

A TPTP Formalization of the Unified Foundational Ontology

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

October 11, 2021

Abstract

This document presents a formalization of the Unified Foundation Ontology (UFO) expressed in first-order logics through the TPTP syntax. This formalization is intended to support verification of UFO's theory through automated provers and consistency checkers.

1 Introduction

This document presents a formalization of the Unified Foundation Ontology (UFO) expressed in first-order logics through the TPTP syntax. This formalization is intended to support verification of UFO's theory through automated provers and consistency checkers.

2 UFO's TPTP Specification

2.1 UFO Taxonomy

2.1.1 Partial Taxonomy of Thing

```
4 % Thing
5
6 fof(ax_thing_taxonomy, axiom, (
7   ![X]: ((type_(X) | individual(X)) <=> (thing(X)))
8 )).
9
10 fof(ax_thing_partition, axiom, (
11   ~?[X]: (type_(X) & individual(X))
12 )).
13
14 % Individual
15
16 fof(ax_individual_taxonomy, axiom, (
17   ![X]: ((concreteIndividual(X) | abstractIndividual(X)) <=> (
18     individual(X)))
```

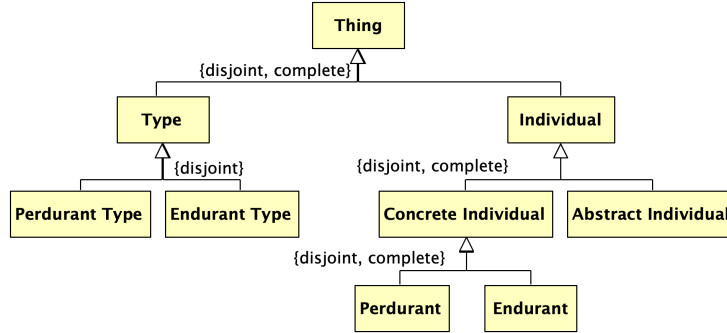


Figure 1: Partial Taxonomy of UFO – Thing.

```

18 ))).
19
20 fof(ax_individual_partition, axiom, (
21   ~?[X]: (concreteIndividual(X) & abstractIndividual(X))
22 ))).
23
24 % Concrete Individual
25
26 fof(ax_concreteIndividual_taxonomy, axiom, (
27   ![X]: ((endurant(X) | perdurant(X)) <=> (concreteIndividual(X)))
28 ))).
29
30 fof(ax_concreteIndividual_partition, axiom, (
31   ~?[X]: (endurant(X) & perdurant(X))
32 ))).
33
34 % Type
35
36 fof(ax_type_taxonomy, axiom, (
37   ![X]: ((endurantType(X) | perdurantType(X)) <=> (type_(X)))
38 ))).
39
40 fof(ax_type_partition, axiom, (
41   ~?[X]: (endurantType(X) & perdurantType(X))
42 ))).
43
44 % Thing partial taxonomy instances
45 % (tested rule out trivial models)
46
47 % fof(ax_thing_instances, axiom, (
48 %   type_(type1) & individual(individual1) & concreteIndividual(
49 %     concreteIndividual1) & abstractIndividual(abstractIndividual1)
50 %     & endurant(endurant1) & perdurant(perdurant1) & endurantType(
51 %       endurantType1) & perdurantType(perdurantType1)
52 %   ))).

