th_cyclicSpecializations_t3, conjecture, (
337 %   ![X,Y]: (specializes(X,Y) => (specializes(X,X) & specializes(Y,
338 %   Y)))
339 % )).
340 % Ax |= "th_transitiveSpecializations_t4"; conjecture commented for
      convenience
341
342 % fof(th_transitiveSpecializations_t4, conjecture, (
343 %   ![X,Y,Z]: ((specializes(X,Y) & specializes(Y,Z)) => (
344 %   specializes(X,Z)))
345 % )).
346 fof(ax_sharedSpecializations_a6, axiom, (
347   ![T1,T2]: (![X,W]: ((iof(X,T1,W) & iof(X,T2,W) & ~specializes(T1,
348   T2) & ~specializes(T2,T1)) => (
349   (?[T3]: (specializes(T1,T3) & specializes(T2,T3) & iof(X,T3,W)
350   )))
351   | (?[T3]: (specializes(T3,T1) & specializes(T3,T2) & iof(X,T3,
352   W))))
353 % Specialization relations
354 % (tested to rule out trivial models)
355
356 % fof(ax_specializesInUse, axiom, (

```

2.3.6 Defining Rigidity and Sortality

```

360 % Sortality and Rigidity %
361
362 % Rigidity
363
364 fof(ax_dRigid_a18, axiom, (
365   ![T]: (rigid(T) <=> (endurantType(T) & (
366   ![X]: ((?[W1]: (world(W1) & iof(X,T,W1))) => (![W2]: (world(W2)
367   => iof(X,T,W2)))))
368   )))
369
370 fof(ax_dAntiRigid_a19, axiom, (
371   ![T]: (antiRigid(T) <=> (endurantType(T) & (
372   ![X]: ((?[W1]: (world(W1) & iof(X,T,W1))) => (?[W2]: (world(W2)
373   & ~iof(X,T,W2)))))
374   )))
375
376 fof(ax_dSemiRigid_a20, axiom, (
377   ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid
378   (T)))
379   )))

```

```

380 % Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented
      for convenience
381
382 % fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
383 %   ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid
      (T)))
384 % )).
385
386 % Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented
      for convenience
387
388 % fof(th_pairwiseDisjointRigidities_t6, conjecture, (
389 %   ![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)
      )) | (rigid(T) & antiRigid(T)))
390 % )).
391
392 % Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture
      commented for convenience
393
394 % fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
395 %   ![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
396 % )).
397
398 % Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8";
      conjecture commented for convenience
399
400 % fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture,
      (
401 %   ![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
402 % )).
403
404 % Rigidity properties
405 % (tested to rule out trivial models)
406
407 % fof(ax_rigidityInUse, axiom, (
408 %   endurantType(t4_1) & endurantType(t4_2) & endurantType(t4_3) &
      rigid(t4_1) & semiRigid(t4_2) & antiRigid(t4_3) &
      properSpecializes(t4_1,t4_2) & properSpecializes(t4_3,t4_1)
409 % )).
410
411 % Sortality
412
413 fof(ax_endurantsKind_a21, axiom, (
414 %   ![E]: (endurant(E) => (
415 %     ?[U]: (kind(U) & (![W]: (world(W) => iof(E,U,W))))
416 %   ))
417 % ).
418
419 fof(ax_uniqueKind_a22, axiom, (
420 %   ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
421 %     ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
422 %   ))
423 % ).
424
425 % Changing "ax_dSortal_a23" from the form it was defined in the
      paper to "sortals are endurant types that specialize some
      ultimate sortal" seem to express the same concept while

```

```

speeding up the execution of SPASS considerably
426
427 % fof(ax_dSortal_a23, axiom, (
428 %   ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,
429 %     W]: (iof(E,S,W) => iof(E,U,W)))))))
430
431 fof(ax_dSortal_a23, axiom, (
432 %   ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) &
433 %     specializes(S,U))))))
434
435 % If we have the taxonomy's axiomatization, then a24 becomes a
436 % theorem
437 % Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24";
438 % conjecture commented for convenience
439
440 % fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
441 %   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
442 % ))
443
444 % Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience
445
446 % fof(th_kindsAreRigid_t9, conjecture, (
447 %   ![U]: ((kind(U)) => (rigid(U)))
448 % ))
449
450 % Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented
451 % for convenience
452
453 % fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
454 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
455 %     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2
456 %       ,W2))))
457 % ))
458
459 % Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented
460 % for convenience
461
462 % fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
463 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
464 %     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
465 % ))
466
467 % Ax |= "th_kindsAreSortal_t12"; conjecture commented for
468 % convenience
469
470 % fof(th_kindsAreSortal_t12, conjecture, (
471 %   ![K]: ((kind(K)) => (sortal(K)))
472 % ))
473
474 % Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for
475 % convenience
476
477 % fof(th_sortalSpecializeKinds_t13, conjecture, (

```



