

# A TPTP Formalization of the Unified Foundational Ontology

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

October 18, 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)))
```

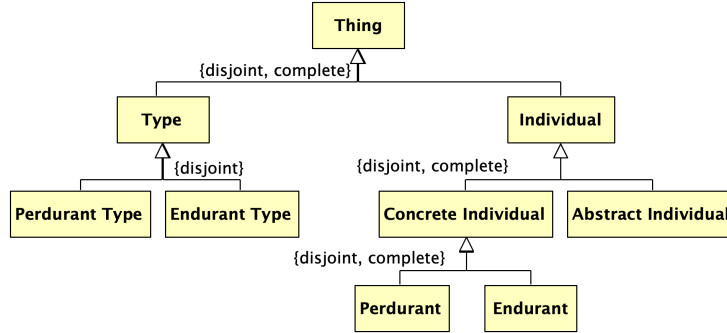


Figure 1: Partial Taxonomy of UFO – Thing.

```

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 to 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

```

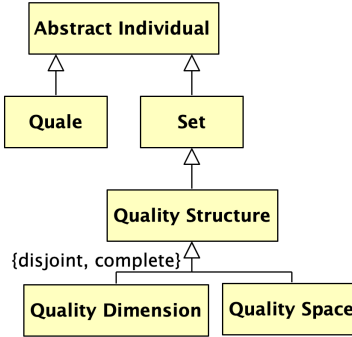


Figure 2: Partial Taxonomy of UFO – Abstract Individual.

```

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 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)

```

### 2.1.3 Partial Taxonomy of Endurant

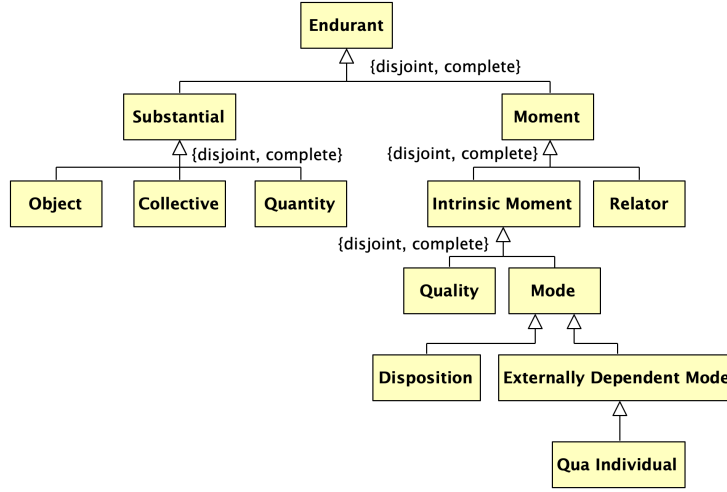


Figure 3: Partial Taxonomy of UFO – Endurant.

```

88 % fof(ax_abstractIndividual_instances, axiom, (
89 %   set_(set1) & quale(quale1) & qualityStructure(qualityStructure1
90 %   ) & qualityDimension(qualityDimension1) & qualitySpace(
91 %   qualitySpace1) & world(world1)
92 % )).
93
94 % Endurant
95
96 fof(ax_endurant_taxonomy, axiom, (
97   ![X]: ((substantial(X) | moment(X)) <=> (endurant(X)))
98 ))).
99
100 fof(ax_endurant_partition, axiom, (
101   ~?[X]: (substantial(X) & moment(X))
102 ))).
103
104 % Substantial
105
106 fof(ax_substantial_taxonomy, axiom, (
107   ![X]: ((object(X) | collective(X) | quantity(X)) <=> (substantial
108   (X)))
109 ))).
110
111 fof(ax_substantial_partition, axiom, (
112   ~?[X]: ((object(X) & collective(X)) | (object(X) & quantity(X)) |
113   (collective(X) & quantity(X)))
114 ))).
115
116 % Moment
117
118 fof(ax_moment_taxonomy, axiom, (
119   ![X]: ((intrinsicMoment(X) | relator(X)) <=> (moment(X)))
120 ))).

```

```

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)

```

#### 2.1.4 Partial Taxonomy of Endurant Type (on ontological natures)

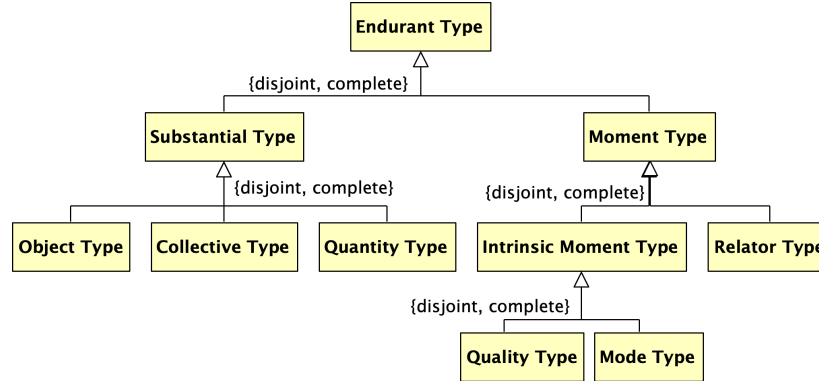


Figure 4: Partial Taxonomy of UFO – Endurant Types (by ontological nature).