```

2.1.2 Partial Taxonomy of Abstract Individual

```

51 % Abstract Individual
52

```

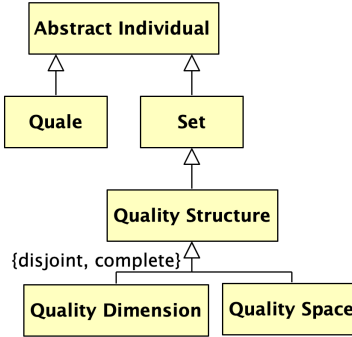


Figure 2: Partial Taxonomy of UFO – Abstract Individual.

```

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 % Set
62
63 fof(ax_set_taxonomy_qualityStructure, axiom, (
64   ![X]: (qualityStructure(X) => (set_(X)))
65 )).
66
67 % Quality Structure
68
69 fof(ax_qualityStructure_taxonomy, axiom, (
70   ![X]: ((qualityDimension(X) | qualitySpace(X)) <=> (
71     qualityStructure(X)))
72 )).
73
74 fof(ax_qualityStructure_partition, axiom, (
75   ~?[X]: (qualityDimension(X) & qualitySpace(X))
76 )).
77
78 % TODO: review the definition of "world" as a subtype of "
79   qualityStructure"
80
81 fof(ax_qualityStructure_taxonomy_world, axiom, (
82   ![X]: (world(X) => (qualityStructure(X)))
83 )).
84
85 % Abstract Individual partial taxonomy instances
86 % (tested rule out trivial models)
87
88 fof(ax_abstractIndividual_instances, axiom, (
89   set_(set1) & quale(quale1) & qualityStructure(qualityStructure1)
90   & qualityDimension(qualityDimension1) & qualitySpace(
91     qualitySpace1) & world(world1)

```

88 %)).

2.1.3 Partial Taxonomy of Endurant

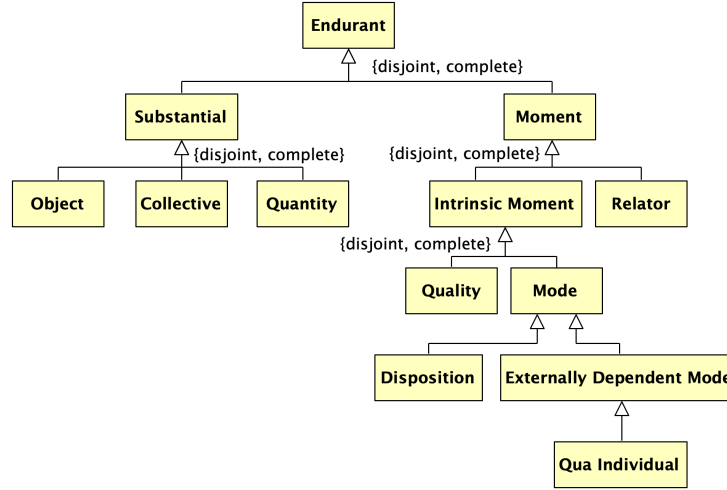


Figure 3: Partial Taxonomy of UFO – Endurant.

```

90 % Endurant
91
92 fof(ax_endurant_taxonomy, axiom, (
93   ![X]: ((substantial(X) | moment(X)) <=> (endurant(X)))
94 ))).
95
96 fof(ax_endurant_partition, axiom, (
97   ~?[X]: (substantial(X) & moment(X))
98 ))).
99
100 % Substantial
101
102 fof(ax_substantial_taxonomy, axiom, (
103   ![X]: ((object(X) | collective(X) | quantity(X)) <=> (substantial
104     (X)))
105 ))).
106
107 fof(ax_substantial_partition, axiom, (
108   ~?[X]: ((object(X) & collective(X)) | (object(X) & quantity(X)) |
109     (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, (

```

```

117 ~?[X]: (intrinsicMoment(X) & relator(X))
118 )).
119
120 % Intrinsic Moment
121
122 fof(ax_intrinsicMoment_taxonomy, axiom, (
123   ![X]: ((quality(X) | mode(X)) <=> (intrinsicMoment(X)))
124 )).
125
126 fof(ax_intrinsicMoment_partition, axiom, (
127   ~?[X]: (quality(X) & mode(X))
128 )).
129
130 % Mode
131
132 fof(ax_mode_taxonomy_externallyDependentMode, axiom, (
133   ![X]: (externallyDependentMode(X) => (mode(X)))
134 )).
135
136 % Externally Dependent Mode
137
138 fof(ax_externallyDependentMode_taxonomy_quaIndividual, axiom, (
139   ![X]: (quaIndividual(X) => (externallyDependentMode(X)))
140 )).
141
142 % Endurant partial taxonomy instances
143 % (tested rule out trivial models)
144
145 % fof(ax_endurant_instances, axiom, (
146 %   substantial(substantial1) & moment(moment1) & object(object1) &
147 %     collective(collective1) & quantity(quantity1) &
148 %       intrinsicMoment(intrinsicMoment1) & relator(relator1) & quality
149 %         (quality1) & mode(mode1) & disposition(disposition1) &
150 %           externallyDependentMode(externallyDependentMode1) &
151 %             quaIndividual(quaIndividual1)
152 % )).