```

473 %   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
474 %   )).
475
476 % Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented
      for convenience
477
478 % fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
479 %   ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) &
      specializes(S,U) & specializes(S,U2) & ~(U=U2))))
480 %   )).
481
482 % Sortality properties
483 % (tested to rule out trivial models)
484
485 % fof(ax_sortalityInUse, axiom, (
486 %   endurant(e5_1) & endurant(e5_2) & world(w5) & kind(k5_1) & kind
      (k5_2) & iof(e5_1,k5_1,w5) & iof(e5_1,k5_1,w5) & ~(k5_1=k5_2)
487 %   )).
488
489 % Sortality + Rigidity
490
491 fof(ax_rigidSortalsAreRigidAndSortal_xx, axiom, (
492   ![T]: ((rigidSortal(T)) <=> (rigid(T) & sortal(T)))
493   )).
494
495 fof(ax_antiRigidSortalsAreAntiRigidAndSortal_xx, axiom, (
496   ![T]: ((antiRigidSortal(T)) <=> (antiRigid(T) & sortal(T)))
497   )).
498
499 fof(ax_rigidNonSortalsAreRigidAndNonSortal_xx, axiom, (
500   ![T]: ((rigidNonSortal(T)) <=> (rigid(T) & nonSortal(T)))
501   )).
502
503 fof(ax_antiRigidNonSortalsAreAntiRigidAndNonSortal_xx, axiom, (
504   ![T]: ((antiRigidNonSortal(T)) <=> (antiRigid(T) & nonSortal(T)))
505   )).
506
507 fof(ax_semiRigidNonSortalsAreSemiRigidAndNonSortal_xx, axiom, (
508   ![T]: ((semiRigidNonSortal(T)) <=> (semiRigid(T) & nonSortal(T)))
509   )).
510
511 % If we have the taxonomy's axiomatization, then a25 becomes a
      theorem
512 % Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented
      for convenience
513
514 % fof(th_kindAndSubkindAreDisjoint_a25, conjecture, (
515 %   ~?[T]: (kind(T) & subkind(T))
516 %   )).
517
518 % If we have the taxonomy's axiomatization, then a26 becomes a
      theorem
519 % Ax |= "th_kindAndSubkindAreRigidSortals_a26"; conjecture
      commented for convenience
520
521 % fof(th_kindAndSubkindAreRigidSortals_a26, conjecture, (
522 %   ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))

```

```

523 %)).
524
525 % If we have the taxonomy's axiomatization, then a27 becomes a
    theorem
526 % Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for
    convenience
527
528 % fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
529 %   ~?[T]: (phase(T) & role(T))
530 %)).
531
532 % If we have the taxonomy's axiomatization, then a28 becomes a
    theorem
533 % Ax |= "th_phaseAndRoleAreAntiRigidSortals_a28"; conjecture
    commented for convenience
534
535 % fof(th_phaseAndRoleAreAntiRigidSortals_a28, conjecture, (
536 %   ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
537 %)).
538
539 % Skipping (a29) because we leave the concept of semi-rigid sortals
    out of this ontology.
540
541 % If we have the taxonomy's axiomatization, then a30 becomes a
    theorem
542 % Ax |= "th_categoriesAreRigidNonSortals_a30"; conjecture commented
    for convenience
543
544 % fof(th_categoriesAreRigidNonSortals_a30, conjecture, (
545 %   ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))
546 %)).
547
548 % If we have the taxonomy's axiomatization, then a31 becomes a
    theorem
549 % Ax |= "th_mixinsAreSemiRigidNonSortals_a31"; conjecture commented
    for convenience
550
551 % fof(th_mixinsAreSemiRigidNonSortals_a31, conjecture, (
552 %   ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
553 %)).
554
555 % If we have the taxonomy's axiomatization, then a32 becomes a
    theorem
556 % Ax |= "th_phaseMixinAndRoleMixinAreDisjoint_a32"; conjecture
    commented for convenience
557
558 % fof(th_phaseMixinAndRoleMixinAreDisjoint_a32, conjecture, (
559 %   ~?[T]: (phaseMixin(T) & roleMixin(T))
560 %)).
561
562 % If we have the taxonomy's axiomatization, then a33 becomes a
    theorem
563 % Ax |= "ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a33";
    conjecture commented for convenience
564
565 % fof(th_phaseMixinAndRoleMixinAreAntiRigidSortals_a33, conjecture,
    (

```

```

566 %   ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) &
567 %   nonSortal(T)))
568 %   )).
569 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture
      commented for convenience
570
571 % fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
572 %   ~?[T]: (endurantType(T) & (
573 %     (
574 %       (kind(T) & subkind(T))
575 %       | (kind(T) & phase(T))
576 %       | (kind(T) & role(T))
577 %       | (kind(T) & category(T))
578 %       | (kind(T) & mixin(T))
579 %       | (kind(T) & phaseMixin(T))
580 %       | (kind(T) & roleMixin(T))
581 %     ) | (
582 %       (subkind(T) & phase(T))
583 %       | (subkind(T) & role(T))
584 %       | (subkind(T) & category(T))
585 %       | (subkind(T) & mixin(T))
586 %       | (subkind(T) & phaseMixin(T))
587 %       | (subkind(T) & roleMixin(T))
588 %     ) | (
589 %       (phase(T) & role(T))
590 %       | (phase(T) & category(T))
591 %       | (phase(T) & mixin(T))
592 %       | (phase(T) & phaseMixin(T))
593 %       | (phase(T) & roleMixin(T))
594 %     ) | (
595 %       (role(T) & category(T))
596 %       | (role(T) & mixin(T))
597 %       | (role(T) & phaseMixin(T))
598 %       | (role(T) & roleMixin(T))
599 %     ) | (
600 %       (category(T) & mixin(T))
601 %       | (category(T) & phaseMixin(T))
602 %       | (category(T) & roleMixin(T))
603 %     ) | (
604 %       (mixin(T) & phaseMixin(T))
605 %       | (mixin(T) & roleMixin(T))
606 %     ) | (
607 %       (phaseMixin(T) & roleMixin(T))
608 %     )
609 %   ))
610 %   )).
611
612 % Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
      "; conjecture commented for convenience
613
614 % fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19,
      conjecture, (
615 %   ![T]: (endurantType(T) => (
616 %     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
617 %     mixin(T) | phaseMixin(T) | roleMixin(T)

```