```

147 % fof(ax_endurant_instances, axiom, (
148 %   substantial(substantial1) & moment(moment1) & object(object1) &
      collective(collective1) & quantity(quantity1) &
      intrinsicMoment(intrinsicMoment1) & relator(relator1) & quality
      (quality1) & mode(mode1) & disposition(disposition1) &

```

```

    externallyDependentMode(externallyDependentMode1) &
    quaIndividual(quaIndividual1)
149 %)).
150
151 % Endurant Type (by ontological nature)
152
153 fof(ax_endurantType_taxonomy_nature, axiom, (
154   ![X]: ((substantialType(X) | momentType(X)) <=> (endurantType(X))
155   ))).
156
157 fof(ax_endurantType_partition_nature, axiom, (
158   ~?[X]: (substantialType(X) & momentType(X))
159   )).
160
161 % Substantial Type
162
163 fof(ax_substantialType_taxonomy, axiom, (
164   ![X]: ((objectType(X) | collectiveType(X) | quantityType(X)) <=>
165   (substantialType(X)))
166   )).
167
168 fof(ax_substantialType_partition, axiom, (
169   ~?[X]: ((objectType(X) & collectiveType(X)) | (objectType(X) &
170   quantityType(X)) | (collectiveType(X) & quantityType(X)))
171   )).
172
173 % Moment Type
174
175 fof(ax_momentType_taxonomy, axiom, (
176   ![X]: ((intrinsicMomentType(X) | relatorType(X)) <=> (momentType(
177   X)))
178   )).
179
180 fof(ax_momentType_partition, axiom, (
181   ~?[X]: (intrinsicMomentType(X) & relatorType(X))
182   )).
183
184 % Intrinsic Moment Type
185
186 fof(ax_intrinsicMomentType_taxonomy, axiom, (
187   ![X]: ((qualityType(X) | modeType(X)) <=> (intrinsicMomentType(X)
188   ))
189   )).
190
191 fof(ax_intrinsicMomentType_partition, axiom, (
192   ~?[X]: (qualityType(X) & modeType(X))
193   )).
194
195 % Endurant Type (by ontological nature) partial taxonomy instances
196 % (tested to rule out trivial models)

```

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

```

194 % fof(ax_endurantType_instances_natures, axiom, (

```

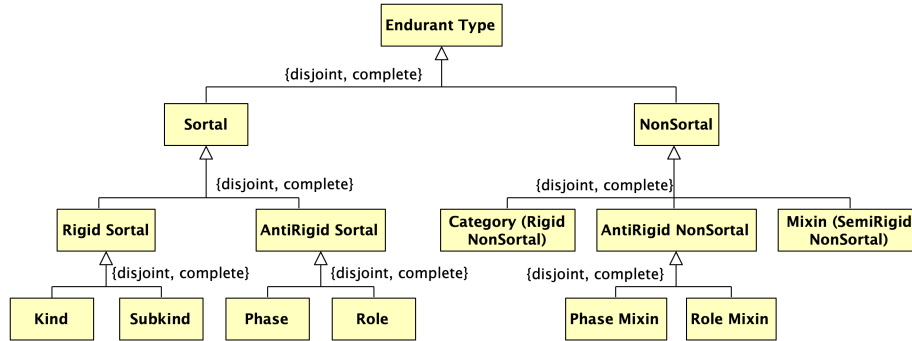


Figure 5: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```

195 %   substantialType(substantialType1) & momentType(momentType1) &
      objectType(objectType1) & collectiveType(collectiveType1) &
      quantityType(quantityType1) & intrinsicMomentType(
196 %   intrinsicMomentType1) & relatorType(relatorType1) & qualityType
      (qualityType1) & modeType(modeType1)
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 fof(ax_antiRigidSortal_taxonomy, axiom, (
230   ![X]: ((phase(X) | role(X)) <=> (antiRigidSortal(X)))
231 ))).
232
233 fof(ax_antiRigidSortal_partition, axiom, (
234   ~?[X]: (phase(X) & role(X))
235 ))).
236
237 % Non-Sortal
238
239 fof(ax_nonSortal_taxonomy, axiom, (
240   ![X]: ((rigidNonSortal(X) | semiRigidNonSortal(X) |
241     antiRigidNonSortal(X)) <=> (nonSortal(X)))
242 ))).
243
244 fof(ax_nonSortal_partition, axiom, (
245   ~?[X]: ((rigidNonSortal(X) & semiRigidNonSortal(X)) | (
246     rigidNonSortal(X) & antiRigidNonSortal(X)) | (
247     semiRigidNonSortal(X) & antiRigidNonSortal(X)))
248 ))).
249
250 % Category
251
252 fof(ax_rigidNonSortal_taxonomy, axiom, (
253   ![X]: (rigidNonSortal(X) <=> (category(X)))
254 ))).
255
256 % Mixin
257
258 fof(ax_semiRigidNonSortal_taxonomy, axiom, (
259   ![X]: (semiRigidNonSortal(X) <=> (mixin(X)))
260 ))).
261
262 % Anti-Rigid Non-Sortal
263
264 fof(ax_antiRigidNonSortal_taxonomy, axiom, (
265   ![X]: ((phaseMixin(X) | roleMixin(X)) <=> (antiRigidNonSortal(X))
266     )
267 ))).
268
269 fof(ax_antiRigidNonSortal_partition, axiom, (
270   ~?[X]: (phaseMixin(X) & roleMixin(X))
271 ))).
272
273 % Endurant Type (by modal properties of types) partial taxonomy
274 instances
275 % (tested to rule out trivial models)

```

## 2.1.6 Defining Types, Individuals, 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)

```



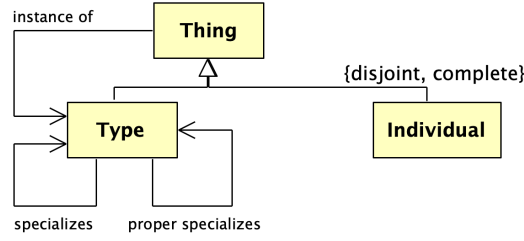


Figure 6: Types, individuals, instantiation, and specialization.