```

2.1.4 Partial Taxonomy of Endurant Type (on ontological natures)

```

149 % Endurant Type (by ontological nature)
150
151 fof(ax_endurantType_taxonomy_nature, axiom, (
152   ![X]: ((substantialType(X) | momentType(X)) <=> (endurantType(X))
153   )
154 )).
155
156 fof(ax_endurantType_partition_nature, axiom, (
157   ~?[X]: (substantialType(X) & momentType(X))
158 )).
159
160 % Substantial Type
161
162 fof(ax_substantialType_taxonomy, axiom, (
163   ![X]: ((objectType(X) | collectiveType(X) | quantityType(X)) <=>
164     (substantialType(X)))
165 )).

```

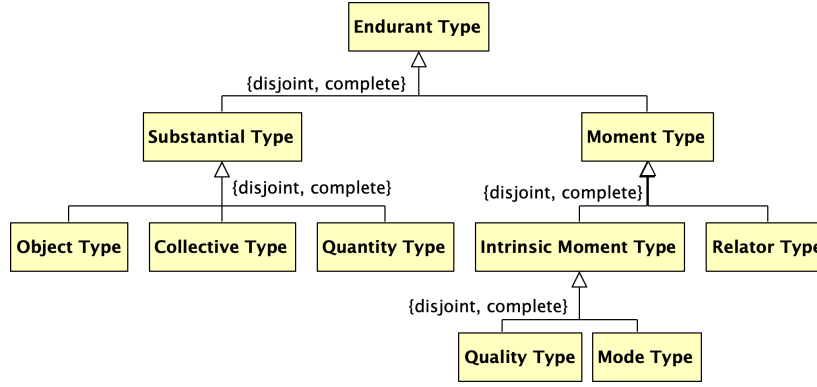


Figure 4: Partial Taxonomy of UFO – Endurant Types (by ontological nature).

```

164
165 fof(ax_substantialType_partition, axiom, (
166   ~?[X]: ((objectType(X) & collectiveType(X)) | (objectType(X) &
167     quantityType(X)) | (collectiveType(X) & quantityType(X)))
168 ))).
169 % Moment Type
170
171 fof(ax_momentType_taxonomy, axiom, (
172   ![X]: ((intrinsicMomentType(X) | relatorType(X)) <=> (momentType(X)
173     X)))
174 ))).
175 fof(ax_momentType_partition, axiom, (
176   ~?[X]: (intrinsicMomentType(X) & relatorType(X))
177 ))).
178
179 % Intrinsic Moment Type
180
181 fof(ax_intrinsicMomentType_taxonomy, axiom, (
182   ![X]: ((qualityType(X) | modeType(X)) <=> (intrinsicMomentType(X)
183     X)))
184 ))).
185 fof(ax_intrinsicMomentType_partition, axiom, (
186   ~?[X]: (qualityType(X) & modeType(X))
187 ))).
188
189 % Endurant Type (by ontological nature) partial taxonomy instances
190 % (tested rule out trivial models)
191
192 % fof(ax_endurantType_instances_natures, axiom, (
193 %   substantialType(substantialType1) & momentType(momentType1) &
194 %   objectType(objectType1) & collectiveType(collectiveType1) &
195 %   quantityType(quantityType1) & intrinsicMomentType(
196 %     intrinsicMomentType1) & relatorType(relatorType1) & qualityType(
197 %     qualityType1) & modeType(modeType1)

```

194 %))).