```

618 %)).
619
620 % Sortality and rigidity properties combined
621 % (tested to rule out trivial models)
622
623 % fof(ax_sortalityAndRigidityInUse, axiom, (
624 %   enduring(e6_1) & enduring(e6_2) & world(w6) & kind(k6_1) & kind
625 %   (k6_2) & iof(e6_1,k6_1,w6) & iof(e6_1,k6_1,w6) & ~(k6_1=k6_2)

```

2.3.7 Defining Endurant Types

```

628
629 % Defining the taxonomy of types of ontological natures through the
630 % categorization of the taxonomy of concrete individuals
631
632 fof(ax_perduranTypeDefinition_a44, axiom, (
633   ![T]: (perduranType(T) <=> (
634     type_(T) & (![P,W]: ((world(W) & iof(P,T,W)) => (perduran(P)))
635   ))
636 ))
637
638 fof(ax_endurantTypeDefinition_a44, axiom, (
639   ![T]: (endurantType(T) <=> (
640     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (endurant(E))))
641 ))
642
643 fof(ax_substantialTypeDefinition_a44, axiom, (
644   ![T]: (substantialType(T) <=> (
645     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (substantial(E)
646   )))
647 ))
648
649 fof(ax_momentTypeDefinition_a44, axiom, (
650   ![T]: (momentType(T) <=> (
651     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (moment(E))))
652 ))
653 ))
654
655 fof(ax_objectTypeDefinition_a44, axiom, (
656   ![T]: (objectType(T) <=> (
657     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (object(E))))
658 ))
659 ))
660
661 fof(ax_collectiveTypeDefinition_a44, axiom, (
662   ![T]: (collectiveType(T) <=> (
663     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (collective(E)
664   )))
665 ))
666
667 fof(ax_quantityTypeDefinition_a44, axiom, (
668   ![T]: (quantityType(T) <=> (
669     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quantity(E))))

```

```

670 ))
671 )).
672
673 fof(ax_intrinsicMomentTypeDefinition_a44, axiom, (
674   ![T]: (intrinsicMomentType(T) <=> (
675     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (
676       intrinsicMoment(E))))
677   ))
678 )).
679
680 fof(ax_relatorTypeDefinition_a44, axiom, (
681   ![T]: (relatorType(T) <=> (
682     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (relator(E))))
683   ))
684 )).
685
686 fof(ax_qualityTypeDefinition_a44, axiom, (
687   ![T]: (qualityType(T) <=> (
688     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quality(E))))
689   ))
690 )).
691
692 fof(ax_modeTypeDefinition_a44, axiom, (
693   ![T]: (modeType(T) <=> (
694     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (mode(E))))
695   ))
696 )).
697
698 % Types Definition
699 % (tested to rule out trivial models)
700 % TODO: investigate why we cannot list four different endurant
701 % types (it may have something to do with "intrinsicMoment" and "
702 %   intrinsicMomentType")
703
704 % fof(ax_typesDefinitionsInstances, axiom, (
705 %   objectType(ot7) & collectiveType(ct7) & modeType(mt7)
706 % )).
707
708 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t21"; conjecture
709 % commented for convenience
710 % Having the previously defined taxonomy, this should be quite
711 % trivial
712
713 % fof(th_leafCategoriesArePairwiseDisjoint_t21, conjecture, (
714 %   ~?[T]: (type_(T) & (
715 %     (
716 %       (objectType(T) & collectiveType(T)) | (objectType(T) &
717 %         quantityType(T)) | (objectType(T) & modeType(T)) | (objectType(
718 %           T) & qualityType(T)) | (objectType(T) & relatorType(T)) | (
719 %             objectType(T) & perdurantType(T))
720 %     ) | (
721 %       (collectiveType(T) & quantityType(T)) | (collectiveType(T)
722 %         & modeType(T)) | (collectiveType(T) & qualityType(T)) | (
723 %           collectiveType(T) & relatorType(T)) | (collectiveType(T) &
724 %             perdurantType(T))
725 %     ) | (
726 %       (quantityType(T) & modeType(T)) | (quantityType(T) &

```