```

275 %)).
276
277 %%%%%%%%% Instance of, Types, and Individuals %%%%%%%%%
278
279 fof(ax_dIof, axiom, (
280   ![X,Y,W]: (iof(X,Y,W) => (type_(Y) & world(W)))
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 % TODO: confirm whether we are including second-order types in this
      formalization
292
293 fof(ax_multiLevel_a3, axiom, (
294   ![X,Y,W]: (iof(X,Y,W) => (type_(X) | individual(X)))
295 )).
296
297 fof(ax_twoLevelConstrained_a4, axiom, (
298   ~?[X,Y,Z,W]: (type_(X) & iof(X,Y,W) & iof(Y,Z,W))
299 )).
300
301 % Instantiation relations
302 % (tested to rule out trivial models)
303
304 % fof(ax_iofInUse, axiom, (
305 %   type_(t2) & individual(i2) & world(w2) & iof(i2,t2,w2)
306 % )).
307
308 % Ax |= "th_everythingIsAThing_t1"; conjecture commented for
      convenience
309
310 % fof(th_everythingIsAThing_t1, conjecture, (
311 %   ![X]: (type_(X) | individual(X))
312 % )).
313
314 % Ax |= "th_thingPartition_t2"; conjecture commented for
      convenience

```

```

315
316 % fof(th_thingPartition_t2, conjecture, (
317 %   ~?[X]: (type_(X) & individual(X))
318 % )).
319
320 %%%%%%%%% Specialization and Proper Specialization %%%%%%%%%
321
322 fof(ax_dSpecializes, axiom, (
323   ![X,Y]: (specializes(X,Y) => (type_(X) & type_(Y)))
324 )).
325
326 fof(ax_specialization_a5, axiom, (
327   ![T1,T2]: (specializes(T1,T2) <=> (
328     type_(T1) & type_(T2) & ![W]: (world(W) => ![E]: (iof(E,T1,W)
329       => iof(E,T2,W)))
330   ))).
331
332 fof(ax_properSpecializes_d1, axiom, (
333   ![X,Y]: (properSpecializes(X,Y) <=> (specializes(X,Y) & ~
334     specializes(Y,X)))
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 % Specialization relations
363 % (tested to rule out trivial models)

```

## 2.1.7 Defining Rigidity and Sortality

```

360 % )).
361
362 %%%%%%%%% Sortality and Rigidity %%%%%%%%%
363

```

```

364 % Rigidity
365
366 % 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.
367 % TODO: verify whether it is a problem not to introduce predicates
      "world(W1) &" and "world(W2) &" before each instantiation
368
369 fof(ax_dRigid_a18, axiom, (
370   ![T]: (rigid(T) <=> (endurantType(T) & (
371     ![X]: ((?[W1]: (world(W1) & iof(X,T,W1))) => (![W2]: (world(W2)
372       => iof(X,T,W2))))
373   )))
374
375 fof(ax_dAntiRigid_a19, axiom, (
376   ![T]: (antiRigid(T) <=> (endurantType(T) & (
377     ![X]: ((?[W1]: (world(W1) & iof(X,T,W1))) => (?[W2]: (world(W2)
378       & ~iof(X,T,W2))))
379   )))
380
381 fof(ax_dSemiRigid_a20, axiom, (
382   ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid
383     (T)))
384
385 % Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented
      for convenience
386
387 % fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
388 %   ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid
389 %     (T)))
390 % ))).
391
392 % Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented
      for convenience
393
394 % fof(th_pairwiseDisjointRigidities_t6, conjecture, (
395 %   ~![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)
396 %     )) | (rigid(T) & antiRigid(T)))
397 % ))).
398
399 % Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture
      commented for convenience
400
401 % fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
402 %   ~![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
403 % ))).
404
405 % Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8";
      conjecture commented for convenience
406
407 % fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture, (
408 %   ~![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
409 % ))).

```

```

408
409 % Rigidity properties
410 % (tested to rule out trivial models)
411
412 % fof(ax_rigidityInUse, axiom, (
413 %   endurantType(t4_1) & endurantType(t4_2) & endurantType(t4_3) &
414 %   rigid(t4_1) & semiRigid(t4_2) & antiRigid(t4_3) &
415 %   properSpecializes(t4_1,t4_2) & properSpecializes(t4_3,t4_1)
416 % )).
417
418 % Sortality
419 fof(ax_endurantsKind_a21, axiom, (
420   ![E]: (endurant(E) => (
421     ?[U]: (kind(U) & (![W]: (world(W) => iof(E,U,W))))
422   ))
423
424 fof(ax_uniqueKind_a22, axiom, (
425   ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
426     ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
427   ))
428
429
430 % Changing "ax_dSortal_a23" from the form it was defined in the
431 % paper to "sortals are endurant types that specialize some
432 % ultimate sortal" seem to express the same concept while
433 % speeding up the execution of SPASS considerably
434
435 % fof(ax_dSortal_a23, axiom, (
436 %   ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,
437 %     W]: (iof(E,S,W) => iof(E,U,W))))))
438 % )).
439
440 fof(ax_dSortal_a23, axiom, (
441   ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) &
442     specializes(S,U))))
443
444
445 % If we have the taxonomy's axiomatization, then a24 becomes a
446 % theorem
447 % Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24";
448 % conjecture commented for convenience
449
450 % fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
451 %   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
452 % )).
453
454 % Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience
455
456 % fof(th_kindsAreRigid_t9, conjecture, (
457 %   ![U]: ((kind(U)) => (rigid(U)))
458 % )).
459
460 % Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented
461 % for convenience
462
463

```