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

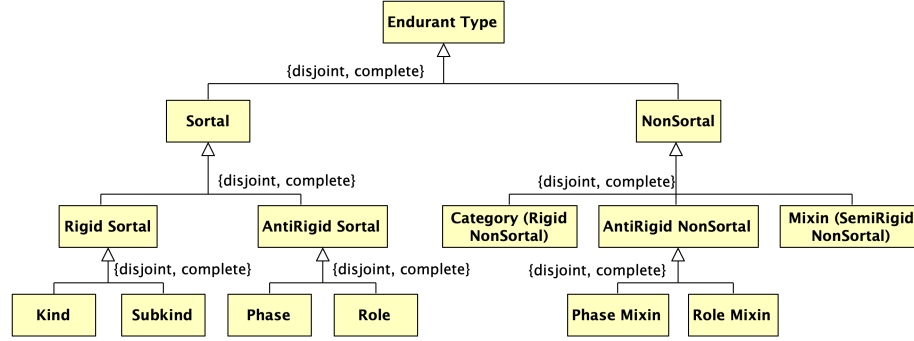


Figure 5: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```

196 % Endurant Type (by modal properties of types)
197
198 fof(ax_endurantType_taxonomy_properties, axiom, (
199   ![X]: ((sortal(X) | nonSortal(X)) <=> (endurantType(X)))
200 )).
201
202 fof(ax_endurantType_partition_properties, axiom, (
203   ~?[X]: (sortal(X) & nonSortal(X))
204 )).
205
206 % Sortal
207
208 fof(ax_sortal_taxonomy, axiom, (
209   ![X]: ((rigidSortal(X) | antiRigidSortal(X)) <=> (sortal(X)))
210 )).
211
212 fof(ax_sortal_partition, axiom, (
213   ~?[X]: (rigidSortal(X) & antiRigidSortal(X))
214 )).
215
216 % Rigid Sortal
217
218 fof(ax_rigidSortal_taxonomy, axiom, (
219   ![X]: ((kind(X) | subkind(X)) <=> (rigidSortal(X)))
220 )).
221
222 fof(ax_rigidSortal_partition, axiom, (
223   ~?[X]: (kind(X) & subkind(X))
224 )).
225
226 % Anti-Rigid Sortal
227
228 fof(ax_antiRigidSortal_taxonomy, axiom, (

```

```

229 ![X]: ((phase(X) | role(X)) <=> (antiRigidSortal(X)))
230 )).
231
232 fof(ax_antiRigidSortal_partition, axiom, (
233   ~?[X]: (phase(X) & role(X))
234 )).
235
236 % Non-Sortal
237
238 fof(ax_nonSortal_taxonomy, axiom, (
239   ![X]: ((rigidNonSortal(X) | semiRigidNonSortal(X) |
240     antiRigidNonSortal(X)) <=> (nonSortal(X)))
241 )).
242
243 fof(ax_nonSortal_partition, axiom, (
244   ~?[X]: ((rigidNonSortal(X) & semiRigidNonSortal(X)) | (
245     rigidNonSortal(X) & antiRigidNonSortal(X)) | (
246     semiRigidNonSortal(X) & antiRigidNonSortal(X)))
247 )).
248
249 % Category
250
251 fof(ax_rigidNonSortal_taxonomy, axiom, (
252   ![X]: (rigidNonSortal(X) <=> (category(X)))
253 )).
254
255 % Mixin
256
257 fof(ax_semiRigidNonSortal_taxonomy, axiom, (
258   ![X]: (semiRigidNonSortal(X) <=> (mixin(X)))
259 )).
260
261 % Anti-Rigid Non-Sortal
262
263 fof(ax_antiRigidNonSortal_taxonomy, axiom, (
264   ![X]: ((phaseMixin(X) | roleMixin(X)) <=> (antiRigidNonSortal(X))
265     )
266 )).
267
268 fof(ax_antiRigidNonSortal_partition, axiom, (
269   ~?[X]: (phaseMixin(X) & roleMixin(X))
270 )).
271
272 % Endurant Type (by modal properties of types) partial taxonomy
273 instances
274 % (tested rule out trivial models)
275
276 fof(ax_endurantType_instances_properties, axiom, (
277   % sortal(sortal1) & nonSortal(nonSortal1) & rigidSortal(
278     rigidSortal1) & antiRigidSortal(antiRigidSortal1) & kind(kind1)
279     & subkind(subkind1) & phase(phase1) & role(role1) & category(
280     category1) & mixin(mixin1) & antiRigidNonSortal(
281     antiRigidNonSortal1) & phaseMixin(phaseMixin1) & roleMixin(
282     roleMixin1)
283 % )).

```


2.1.6 Defining Types, Individuals, and Specialization

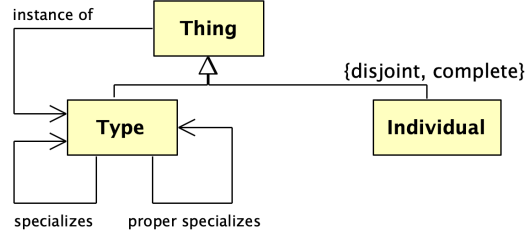


Figure 6: Types, individuals, instantiation, and specialization.