```

    qualityType(T)) | (quantityType(T) & relatorType(T)) | (
716 %    quantityType(T) & perdurantType(T))
717 %    ) | (
718 %    (modeType(T) & qualityType(T)) | (modeType(T) & relatorType
719 %    (T)) | (modeType(T) & perdurantType(T))
720 %    ) | (
721 %    relatorType(T) & perdurantType(T)
722 %    )
723 %    ))
724 %    )).
725
726 % Ultimate Sortals Definitions (by ontological nature)
727
728 fof(ax_objectKindDefinition_a45, axiom, (
729   ![T]: (objectKind(T) <=> (objectType(T) & kind(T)))
730 )).
731
732 fof(ax_collectiveKindDefinition_a45, axiom, (
733   ![T]: (collectiveKind(T) <=> (collectiveType(T) & kind(T)))
734 )).
735
736 fof(ax_quantityKindDefinition_a45, axiom, (
737   ![T]: (quantityKind(T) <=> (quantityType(T) & kind(T)))
738 )).
739
740 fof(ax_modeKindDefinition_a45, axiom, (
741   ![T]: (modeKind(T) <=> (modeType(T) & kind(T)))
742 )).
743
744 fof(ax_qualityKindDefinition_a45, axiom, (
745   ![T]: (qualityKind(T) <=> (qualityType(T) & kind(T)))
746 )).
747
748 fof(ax_relatorKindDefinition_a45, axiom, (
749   ![T]: (relatorKind(T) <=> (relatorType(T) & kind(T)))
750 )).
751
752 % Ultimate sortals (by ontological nature) instances
753 % (tested to rule out trivial models)
754 % TODO: investigate why we cannot list all different types of
       ultimate sortals at once
755
756 % fof(ax_typesDefinitionsInstances, axiom, (
757 %   objectKind(ok9) & collectiveKind(ck9) & quantityKind(quank9) &
       relatorKind(rk9) & modeKind(mk9) & qualityKind(qualk9)
758 % )).
759
760 % Skipping (t22) because (a21) makes it trivial
761
762 % Ax |= "th_endurantsInstantiateEndurantKindsOfSomeNature_a46";
       conjecture commented for convenience
763 % This axiom is actually a theorem in this version of the
       axiomatization
764

```

```

765 % fof(th_endurantsInstantiateEndurantKindsOfSomeNature_a46,
      conjecture, (
766 %   ![E]: (endurant(E) => (
767 %     ?[K,W]: ((objectKind(K) | collectiveKind(K) | quantityKind(K)
768 %       | modeKind(K) | qualityKind(K) | relatorKind(K))
769 %     & iof(E,K,W))
770 %   ))
771 % ))
772 % Ax |= "th_endurantSortalsCompleteness_t23"; conjecture commented
      for convenience
773 % Thanks to the taxonomy, we already have "sortal(T) =>
      endurantType(T)", but I leave it like this to be consistent
      with the paper
774
775 % fof(th_endurantSortalsCompleteness_t23, conjecture, (
776 %   ![T]: ((endurantType(T) & sortal(T)) => (objectKind(T) |
      collectiveKind(T) | quantityKind(T) | qualityKind(T) | modeKind
      (T) | relatorKind(T) | phase(T) | role(T)))
777 % ))
778
779 % Ax |= "th_objectTypesSpecializeAKindOfSameNature_t24"; conjecture
      commented for convenience
780
781 % fof(th_objectTypesSpecializeAKindOfSameNature_t24, conjecture, (
782 %   ![T]: ((objectType(T) & sortal(T)) <=> (?[K]: (objectKind(K) &
      specializes(T,K))))
783 % ))
784
785 % Ax |= "th_collectiveTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
786
787 % fof(th_collectiveTypesSpecializeAKindOfSameNature_t24, conjecture
      , (
788 %   ![T]: ((collectiveType(T) & sortal(T)) <=> (?[K]: (
      collectiveKind(K) & specializes(T,K))))
789 % ))
790
791 % Ax |= "th_quantityTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
792
793 % fof(th_quantityTypesSpecializeAKindOfSameNature_t24, conjecture,
      (
794 %   ![T]: ((quantityType(T) & sortal(T)) <=> (?[K]: (quantityKind(K)
      ) & specializes(T,K))))
795 % ))
796
797 % Ax |= "th_modeTypesSpecializeAKindOfSameNature_t24"; conjecture
      commented for convenience
798
799 % fof(th_modeTypesSpecializeAKindOfSameNature_t24, conjecture, (
800 %   ![T]: ((modeType(T) & sortal(T)) <=> (?[K]: (modeKind(K) &
      specializes(T,K))))
801 % ))
802
803 % Ax |= "th_qualityTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience

```

```

804
805 % fof(th_qualityTypesSpecializeAKindOfSameNature_t24, conjecture, (
806 %   ![T]: ((qualityType(T) & sortal(T)) <=> (?[K]: (qualityKind(K)
      & specializes(T,K))))
807 % )).
808
809 % Ax |= "th_relatorTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
810
811 % fof(th_relatorTypesSpecializeAKindOfSameNature_t24, conjecture, (
812 %   ![T]: ((relatorType(T) & sortal(T)) <=> (?[K]: (relatorKind(K)
      & specializes(T,K))))
813 % )).
814
815 % Ax |= "th_sortalLeafCategoriesAreDisjoint_t25"; conjecture
      commented for convenience
816
817 % fof(th_sortalLeafCategoriesAreDisjoint_t25, conjecture, (
818 %   ![T]: (objectKind(T) => (~ (collectiveKind(T) | quantityKind(T)
      | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
      mixin(T) | phaseMixin(T) | roleMixin(T))))
819 %   & ![T]: (collectiveKind(T) => (~ (objectKind(T) | quantityKind(T)
      | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
820 %   & ![T]: (quantityKind(T) => (~ (objectKind(T) | collectiveKind(T)
      | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
821 %   & ![T]: (modeKind(T) => (~ (objectKind(T) | quantityKind(T) |
      collectiveKind(T) | qualityKind(T) | relatorKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
822 %   & ![T]: (qualityKind(T) => (~ (objectKind(T) | quantityKind(T) |
      modeKind(T) | collectiveKind(T) | relatorKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
823 %   & ![T]: (relatorKind(T) => (~ (objectKind(T) | quantityKind(T) |
      modeKind(T) | qualityKind(T) | collectiveKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
824 %   & ![T]: (category(T) => (~ (objectKind(T) | quantityKind(T) |
      modeKind(T) | qualityKind(T) | relatorKind(T) | collectiveKind(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
825 %   & ![T]: (mixin(T) => (~ (objectKind(T) | quantityKind(T) |
      modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
      collectiveKind(T) | phaseMixin(T) | roleMixin(T))))
826 %   & ![T]: (phaseMixin(T) => (~ (objectKind(T) | quantityKind(T) |
      modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
      mixin(T) | collectiveKind(T) | roleMixin(T))))
827 %   & ![T]: (roleMixin(T) => (~ (objectKind(T) | quantityKind(T) |
      modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
      mixin(T) | phaseMixin(T) | collectiveKind(T))))
828 % )).
829
830 % Ax |= "th_sortalLeafCategoriesAreComplete_t26"; conjecture
      commented for convenience
831
832 % fof(th_sortalLeafCategoriesAreComplete_t26, conjecture, (
833 %   ![T]: ((endurantType(T)) => (objectKind(T) | collectiveKind(T)
      | quantityKind(T) | qualityKind(T) | modeKind(T) | relatorKind(T)
      | phase(T) | role(T) | category(T) | mixin(T) | phaseMixin(T)

```