```

455 % fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
456 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
457 %     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2
      ,W2)))
458 %   )
459 % )).
460
461 % Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented
      for convenience
462
463 % fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
464 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
465 %     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
466 %   )
467 % )).
468
469 % Ax |= "th_kindsAreSortal_t12"; conjecture commented for
      convenience
470
471 % fof(th_kindsAreSortal_t12, conjecture, (
472 %   ![K]: ((kind(K)) => (sortal(K)))
473 % )).
474
475 % Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for
      convenience
476
477 % fof(th_sortalSpecializeKinds_t13, conjecture, (
478 %   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
479 % )).
480
481 % Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented
      for convenience
482
483 % fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
484 %   ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) &
      specializes(S,U) & specializes(S,U2) & ~(U=U2))))
485 % )).
486
487 % Sortality properties
488 % (tested to rule out trivial models)
489
490 % fof(ax_sortalityInUse, axiom, (
491 %   enduring(e5_1) & enduring(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)
492 % )).
493
494 % Sortality + Rigidity
495
496 fof(ax_rigidSortalsAreRigidAndSortal_xx, axiom, (
497 %   ![T]: ((rigidSortal(T)) <=> (rigid(T) & sortal(T)))
498 % )).
499
500 fof(ax_antiRigidSortalsAreAntiRigidAndSortal_xx, axiom, (
501 %   ![T]: ((antiRigidSortal(T)) <=> (antiRigid(T) & sortal(T)))
502 % )).
503
504 fof(ax_rigidNonSortalsAreRigidAndNonSortal_xx, axiom, (

```

```

505 ![T]: ((rigidNonSortal(T)) <=> (rigid(T) & nonSortal(T)))
506 )).
507
508 fof(ax_antiRigidNonSortalsAreAntiRigidAndNonSortal_xx, axiom, (
509   ![T]: ((antiRigidNonSortal(T)) <=> (antiRigid(T) & nonSortal(T)))
510 )).
511
512 fof(ax_semiRigidNonSortalsAreSemiRigidAndNonSortal_xx, axiom, (
513   ![T]: ((semiRigidNonSortal(T)) <=> (semiRigid(T) & nonSortal(T)))
514 )).
515
516 % If we have the taxonomy's axiomatization, then a25 becomes a
517 % theorem
518 % Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented
519 % for convenience
520
521 % fof(th_kindAndSubkindAreDisjoint_a25, conjecture, (
522 %   ~?[T]: (kind(T) & subkind(T))
523 % )).
524
525 % If we have the taxonomy's axiomatization, then a26 becomes a
526 % theorem
527 % Ax |= "th_kindAndSubkindAreRigidSortals_a26"; conjecture
528 % commented for convenience
529
530 % fof(th_kindAndSubkindAreRigidSortals_a26, conjecture, (
531 %   ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))
532 % )).
533
534 % If we have the taxonomy's axiomatization, then a27 becomes a
535 % theorem
536 % Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for
537 % convenience
538
539 % fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
540 %   ~?[T]: (phase(T) & role(T))
541 % )).
542
543 % If we have the taxonomy's axiomatization, then a28 becomes a
544 % theorem
545 % Ax |= "th_phaseAndRoleAreAntiRigidSortals_a28"; conjecture
546 % commented for convenience
547
548 % fof(th_phaseAndRoleAreAntiRigidSortals_a28, conjecture, (
549 %   ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
550 % )).
551
552 % Skipping (a29) because we leave the concept of semi-rigid sortals
553 % out of this ontology.
554
555 % If we have the taxonomy's axiomatization, then a30 becomes a
556 % theorem
557 % Ax |= "th_categoriesAreRigidNonSortals_a30"; conjecture commented
558 % for convenience
559
560 % fof(th_categoriesAreRigidNonSortals_a30, conjecture, (
561 %   ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))

```

```

551 %)).
552
553 % If we have the taxonomy's axiomatization, then a31 becomes a
    theorem
554 % Ax |= "th_mixinsAreSemiRigidNonSortals_a31"; conjecture commented
    for convenience
555
556 % fof(th_mixinsAreSemiRigidNonSortals_a31, conjecture, (
557 %   ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
558 % )).
559
560 % If we have the taxonomy's axiomatization, then a32 becomes a
    theorem
561 % Ax |= "th_phaseMixinAndRoleMixinAreDisjoint_a32"; conjecture
    commented for convenience
562
563 % fof(th_phaseMixinAndRoleMixinAreDisjoint_a32, conjecture, (
564 %   ~?[T]: (phaseMixin(T) & roleMixin(T))
565 % )).
566
567 % If we have the taxonomy's axiomatization, then a33 becomes a
    theorem
568 % Ax |= "ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a33";
    conjecture commented for convenience
569
570 % fof(th_phaseMixinAndRoleMixinAreAntiRigidSortals_a33, conjecture,
    (
571 %   ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) &
    nonSortal(T)))
572 % )).
573
574 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture
    commented for convenience
575
576 % fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
577 %   ~?[T]: (endurantType(T) & (
578 %     (
579 %       (kind(T) & subkind(T))
580 %       | (kind(T) & phase(T))
581 %       | (kind(T) & role(T))
582 %       | (kind(T) & category(T))
583 %       | (kind(T) & mixin(T))
584 %       | (kind(T) & phaseMixin(T))
585 %       | (kind(T) & roleMixin(T))
586 %     ) | (
587 %       (subkind(T) & phase(T))
588 %       | (subkind(T) & role(T))
589 %       | (subkind(T) & category(T))
590 %       | (subkind(T) & mixin(T))
591 %       | (subkind(T) & phaseMixin(T))
592 %       | (subkind(T) & roleMixin(T))
593 %     ) | (
594 %       (phase(T) & role(T))
595 %       | (phase(T) & category(T))
596 %       | (phase(T) & mixin(T))
597 %       | (phase(T) & phaseMixin(T))
598 %       | (phase(T) & roleMixin(T))

```