```

275 %%%%%%%%% Instance of, Types, and Individuals %%%%%%%%%
276
277 fof(ax_dIof, axiom, (
278   ![X,Y,W]: (iof(X,Y,W) => (type_(Y) & world(W)))
279 )).
280
281 fof(ax_dType_a1, axiom, (
282   ![X]: (type_(X) <=> (?[Y,W]: iof(Y,X,W)))
283 )).
284
285 fof(ax_dIndividual_a2, axiom, (
286   ![X]: (individual(X) <=> (~?[Y,W]: iof(Y,X,W)))
287 )).
288
289 % TODO: confirm whether we are including second-order types in this
      formalization
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 % fof(ax_iofInUse, axiom, (
300 %   type_(t2) & individual(i2) & world(w2) & iof(i2,t2,w2)
301 % )).
302
303 % Ax |= "th_everythingIsAThing_t1"; conjecture commented for
      convenience
304
305 % fof(th_everythingIsAThing_t1, conjecture, (
306 %   ![X]: (type_(X) | individual(X))
307 % )).
308
309 % Ax |= "th_thingPartition_t2"; conjecture commented for
      convenience
310
311 % fof(th_thingPartition_t2, conjecture, (
312 %   ~?[X]: (type_(X) & individual(X))

```

```

313 %)).
314
315 %%%%%%%%% Specialization and Proper Specialization %%%%%%%%%
316
317 fof(ax_dSpecializes, axiom, (
318   ![X,Y]: (specializes(X,Y) => (type_(X) & type_(Y)))
319 )).
320
321 fof(ax_specialization_a5, axiom, (
322   ![T1,T2]: (specializes(T1,T2) <=> (
323     type_(T1) & type_(T2) & ![W]: (world(W) => ![E]: (iof(E,T1,W)
324       => iof(E,T2,W))))
325 )).
326
327 fof(ax_properSpecializes_d1, axiom, (
328   ![X,Y]: (properSpecializes(X,Y) <=> (specializes(X,Y) & ~
329     specializes(Y,X)))
330 )).
331
332 % fof(ax_specializesInUse, axiom, (
333 %   type_(t3_1) & type_(t3_2) & specializes(t3_1,t3_2) &
334 %   properSpecializes(t3_1,t3_2) & specializes(t3_1,t3_1)
335 % )).
336
337 % Ax |= "th_cyclicSpecializations_t3"; conjecture commented for
338 %   convenience
339
340 fof(th_cyclicSpecializations_t3, conjecture, (
341   ![X,Y]: (specializes(X,Y) => (specializes(X,X) & specializes(Y,
342     Y))))
343 %)).
344
345 % Ax |= "th_transitiveSpecializations_t4"; conjecture commented for
346 %   convenience
347
348 fof(th_transitiveSpecializations_t4, conjecture, (
349   ![X,Y,Z]: ((specializes(X,Y) & specializes(Y,Z)) => (
350     specializes(X,Z)))
351 %)).
352
353 fof(ax_sharedSpecializations_a6, axiom, (
354   ![T1,T2]: (?[X,W]: ((iof(X,T1,W) & iof(X,T2,W) & ~specializes(T1,
355     T2) & ~specializes(T2,T1)) => (
356     (?[T3]: (specializes(T1,T3) & specializes(T2,T3) & iof(X,T3,W)
357       )))|
358     (?[T3]: (specializes(T3,T1) & specializes(T3,T2) & iof(X,T3,W)
359       )))
360 )))
361 )).
362

```

2.1.7 Defining Rigidity and Sortality