```

    ) | roleMixin(T)))
834 % )).

```

2.3.8 Mereology

```

837
838 % TODO: review whether it is necessary to reduce mereology to
      concrete individuals; I am leaving this axiom out for the
      moment
839
840 % fof(ax_partArguments, axiom, (
841 %   ![X,Y]: (part(X,Y) => (concreteIndividual(X) &
      concreteIndividual(Y)))
842 % )).
843
844 fof(ax_reflexiveParthood, axiom, (
845   ![X]: (partOf(X,X))
846 )).
847
848 fof(ax_antiSymmetricParthood_a47, axiom, (
849   ![X,Y]: ((partOf(X,Y) & partOf(Y,X)) => (X=Y))
850 )).
851
852 fof(ax_antiSymmetricParthood_a48, axiom, (
853   ![X,Y]: ((partOf(X,Y) & partOf(Y,X)) => (X=Y))
854 )).
855
856 fof(ax_transitiveParthood_a49, axiom, (
857   ![X,Y,Z]: ((partOf(X,Y) & partOf(Y,Z)) => (partOf(X,Z)))
858 )).
859
860 fof(ax_overlappingWholes_a50, axiom, (
861   ![X,Y]: ((overlap(X,Y)) <=> (?[Z]: (partOf(Z,X) & partOf(Z,Y))))
862 )).
863
864 fof(ax_strongSupplementation_a51, axiom, (
865   ![X,Y]: (~partOf(X,Y) <=> ?[Z]: (partOf(Z,X) & ~overlap(Z,Y)))
866 )).
867
868 fof(ax_properPart_a52, axiom, (
869   ![X,Y]: (properPartOf(X,Y) <=> (partOf(X,Y) & ~partOf(Y,X)))
870 )).
871
872 fof(ax_binarySum_a53, axiom, (
873   ![X,Y,Z]: (sum(Z,X,Y) <=> ![W]: (overlap(W,Z) <=> (overlap(W,X) |
      overlap(W,Y))))
874 )).
875
876 % Mereology in use
877 % (tested to rule out trivial models)
878
879 % fof(ax_mereologyInUse, axiom, (
880 %   concreteIndividual(ci10_1) & concreteIndividual(ci10_2) &
      concreteIndividual(ci10_3) & concreteIndividual(ci10_4) &
      concreteIndividual(ci10_5) & ~(ci10_1=ci10_2) & ~(ci10_2=ci10_3)
      & ~(ci10_3=ci10_4) & ~(ci10_4=ci10_5) & properPart(ci10_1,
      ci10_2) & properPart(ci10_3,ci10_4) & sum(ci10_5,ci10_3,ci10_4)
881 % )).

```

```

882
883 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Composition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
884
885 fof(ax_function, axiom, (
886   ![X,Y]: (functionsAs(X,Y) => (endurant(X) & type_(Y)))
887 )).

```

2.3.9 Composition

```

886   ![X,Y]: (functionsAs(X,Y) => (endurant(X) & type_(Y)))
887 )).
888
889 fof(ax_genericFunctionalDependence_a55, axiom, (
890   ![T1,T2,W]: (gfd(T1,T2,W) <=>
891     ![E1]: ((iof(T1,E1,W) & functionsAs(T1,E1)) => ?[E2]: (~ (E1=E2)
892       & iof(T2,E2,W) & functionsAs(T2,E2))))
893 ))).
894
895 fof(ax_individualFunctionalDependence_a56, axiom, (
896   ![E1,T1,E2,T2,W]: (ifd(E1,T1,E2,T2,W) <=> (
897     gfd(T1,T2,W) & iof(E1,T1,W) & iof(E2,T2,W) & (functionsAs(E1,T1)
898       ) => functionsAs(E2,T2))
899 ))).
900
901 fof(ax_componentOf_a57, axiom, (
902   ![E1,T1,E2,T2,W]: (componentOf(E1,T1,E2,T2,W) <=> (properPartOf(
903     E1,E2) & ifd(E1,T1,E2,T2,W)))
904 ))).
905
906 % Composition in use
907 % (tested to rule out trivial models)
908
909 % fof(ax_compositionInUse, axiom, (
910 %   componentOf(e11_1,t11_1,e11_2,t11_2,w11) & ~(e11_1=e11_2) & ~(
911 %     e11_1=t11_1) & ~(e11_2=t11_2) & ~(e11_1=t11_2) & ~(e11_2=t11_1)
912 %   & ~(t11_1=t11_2)
913 % )).
914
915 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Constitution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
916
917 fof(ax_constitutedByInvolvedNatures_a58, axiom, (
918   ![X,Y,W]: (constitutedBy(X,Y,W) => ((endurant(X) <=> endurant(Y))
919     & (perdurant(X) <=> perdurant(Y)) & world(W)))
920 ))).
921
922 % Ax |= "th_noSelfConstitution_t27"; conjecture commented for
923 % convenience
924
925 % fof(th_noSelfConstitution_t27, conjecture, (

```

2.3.10 Constitution