```

599 %      ) | (
600 %      (role(T) & category(T))
601 %      | (role(T) & mixin(T))
602 %      | (role(T) & phaseMixin(T))
603 %      | (role(T) & roleMixin(T))
604 %      ) | (
605 %      (category(T) & mixin(T))
606 %      | (category(T) & phaseMixin(T))
607 %      | (category(T) & roleMixin(T))
608 %      ) | (
609 %      (mixin(T) & phaseMixin(T))
610 %      | (mixin(T) & roleMixin(T))
611 %      ) | (
612 %      (phaseMixin(T) & roleMixin(T))
613 %      )
614 %    ))
615 % )).
616
617 % Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
        "; conjecture commented for convenience
618
619 % fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19,
        conjecture, (
620 %   ![T]: (endurantType(T) => (
621 %     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
        mixin(T) | phaseMixin(T) | roleMixin(T)
622 %   ))
623 % )).
624
625 % Sortality and rigidity properties combined
626 % (tested to rule out trivial models)

```

### 2.1.8 Defining Endurant Types

```

628 % fof(ax_sortalityAndRigidityInUse, axiom, (
629 %   endurant(e6_1) & endurant(e6_2) & world(w6) & kind(k6_1) & kind
        (k6_2) & iof(e6_1,k6_1,w6) & iof(e6_1,k6_1,w6) & ~(k6_1=k6_2)
630 % )).
631
632 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Types Definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
633
634 % Defining the taxonomy of types of ontological natures through the
        categorization of the taxonomy of concrete individuals
635
636 fof(ax_perdurantTypeDefinition_a44, axiom, (
637 %   ![T]: (perdurantType(T) <=> (
638 %     type_(T) & (![P,W]: ((world(W) & iof(P,T,W)) => (perdurant(P)))
        )
639 %   ))
640 % )).
641
642 fof(ax_endurantTypeDefinition_a44, axiom, (
643 %   ![T]: (endurantType(T) <=> (
644 %     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (endurant(E))))
        )
645 %   ))
646 % )).
647
648 fof(ax_substantialTypeDefinition_a44, axiom, (

```



```

649     ![T]: (substantialType(T) <=> (
650         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (substantial(E)
651         )))
652     )).
653
654 fof(ax_momentTypeDefinition_a44, axiom, (
655     ![T]: (momentType(T) <=> (
656         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (moment(E))))
657     ))
658 )).
659
660 fof(ax_objectTypeDefinition_a44, axiom, (
661     ![T]: (objectType(T) <=> (
662         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (object(E))))
663     ))
664 )).
665
666 fof(ax_collectiveTypeDefinition_a44, axiom, (
667     ![T]: (collectiveType(T) <=> (
668         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (collective(E)))
669     ))
670 )).
671
672 fof(ax_quantityTypeDefinition_a44, axiom, (
673     ![T]: (quantityType(T) <=> (
674         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quantity(E))))
675     ))
676 )).
677
678 fof(ax_intrinsicMomentTypeDefinition_a44, axiom, (
679     ![T]: (intrinsicMomentType(T) <=> (
680         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (
681             intrinsicMoment(E))))
682     ))
683 )).
684
685 fof(ax_relatorTypeDefinition_a44, axiom, (
686     ![T]: (relatorType(T) <=> (
687         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (relator(E))))
688     ))
689 )).
690
691 fof(ax_qualityTypeDefinition_a44, axiom, (
692     ![T]: (qualityType(T) <=> (
693         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quality(E))))
694     ))
695 )).
696
697 fof(ax_modeTypeDefinition_a44, axiom, (
698     ![T]: (modeType(T) <=> (
699         type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (mode(E))))
700     ))
701 )).
702 % Types Definition

```

```

703 % (tested to rule out trivial models)
704 % TODO: investigate why we cannot list four different enduring
      types (it may have something to do with "intrinsicMoment" and "
      intrinsicMomentType")
705
706 % fof(ax_typesDefinitionsInstances, axiom, (
707 %   objectType(ot7) & collectiveType(ct7) & modeType(mt7)
708 % )).
709
710 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t21"; conjecture
      commented for convenience
711 % Having the previously defined taxonomy, this should be quite
      trivial
712
713 % fof(th_leafCategoriesArePairwiseDisjoint_t21, conjecture, (
714 %   ~?[T]: (type_(T) & (
715 %     (
716 %       (objectType(T) & collectiveType(T)) | (objectType(T) &
          quantityType(T)) | (objectType(T) & modeType(T)) | (objectType(
          T) & qualityType(T)) | (objectType(T) & relatorType(T)) | (
          objectType(T) & perdurantType(T))
717 %     ) | (
718 %       (collectiveType(T) & quantityType(T)) | (collectiveType(T)
          & modeType(T)) | (collectiveType(T) & qualityType(T)) | (
          collectiveType(T) & relatorType(T)) | (collectiveType(T) &
          perdurantType(T))
719 %     ) | (
720 %       (quantityType(T) & modeType(T)) | (quantityType(T) &
          qualityType(T)) | (quantityType(T) & relatorType(T)) | (
          quantityType(T) & perdurantType(T))
721 %     ) | (
722 %       (modeType(T) & qualityType(T)) | (modeType(T) & relatorType
          (T)) | (modeType(T) & perdurantType(T))
723 %     ) | (
724 %       (qualityType(T) & relatorType(T)) | (qualityType(T) &
          perdurantType(T))
725 %     ) | (
726 %       relatorType(T) & perdurantType(T)
727 %     )
728 %   ))
729 % )).
730
731 % Ultimate Sortals Definitions (by ontological nature)
732
733 fof(ax_objectKindDefinition_a45, axiom, (
734   ![T]: (objectKind(T) <=> (objectType(T) & kind(T)))
735 )).
736
737 fof(ax_collectiveKindDefinition_a45, axiom, (
738   ![T]: (collectiveKind(T) <=> (collectiveType(T) & kind(T)))
739 )).
740
741 fof(ax_quantityKindDefinition_a45, axiom, (
742   ![T]: (quantityKind(T) <=> (quantityType(T) & kind(T)))
743 )).
744
745 fof(ax_modeKindDefinition_a45, axiom, (

```