```

354 %%%%%%%%% Sortality and Rigidity %%%%%%%%%
355
356 % Rigidity

```

```

357
358 % TODO: I don't find we need to attach the "rigid(T)" predicate to
      the "endurant(T)" predicate like the paper does, so let's
      review this idea.
359 % TODO: verify whether it is a problem not to introduce predicates
      "world(W1) &" and "world(W2) &" before each instantiation
360
361 fof(ax_dRigid_a18, axiom, (
362   ![T]: (rigid(T) <=> (endurantType(T) & (
363     ![X]: ((![W1]: (world(W1) & iof(X,T,W1))) => (![W2]: (world(W2)
364       => iof(X,T,W2))))
365   )))
366
367 fof(ax_dAntiRigid_a19, axiom, (
368   ![T]: (antiRigid(T) <=> (endurantType(T) & (
369     ![X]: ((![W1]: (world(W1) & iof(X,T,W1))) => (?[W2]: (world(W2)
370       & ~iof(X,T,W2))))
371   )))
372
373 fof(ax_dSemiRigid_a20, axiom, (
374   ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid
375     (T)))
376
377 % Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented
      for convenience
378
379 % fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
380 %   ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid
381 %     (T)))
382 % ))).
383
384 % Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented
      for convenience
385
386 % fof(th_pairwiseDisjointRigidities_t6, conjecture, (
387 %   ~![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)
388 %     )) | (rigid(T) & antiRigid(T)))
389 % ))).
390
391 % Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture
      commented for convenience
392
393 % fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
394 %   ~![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
395 % ))).
396
397 % Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8";
      conjecture commented for convenience
398
399 % fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture,
      (
400 %   ~![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))

```

```

401 % % Sortality
402
403 fof(ax_endurantsKind_a21, axiom, (
404   ![E]: (endurant(E) => (
405     ?[U]: (kind(U) & (![W]: (world(W) & iof(E,U,W))))
406   ))
407)).
408
409 fof(ax_uniqueKind_a22, axiom, (
410   ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
411     ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
412   ))
413)).
414
415 % 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
416
417 % fof(ax_dSortal_a23, axiom, (
418 %   ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,
      W]: (iof(E,S,W) => iof(E,U,W)))))))
419 %)).
420
421 fof(ax_dSortal_a23, axiom, (
422   ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) &
      specializes(S,U))))
423)).
424
425 % If we have the taxonomy's axiomatization, then a24 becomes a
      theorem
426 % Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24";
      conjecture commented for convenience
427
428 % fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
429 %   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
430 %)).
431
432 % Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience
433
434 % fof(th_kindsAreRigid_t9, conjecture, (
435 %   ![U]: ((kind(U)) => (rigid(U)))
436 %)).
437
438 % Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented
      for convenience
439
440 % fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
441 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
442 %     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2
      ,W2)))
443 %   ))
444 %)).
445
446 % Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented
      for convenience
447

```

```

448 % fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
449 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
450 %     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
451 %   )
452 % )).
453
454 % Ax |= "th_kindsAreSortal_t12"; conjecture commented for
      convenience
455
456 % fof(th_kindsAreSortal_t12, conjecture, (
457 %   ![K]: ((kind(K)) => (sortal(K)))
458 % )).
459
460 % Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for
      convenience
461
462 % fof(th_sortalSpecializeKinds_t13, conjecture, (
463 %   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
464 % )).
465
466 % Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented
      for convenience
467
468 % fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
469 %   ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) &
470 %     specializes(S,U) & specializes(S,U2) & ~(U=U2))))
471 % )).
472
473 fof(ax_rigidSortalsAreRigidAndSortal_xx, axiom, (
474 %   ![T]: ((rigidSortal(T)) <=> (rigid(T) & sortal(T)))
475 % )).
476
477 fof(ax_antiRigidSortalsAreAntiRigidAndSortal_xx, axiom, (
478 %   ![T]: ((antiRigidSortal(T)) <=> (antiRigid(T) & sortal(T)))
479 % )).
480
481 fof(ax_rigidNonSortalsAreRigidAndNonSortal_xx, axiom, (
482 %   ![T]: ((rigidNonSortal(T)) <=> (rigid(T) & nonSortal(T)))
483 % )).
484
485 fof(ax_antiRigidNonSortalsAreAntiRigidAndNonSortal_xx, axiom, (
486 %   ![T]: ((antiRigidNonSortal(T)) <=> (antiRigid(T) & nonSortal(T)))
487 % )).
488
489 fof(ax_semiRigidNonSortalsAreSemiRigidAndNonSortal_xx, axiom, (
490 %   ![T]: ((semiRigidNonSortal(T)) <=> (semiRigid(T) & nonSortal(T)))
491 % )).
492
493 % If we have the taxonomy's axiomatization, then a25 becomes a
      theorem
494
495 % Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented
      for convenience
496
497 % fof(th_kindAndSubkindAreDisjoint_a25, conjecture, (
498 %   ~?[T]: (kind(T) & subkind(T))

```