```

916
917 fof(ax_constitutedByDifferentKinds_a59, axiom, (
918   ![E1,E2,T1,T2,W]: ((constitutedBy(E1,E2,W) & iof(E1,T1,W) & iof(
919     E2,T2,W) & kind(T1) & kind(T2)) => (~ (T1=T2)))
920 ))).
921
922 % Ax |= "th_noSelfConstitution_t27"; conjecture commented for
923 % convenience
924
925 % fof(th_noSelfConstitution_t27, conjecture, (

```

```

924 % ~?[X,W]: (endurant(X) & constitutedBy(X,X,W))
925 % )).
926
927 fof(ax_genericConstitutionalDependence_a60, axiom, (
928   ![T1,T2]: (genericConstitutionalDependence(T1,T2) <=> (
929     type_(T1) & type_(T2) & ![E1,W]: (iof(E1,T1,W) => (
930       ?[E2]: (constitutedBy(E1,E2,W) & iof(E2,T2,W)
931     )))
932   ))
933 )).
934
935 fof(ax_constitution_a61, axiom, (
936   ![E1,T1,E2,T2,W]: (constitution(E1,T1,E2,T2,W) <=> (
937     iof(E1,T1,W) & iof(E2,T2,W) & genericConstitutionalDependence(
938       T1,T2) & constitutedBy(E1,E2,W)
939   ))
940 )).
941
942 fof(
943   ax_wheneverAConstitutedPerdurantExistsTheConstitutedByRelationHolds_a62
944   , axiom, (
945     ![P1,P2,W1]: ((constitutedBy(P1,P2,W1) & perdurant(P1)) => (![W2
946       ]: (exists(P1,W2) => constitutedBy(P1,P2,W2))))
947   )).
948
949 fof(ax_constitutedByIsAsymmetric_a63, axiom, (
950   ![E1,E2,W]: (constitutedBy(E1,E2,W) => ~constitutedBy(E2,E1,W))
951 )).
952
953 % Constitution in use
954 % (tested to rule out trivial models)
955
956 % fof(ax_constitutionInUse, axiom, (
957 %   object(e12_1) & object(e12_2) & objectKind(k12_1) & objectKind(
958 %     k12_2) & world(w12) & ~(k12_1=k12_2) & iof(e12_1,k12_1,w12) &
959 %     iof(e12_2,k12_2,w12) & constitutedBy(e12_1,e12_2,w12) &
960 %     genericConstitutionalDependence(k12_1,k12_2) & constitution(
961 %       e12_1,k12_1,e12_2,k12_2,w12)
962 % )).
963
964 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Existential Dependence %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
965
966 fof(ax_exists_a64, axiom, (
967   ![X,W]: (exists(X,W) => (thing(X) & world(W)))
968 )).

```

2.3.11 Existential Dependence

```

961
962 fof(ax_existentiallyDependsOn_a65, axiom, (
963   ![X,Y]: (existentiallyDependsOn(X,Y) <=> (![W]: (exists(X,W) =>
964     exists(Y,W))))
965 )).
966
967 fof(ax_existentiallyIndependentOf_a66, axiom, (
968   ![X,Y]: (existentiallyIndependentOf(X,Y) <=> (~
969     existentiallyDependsOn(X,Y) & ~existentiallyDependsOn(Y,X)))
970 )).

```

```

970 % Existential dependence in use
971 % (tested to rule out trivial models)
972
973 % fof(ax_constitutionInUse, axiom, (
974 %   object(e13_1) & object(e13_2) & object(e13_3) & ~(e13_1=e13_2)
975 %   & ~(e13_1=e13_3) & ~(e13_2=e13_3) & existentiallyDependsOn(
976 %     e13_2,e13_1) & existentiallyIndependentOf(e13_3,e13_1)
977 % )).
978
979 % TODO: introduce transitivity and anti-symmetry of existential
980 %       dependence
981 % TODO: introduce continuity of existence with perdurants never
982 %       ceasing to exist
983
984 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Moments and Inherence %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
985
986 % Inherence

```

2.3.12 Moments and Inherence