```

746 ![T]: (modeKind(T) <=> (modeType(T) & kind(T)))
747 )).
748
749 fof(ax_qualityKindDefinition_a45, axiom, (
750   ![T]: (qualityKind(T) <=> (qualityType(T) & kind(T)))
751 )).
752
753 fof(ax_relatorKindDefinition_a45, axiom, (
754   ![T]: (relatorKind(T) <=> (relatorType(T) & kind(T)))
755 )).
756
757 % Ultimate sortals (by ontological nature) instances
758 % (tested to rule out trivial models)
759 % TODO: investigate why we cannot list all different types of
       ultimate sortals at once
760
761 % fof(ax_typesDefinitionsInstances, axiom, (
762 %   objectKind(ok9) & collectiveKind(ck9) & quantityKind(quank9) &
       relatorKind(rk9) & modeKind(mk9) & qualityKind(qualk9)
763 % )).
764
765 % Skipping (t22) because (a21) makes it trivial
766
767 % Ax |= "th_endurantsInstantiateEndurantKindsOfSomeNature_a46";
       conjecture commented for convenience
768 % This axiom is actually a theorem in this version of the
       axiomatization
769
770 % fof(th_endurantsInstantiateEndurantKindsOfSomeNature_a46,
       conjecture, (
771 %   ![E]: (endurant(E) => (
772 %     ?[K,W]: ((objectKind(K) | collectiveKind(K) | quantityKind(K)
773 %       | modeKind(K) | qualityKind(K) | relatorKind(K))
774 %     & iof(E,K,W))
775 %   ))
776 % )).
777
778 % Ax |= "th_endurantSortalsCompleteness_t23"; conjecture commented
       for convenience
779
780 % Thanks to the taxonomy, we already have "sortal(T) =>
       endurantType(T)", but I leave it like this to be consistent
       with the paper
781
782 % fof(th_endurantSortalsCompleteness_t23, conjecture, (
783 %   ![T]: ((endurantType(T) & sortal(T)) => (objectKind(T) |
       collectiveKind(T) | quantityKind(T) | qualityKind(T) | modeKind
       (T) | relatorKind(T) | phase(T) | role(T)))
784 % )).
785
786 % Ax |= "th_objectTypesSpecializeAKindOfSameNature_t24"; conjecture
       commented for convenience
787
788 % fof(th_objectTypesSpecializeAKindOfSameNature_t24, conjecture, (
789 %   ![T]: ((objectType(T) & sortal(T)) <=> (?[K]: (objectKind(K) &
       specializes(T,K))))
790 % )).

```

```

790 % Ax |= "th_collectiveTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
791
792 % fof(th_collectiveTypesSpecializeAKindOfSameNature_t24, conjecture
      , (
793 %   ![T]: ((collectiveType(T) & sortal(T)) <=> (?[K]: (
      collectiveKind(K) & specializes(T,K))))
794 % )).
795
796 % Ax |= "th_quantityTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
797
798 % fof(th_quantityTypesSpecializeAKindOfSameNature_t24, conjecture,
      (
799 %   ![T]: ((quantityType(T) & sortal(T)) <=> (?[K]: (quantityKind(K)
      ) & specializes(T,K))))
800 % )).
801
802 % Ax |= "th_modeTypesSpecializeAKindOfSameNature_t24"; conjecture
      commented for convenience
803
804 % fof(th_modeTypesSpecializeAKindOfSameNature_t24, conjecture, (
805 %   ![T]: ((modeType(T) & sortal(T)) <=> (?[K]: (modeKind(K) &
      specializes(T,K))))
806 % )).
807
808 % Ax |= "th_qualityTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
809
810 % fof(th_qualityTypesSpecializeAKindOfSameNature_t24, conjecture, (
811 %   ![T]: ((qualityType(T) & sortal(T)) <=> (?[K]: (qualityKind(K)
      & specializes(T,K))))
812 % )).
813
814 % Ax |= "th_relatorTypesSpecializeAKindOfSameNature_t24";
      conjecture commented for convenience
815
816 % fof(th_relatorTypesSpecializeAKindOfSameNature_t24, conjecture, (
817 %   ![T]: ((relatorType(T) & sortal(T)) <=> (?[K]: (relatorKind(K)
      & specializes(T,K))))
818 % )).
819
820 % Ax |= "th_sortalLeafCategoriesAreDisjoint_t25"; conjecture
      commented for convenience
821
822 % fof(th_sortalLeafCategoriesAreDisjoint_t25, conjecture, (
823 %   ![T]: (objectKind(T) => (~(collectiveKind(T) | quantityKind(T)
      | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
      mixin(T) | phaseMixin(T) | roleMixin(T))))
824 %   & ![T]: (collectiveKind(T) => (~(objectKind(T) | quantityKind(T)
      | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
825 %   & ![T]: (quantityKind(T) => (~(objectKind(T) | collectiveKind(T)
      | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T)
      | mixin(T) | phaseMixin(T) | roleMixin(T))))
826 %   & ![T]: (modeKind(T) => (~(objectKind(T) | quantityKind(T) |
      collectiveKind(T) | qualityKind(T) | relatorKind(T) | category(

```