```

499 % If we have the taxonomy's axiomatization, then a26 becomes a
      theorem
500 % Ax |= "th_kindAndSubkindAreRigidSortals_a26"; conjecture
      commented for convenience
501
502 % fof(th_kindAndSubkindAreRigidSortals_a26, conjecture, (
503 %   ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))
504 % )).
505
506 % If we have the taxonomy's axiomatization, then a27 becomes a
      theorem
507 % Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for
      convenience
508
509 % fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
510 %   ~?[T]: (phase(T) & role(T))
511 % )).
512
513 % If we have the taxonomy's axiomatization, then a28 becomes a
      theorem
514 % Ax |= "th_phaseAndRoleAreAntiRigidSortals_a28"; conjecture
      commented for convenience
515
516 % fof(th_phaseAndRoleAreAntiRigidSortals_a28, conjecture, (
517 %   ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
518 % )).
519
520 % Skipping (a29) because we leave the concept of semi-rigid sortals
      out of this ontology.
521
522 % If we have the taxonomy's axiomatization, then a30 becomes a
      theorem
523 % Ax |= "th_categoriesAreRigidNonSortals_a30"; conjecture commented
      for convenience
524
525 % fof(th_categoriesAreRigidNonSortals_a30, conjecture, (
526 %   ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))
527 % )).
528
529 % If we have the taxonomy's axiomatization, then a31 becomes a
      theorem
530 % Ax |= "th_mixinsAreSemiRigidNonSortals_a31"; conjecture commented
      for convenience
531
532 % fof(th_mixinsAreSemiRigidNonSortals_a31, conjecture, (
533 %   ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
534 % )).
535
536 % If we have the taxonomy's axiomatization, then a32 becomes a
      theorem
537 % Ax |= "th_phaseMixinAndRoleMixinAreDisjoint_a32"; conjecture
      commented for convenience
538
539 % fof(th_phaseMixinAndRoleMixinAreDisjoint_a32, conjecture, (
540 %   ~?[T]: (phaseMixin(T) & roleMixin(T))
541 % )).
542

```

```

543 % If we have the taxonomy's axiomatization, then a33 becomes a
    theorem
544 % Ax |= "ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a33";
    conjecture commented for convenience
545
546 % fof(th_phaseMixinAndRoleMixinAreAntiRigidSortals_a33, conjecture,
    (
547 %   ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) &
    nonSortal(T)))
548 % )).
549
550 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture
    commented for convenience
551
552 % fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
553 %   ~?[T]: (endurantType(T) & (
554 %     (
555 %       (kind(T) & subkind(T))
556 %       | (kind(T) & phase(T))
557 %       | (kind(T) & role(T))
558 %       | (kind(T) & category(T))
559 %       | (kind(T) & mixin(T))
560 %       | (kind(T) & phaseMixin(T))
561 %       | (kind(T) & roleMixin(T))
562 %     ) | (
563 %       (subkind(T) & phase(T))
564 %       | (subkind(T) & role(T))
565 %       | (subkind(T) & category(T))
566 %       | (subkind(T) & mixin(T))
567 %       | (subkind(T) & phaseMixin(T))
568 %       | (subkind(T) & roleMixin(T))
569 %     ) | (
570 %       (phase(T) & role(T))
571 %       | (phase(T) & category(T))
572 %       | (phase(T) & mixin(T))
573 %       | (phase(T) & phaseMixin(T))
574 %       | (phase(T) & roleMixin(T))
575 %     ) | (
576 %       (role(T) & category(T))
577 %       | (role(T) & mixin(T))
578 %       | (role(T) & phaseMixin(T))
579 %       | (role(T) & roleMixin(T))
580 %     ) | (
581 %       (category(T) & mixin(T))
582 %       | (category(T) & phaseMixin(T))
583 %       | (category(T) & roleMixin(T))
584 %     ) | (
585 %       (mixin(T) & phaseMixin(T))
586 %       | (mixin(T) & roleMixin(T))
587 %     ) | (
588 %       (phaseMixin(T) & roleMixin(T))
589 %     )
590 %   ))
591 % )).
592
593 % Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
    "; conjecture commented for convenience

```