```

985   ![M,X]: (inheresIn(M,X) => existentiallyDependsOn(M,X))
986   )).
987
988 fof(ax_thingsInvolvedInInherence_a68, axiom, (
989   ![M,X]: (inheresIn(M,X) => (moment(M) & (type_(X) | enduring(X)))
990   )
991   )).
992
993 % TODO: add definition (d5) for the "bearer" axiom
994
995 fof(ax_irreflexiveInherence, axiom, (
996   ![X]: (~inheresIn(X,X))
997   )).
998
999 fof(ax_asymmetricInherence, axiom, (
1000   ![X,Y]: (inheresIn(X,Y) => ~inheresIn(Y,X))
1001   )).
1002
1003 fof(ax_intransitiveInherence, axiom, (
1004   ![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(Y,Z)) => ~inheresIn(X,Z))
1005   )).
1006
1007 fof(ax_uniqueInherence_a69, axiom, (
1008   ![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(X,Z)) => (Y=Z))
1009   )).
1010
1011 % Moments
1012
1013 fof(ax_dMomentOf_d6, axiom, (
1014   ![M,X]: (momentOf(M,X) <=> (inheresIn(M,X) | (
1015     ?[M2]: (inheresIn(M,M2) & momentOf(M2,X))
1016   )))
1017   )).
1018
1019 fof(ax_dUltimateBearerOf_d7, axiom, (
1020   ![B,M]: (ultimateBearerOf(B,M) <=> (~moment(B) & momentOf(M,B)))
1021   )).

```

```

1022 fof(ax_everyMomentHasUniqueAUltimateBearer_a70, axiom, (
1023   ![M]: (moment(M) => (?[B]: (ultimateBearerOf(B,M) & (
1024     ![B2]: (ultimateBearerOf(B2,M) <=> (B=B2))
1025   ))))
1026)).
1027
1028 fof(ax_noMomentOfCycles, axiom, (
1029   ~?[M]: momentOf(M,M)
1030)).
1031
1032 % Ax |= "th_irreflexiveInherence_t28"; conjecture commented for
      convenience
1033
1034 % fof(th_irreflexiveInherence_t28, conjecture, (
1035 %   ~?[X]: (inheresIn(X,X))
1036 %)).
1037
1038 % Ax |= "th_asymmetricInherence_t29"; conjecture commented for
      convenience
1039
1040 % fof(th_asymmetricInherence_t29, conjecture, (
1041 %   ~?[X,Y]: (inheresIn(X,Y) & inheresIn(Y,X))
1042 %)).
1043
1044 % Ax |= "th_antiTransitiveInherence_t30"; conjecture commented for
      convenience
1045
1046 % fof(th_antiTransitiveInherence_t30, conjecture, (
1047 %   ![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(Y,Z)) => (~inheresIn(X,Z)
1048 %   )))
1049 %)).
1050 % TODO: add instances
1051
1052 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Relators %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1053
1054 % External Dependence and Externally Dependent Modes

```

2.3.13 Relators

```

1057 ~?[M,X]: (externallyDependsOn(M,X) <=> (existentiallyDependsOn(M,
      X) & (![Y]: (inheresIn(M,Y) => existentiallyIndependentOf(X,Y))
1058 )))
1059)).
1060
1061 fof(ax_dExternallyDependentMode_a72, axiom, (
1062   ![M]: (externallyDependentMode(M) <=> (mode(M) & (?[X]: (
      externallyDependsOn(M,X))))))
1063)).
1064
1065 % Founded by
1066
1067 fof(ax_foundedByInvolvedThings_a73, axiom, (
1068   ![M,P]: (foundedBy(M,P) <=> ((externallyDependentMode(M) |
      relator(M)) & perdurant(P)))
1069)).
1070
1071 fof(ax_relationalModesHaveAFoundationEvent_a74, axiom, (

```

```

1071 ! [M]: ((externallyDependentMode(M) | relator(M)) => (?[P]: (
    foundedBy(M,P))))
1072 )).
1073
1074 fof(ax_uniqueFoundationEvents_a74, axiom, (
1075 ! [M,P1,P2]: ((foundedBy(M,P1) & foundedBy(M,P2)) => (P1=P2))
1076 )).
1077
1078 % TODO: add definition (d8) for the "foundationOf" axiom
1079
1080 % Qua Individual
1081
1082 fof(ax_dQuaIndividualOf_a75, axiom, (
1083 ! [X,Y]: (quaIndividualOf(X,Y) <=> (![Z]: (overlap(Z,X) <=> (
1084     externallyDependentMode(Z) & inheresIn(Z,Y) & (![P]: (foundedBy
1085         (X,P) => foundedBy(Z,P))))
1086 ))).
1087
1088 % Ax |= "
    th_thePartsOfAQuaIndividualShareTheFoundationOfTheWhole_t31";
    conjecture commented for convenience
1089
1090 % fof(th_thePartsOfAQuaIndividualShareTheFoundationOfTheWhole_t31,
    conjecture, (
1091 % ! [X,Y,Z]: ((quaIndividual(X) & partOf(Z,X)) => (![P]: (
    foundedBy(Z,P) => foundedBy(X,P))))
1092 % )).
1093
1094 fof(ax_dQuaIndividual_a76, axiom, (
1095 ! [X]: (quaIndividual(X) <=> ?[Y]: (quaIndividualOf(X,Y)))
1096 )).
1097
1098 % Qua Individual is already defined as a subtype of Externally
    Dependent Mode in the taxonomy; skipping (a78)
1099
1100 % Skipping (a79); already defined in (a74)
1101
1102 fof(ax_thePartsOfARelatorShareTheFoundationOfTheWhole_a80, axiom, (
1103 ! [X,Y,Z]: ((relator(X) & partOf(Z,X)) => (![P]: (foundedBy(Z,P)
    => foundedBy(X,P))))
1104 )).
1105
1106 fof(ax_dRelator_a81, axiom, (
1107 ! [R]: (relator(R) <=> (
1108     (?[X]: (properPartOf(X,R))
1109     & (![Y,Z]: ((properPartOf(Y,R) & properPartOf(Z,R)) => (
    quaIndividual(Y) & quaIndividual(Z) & existentiallyDependsOn(Y,
    Z) & existentiallyDependsOn(Z,Y) & (![P]: (foundedBy(Y,P) <=>
    foundedBy(Z,P))))))
1110     & (![Y2,Z2]: ((properPartOf(Y2,R) & quaIndividual(Z2) &
    existentiallyDependsOn(Y2,Z2) & existentiallyDependsOn(Z2,Y2) &
    (![P2]: (foundedBy(Y2,P2) <=> foundedBy(Z2,P2)))) => (
    properPartOf(Z2,R))))
1111 ))).
1112 )).
1113

```