```

T) | mixin(T) | phaseMixin(T) | roleMixin(T)))
827 % & ![T]: (qualityKind(T) => (~ (objectKind(T) | quantityKind(T) |
modeKind(T) | collectiveKind(T) | relatorKind(T) | category(T)
| mixin(T) | phaseMixin(T) | roleMixin(T))))
828 % & ![T]: (relatorKind(T) => (~ (objectKind(T) | quantityKind(T) |
modeKind(T) | qualityKind(T) | collectiveKind(T) | category(T)
| mixin(T) | phaseMixin(T) | roleMixin(T))))
829 % & ![T]: (category(T) => (~ (objectKind(T) | quantityKind(T) |
modeKind(T) | qualityKind(T) | relatorKind(T) | collectiveKind(
T) | mixin(T) | phaseMixin(T) | roleMixin(T))))
830 % & ![T]: (mixin(T) => (~ (objectKind(T) | quantityKind(T) |
modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
collectiveKind(T) | phaseMixin(T) | roleMixin(T))))
831 % & ![T]: (phaseMixin(T) => (~ (objectKind(T) | quantityKind(T) |
modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
mixin(T) | collectiveKind(T) | roleMixin(T))))
832 % & ![T]: (roleMixin(T) => (~ (objectKind(T) | quantityKind(T) |
modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
mixin(T) | phaseMixin(T) | collectiveKind(T))))
833 % )).
834
835 % Ax |= "th_sortalLeafCategoriesAreComplete_t26"; conjecture
commented for convenience

```

## 2.1.9 Mereology

```

837 % fof(th_sortalLeafCategoriesAreComplete_t26, conjecture, (
838 %   ![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)))
839 % )).
840
841 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Mereology %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
842
843 % TODO: review whether it is necessary to reduce mereology to
concrete individuals; I am leaving this axiom out for the
moment
844
845 % fof(ax_partArguments, axiom, (
846 %   ![X,Y]: (part(X,Y) => (concreteIndividual(X) &
concreteIndividual(Y)))
847 % )).
848
849 fof(ax_reflexiveParthood, axiom, (
850   ![X]: (partOf(X,X))
851 )).
852
853 fof(ax_antiSymmetricParthood_a47, axiom, (
854   ![X,Y]: ((partOf(X,Y) & partOf(Y,X)) => (X=Y))
855 )).
856
857 fof(ax_antiSymmetricParthood_a48, axiom, (
858   ![X,Y]: ((partOf(X,Y) & partOf(Y,X)) => (X=Y))
859 )).
860
861 fof(ax_transitiveParthood_a49, axiom, (
862   ![X,Y,Z]: ((partOf(X,Y) & partOf(Y,Z)) => (partOf(X,Z)))

```

```

863 ))).
864
865 fof(ax_overlappingWholes_a50, axiom, (
866   ![X,Y]: ((overlap(X,Y)) <=> (?[Z]: (partOf(Z,X) & partOf(Z,Y))))
867 ))).
868
869 fof(ax_strongSupplementation_a51, axiom, (
870   ![X,Y]: (~partOf(X,Y) <=> ?[Z]: (partOf(Z,X) & ~overlap(Z,Y)))
871 ))).
872
873 fof(ax_properPart_a52, axiom, (
874   ![X,Y]: (properPartOf(X,Y) <=> (partOf(X,Y) & ~partOf(Y,X)))
875 ))).
876
877 fof(ax_binarySum_a53, axiom, (
878   ![X,Y,Z]: (sum(Z,X,Y) <=> ![W]: (overlap(W,Z) <=> (overlap(W,X) |
879     overlap(W,Y))))
880 ))).
881
882 % TODO: check whether it is necessary to introduce fusion and
883 %       existence of sums, and how to do it
884
885 % Mereology in use
886 % (tested to rule out trivial models)
887
888 % fof(ax_mereologyInUse, axiom, (
889 %   concreteIndividual(ci10_1) & concreteIndividual(ci10_2) &
890 %   concreteIndividual(ci10_3) & concreteIndividual(ci10_4) &
891 %   concreteIndividual(ci10_5) & ~(ci10_1=ci10_2) & ~(ci10_2=ci10_3)
892 %   & ~(ci10_3=ci10_4) & ~(ci10_4=ci10_5) & properPart(ci10_1,
893 %     ci10_2) & properPart(ci10_3,ci10_4) & sum(ci10_5,ci10_3,ci10_4)
894 % ))).

```

## 2.1.10 Composition

```

886 % fof(ax_mereologyInUse, axiom, (
887 %   concreteIndividual(ci10_1) & concreteIndividual(ci10_2) &
888 %   concreteIndividual(ci10_3) & concreteIndividual(ci10_4) &
889 %   concreteIndividual(ci10_5) & ~(ci10_1=ci10_2) & ~(ci10_2=ci10_3)
890 %   & ~(ci10_3=ci10_4) & ~(ci10_4=ci10_5) & properPart(ci10_1,
891 %     ci10_2) & properPart(ci10_3,ci10_4) & sum(ci10_5,ci10_3,ci10_4)
892 % ))).
893
894 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
895 % Composition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
896
897 % TODO: review why we need to constrain functions to hold between
898 %       endurants and types only (not even "endurant types")
899
900 fof(ax_function, axiom, (
901   ![X,Y]: (function(X,Y) => (endurant(X) & type_(Y)))
902 ))).
903
904 fof(ax_genericFunctionalDependence_a55, axiom, (
905   ![T1,T2,W]: (gfd(T1,T2,W) <=>
906     ![E1]: ((iof(T1,E1,W) & function(T1,E1)) => ?[E2]: (~ (E1=E2) &
907       iof(T2,E2,W) & function(T2,E2))))
908 ))).
909
910

```