```

594
595 % fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19,
    conjecture, (
596 %   ![T]: (endurantType(T) => (
597 %     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
    mixin(T) | phaseMixin(T) | roleMixin(T)
598 %   ))
599 % )).

```

2.1.8 Defining Endurant Types

```

601 %%%%%%%%%%%%% Endurant Types Definition %%%%%%%%%%%%%%
602
603 fof(ax_endurantTypeDefinition_xx, axiom, (
604   ![T]: (endurantType(T) <=> (
605     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (endurant(E))))
606   ))
607 )).
608
609 fof(ax_substantialTypeDefinition_xx, axiom, (
610   ![T]: (substantialType(T) <=> (
611     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (substantial(E)
612     )))
613   ))
614 )).
615
616 fof(ax_momentTypeDefinition_xx, axiom, (
617   ![T]: (momentType(T) <=> (
618     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (moment(E))))
619   ))
620 )).
621
622 fof(ax_objectTypeDefinition_xx, axiom, (
623   ![T]: (objectType(T) <=> (
624     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (object(E))))
625   ))
626 )).
627
628 fof(ax_collectiveTypeDefinition_xx, axiom, (
629   ![T]: (collectiveType(T) <=> (
630     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (collective(E)
631     )))
632   ))
633 )).
634
635 fof(ax_quantityTypeDefinition_xx, axiom, (
636   ![T]: (quantityType(T) <=> (
637     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quantity(E))))
638   ))
639 )).
640
641 fof(ax_intrinsicMomentTypeDefinition_xx, axiom, (
642   ![T]: (intrinsicMomentType(T) <=> (
643     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (
644       intrinsicMoment(E))))
645   ))
646 )).

```



```

645 fof(ax_relatorTypeDefinition_xx, axiom, (
646   ![T]: (relatorType(T) <=> (
647     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (relator(E))))
648   ))
649 ).
650
651 fof(ax_qualityTypeDefinition_xx, axiom, (
652   ![T]: (qualityType(T) <=> (
653     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quality(E))))
654   ))
655 ).
656
657 fof(ax_modeTypeDefinition_xx, axiom, (
658   ![T]: (modeType(T) <=> (
659     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (mode(E))))
660   ))
661 ).
662
663 fof(ax_endurantTypeDefinition_instances, axiom, (
664   substantial(substantial3) & substantialType(substantialType3)
665 ).
666
667 % fof(th_world, conjecture, (
668 %   ~?[X,Y,Z]: (iof(X,Y,Z) & type_(Z))
669 % ).
670
671 % fof(ax_endurantTypeDefinition_instances, axiom, (
672 %   substantial(substantial3) & moment(moment3) & object(object3) &
        collective(collective3) & quantity(quantity3) &
        intrinsicMoment(intrinsicMoment3) & relator(relator3) & quality
        (quality3) & mode(mode3) & disposition(disposition3) &
        externallyDependentMode(externallyDependentMode3) &
        quaIndividual(quaIndividual3) &
673 %   substantialType(substantialType3) & momentType(momentType3) &
        objectType(objectType3) & collectiveType(collectiveType3) &
        quantityType(quantityType3) & intrinsicMomentType(
        intrinsicMomentType3) & relatorType(relatorType3) & qualityType
        (qualityType3) & modeType(modeType3)
674 % ).

```