```

1114 % Ax |= "th_relatorsImplyTheExistenceOfAtLeastTwoQualIndividuals_t32
      "; conjecture commented for convenience
1115
1116 % fof(th_relatorsImplyTheExistenceOfAtLeastTwoQualIndividuals_t32,
      conjecture, (
1117 %   ![R]: (relator(R) => (?[Q1,X,Q2,Y]: (qualIndividualOf(Q1,X) &
      qualIndividualOf(Q2,Y) & ~(Q1=Q2))))
1118 % )).
1119
1120 fof(ax_dMediates_a82, axiom, (
1121 %   ![R,E]: (mediates(R,E) <=> (relator(R) & endurant(E) & (?[Q]: (
      qualIndividualOf(Q,E) & partOf(Q,R))))
1122 % )).
1123
1124 % Ax |= "th_relatorsMediateAtLeastTwoThings_t33"; conjecture
      commented for convenience
1125
1126 % fof(th_relatorsMediateAtLeastTwoThings_t33, conjecture, (
1127 %   ![R]: (relator(R) => (?[E1,E2]: ~(E1=E2) & mediates(R,E1) &
      mediates(R,E2))))
1128 % )).
1129
1130 % TODO: add definition (d9) for the "relator bearer" axiom
1131
1132 % TODO: add instances
1133
1134 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Characterization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1135
1136 fof(ax_endurantTypeCharacterizationByMomentTypes_a83, axiom, (
1137 %   ![ET,MT]: (characterizes(MT,ET) => (

```

2.3.14 Characterization

```

1139 %   & momentType(M)
1140 %   & (![E,W]: (iof(E,ET,W) => (?[M]: (iof(M,MT,W) & inheresIn(M,E)
      ))))
1141 %   & (![M2,W2]: (iof(M2,MT,W2) => (?[E2]: (iof(E2,ET,W2) &
      inheresIn(M2,E2))))))
1142 % )).
1143 % )).
1144
1145 % Ax |= "
      th_qualitiesInheresInAUniqueEndurantConnectThroughCharacterization_a84
      "; conjecture commented for convenience
1146
1147 % fof(
      th_qualitiesInheresInAUniqueEndurantConnectThroughCharacterization_a84
      , conjecture, (
1148 %   ![QT,ET]: ((characterizes(QT,ET) & qualityType(QT)) => (![Q,W]:
      (iof(Q,QT,W) => (?[E]: (iof(E,ET,W) & inheresIn(Q,E) & (![E2]:
      (inheresIn(Q,E2) <=> (E=E2))))))))))
1149 % )).
1150
1151 % TODO: add instances
1152
1153 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Qualities and Quality Structures %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1154
1155 % Skipping (a85); previously introduced in the taxonomy

```

```
1156 % Skipping (a86); previously introduced in the taxonomy
```

2.3.15 Qualities and Quality Structures

```
1162
1163 % Quality Structures
1164
1165 fof(ax_dQualityStructure_d10, axiom, (
1166   ![QS]: (qualityStructure(QS) <=> (?[QT]: (qualityType(QT) &
1167     associatedWith(QS,QT))))
1168 ))).
1169
1170 fof(ax_dQualityStructure_d10, axiom, (
1171   ![QS]: (qualityStructure(QS) <=> (?[QT]: (qualityType(QT) &
1172     associatedWith(QS,QT))))
1173 ))).
1174
1175 %%%%%%%%%%%%% Endurants and Perdurants %%%%%%%%%%%%%
1176
1177 fof(ax_manifestsInvolvedThings_a104, axiom, (
1178   ![E,P]: (manifests(E,P) => (endurant(E) & perdurant(P)))
1179 ))).
1180
1181 fof(ax_lifeOfInvolvedThings_a105, axiom, (
1182   ![E,P]: (lifeOf(P,E) => (
```

2.3.16 Endurants and Perdurants

```
1182     & (![P2]: (overlap(P2,P) <=> (perdurant(P2) & manifests(E,P2)))
1183   )
1184 ))).
1185
1186 % TODO: review ax_lifeOfInvolvedThings_a105 and its translation of
1187   the small sigma predicate schema in (a105)
1188
1189 fof(ax_meetsInvolvedThings_a106, axiom, (
1190   ![P1,P2]: (meets(P1,P2) => (perdurant(P1) & perdurant(P2)))
1191 ))).
1192
1193 % TODO: add instances
```