```

903 fof(ax_individualFunctionalDependence_a56, axiom, (
904   ![E1,T1,E2,T2,W]: (ifd(E1,T1,E2,T2,W) <=> (
905     gfd(T1,T2,W) & ioef(E1,T1,W) & ioef(E2,T2,W) & (function(E1,T1)
906     => function(E2,T2))
907   ))).
908
909 fof(ax_componentOf_a57, axiom, (
910   ![E1,T1,E2,T2,W]: (componentOf(E1,T1,E2,T2,W) <=> (properPartOf(
911     E1,E2) & ifd(E1,T1,E2,T2,W)))
912 ).
913 % Composition in use
914 % (tested to rule out trivial models)

```

### 2.1.11 Constitution

```

916 % fof(ax_compositionInUse, axiom, (
917 %   componentOf(e11_1,t11_1,e11_2,t11_2,w11) & ~(e11_1=e11_2) & ~(
918 %     e11_1=t11_1) & ~(e11_2=t11_2) & ~(e11_1=t11_2) & ~(e11_2=t11_1)
919 %   & ~(t11_1=t11_2)
920 % )).
921 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Constitution %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
922 fof(ax_constitutedByInvolvedNatures_a58, axiom, (
923   ![X,Y,W]: (constitutedBy(X,Y,W) => ((endurant(X) <=> endurant(Y))
924     & (perdurant(X) <=> perdurant(Y)) & world(W)))
925 ).
926 fof(ax_constitutedByDifferentKinds_a59, axiom, (
927   ![E1,E2,T1,T2,W]: ((constitutedBy(E1,E2,W) & ioef(E1,T1,W) & ioef(
928     E2,T2,W) & kind(T1) & kind(T2)) => (~(T1=T2)))
929 ).
930 % Ax |= "th_noSelfConstitution_t27"; conjecture commented for
931 % convenience
932 % fof(th_noSelfConstitution_t27, conjecture, (
933 %   ~?[X,W]: (endurant(X) & constitutedBy(X,X,W))
934 % )).
935
936 fof(ax_genericConstitutionalDependence_a60, axiom, (
937   ![T1,T2]: (genericConstitutionalDependence(T1,T2) <=> (
938     type_(T1) & type_(T2) & ![E1,W]: (ioef(E1,T1,W) => (
939       ?[E2]: (constitutedBy(E1,E2,W) & ioef(E2,T2,W)
940     ))))
941   ))).
942
943
944 fof(ax_constitution_a61, axiom, (
945   ![E1,T1,E2,T2,W]: (constitution(E1,T1,E2,T2,W) <=> (
946     ioef(E1,T1,W) & ioef(E2,T2,W) & genericConstitutionalDependence(
947       T1,T2) & constitutedBy(E1,E2,W)
948   ))).
949

```

```

950 fof(
    ax_whenverAConstitutedPerdurantExistsTheConstitutedByRelationHolds_a62
    , axiom, (
951   ![P1,P2,W1]: ((constitutedBy(P1,P2,W1) & perdurant(P1)) => (![W2
      ]: (exists(P1,W2) => constitutedBy(P1,P2,W2))))
952   )).
953
954 fof(ax_constitutedByIsAsymmetric_a63, axiom, (
955   ![E1,E2,W]: (constitutedBy(E1,E2,W) => ~constitutedBy(E2,E1,W))
956   )).
957
958 % Constitution in use
959 % (tested to rule out trivial models)

```

### 2.1.12 Existential Dependence

```

961 % fof(ax_constitutionInUse, axiom, (
962 %   object(e12_1) & object(e12_2) & objectKind(k12_1) & objectKind(
      k12_2) & world(w12) & ~(k12_1=k12_2) & iof(e12_1,k12_1,w12) &
      iof(e12_2,k12_2,w12) & constitutedBy(e12_1,e12_2,w12) &
      genericConstitutionalDependence(k12_1,k12_2) & constitution(
      e12_1,k12_1,e12_2,k12_2,w12)
963 % )).
964
965 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Existential Dependence %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
966
967 fof(ax_exists_a64, axiom, (
968   ![X,W]: (exists(X,W) => (thing(X) & world(W)))
969   )).
970
971 fof(ax_existentiallyDependsOn_a65, axiom, (
972   ![X,Y]: (existentiallyDependsOn(X,Y) <=> (![W]: (exists(X,W) =>
      exists(Y,W))))
973   )).
974
975 fof(ax_existentiallyIndependentOf_a66, axiom, (
976   ![X,Y]: (existentiallyIndependentOf(X,Y) <=> (~
      existentiallyDependsOn(X,Y) & ~existentiallyDependsOn(Y,X)))
977   )).
978
979 % Existential dependence in use
980 % (tested to rule out trivial models)
981
982 % fof(ax_constitutionInUse, axiom, (
983 %   object(e13_1) & object(e13_2) & object(e13_3) & ~(e13_1=e13_2)
      & ~(e13_1=e13_3) & ~(e13_2=e13_3) & existentiallyDependsOn(
      e13_2,e13_1) & existentiallyIndependentOf(e13_3,e13_1)

```

### 2.1.13 Moments and Inherence

```

985
986 % TODO: introduce transitivity and anti-symmetry of existential
      dependence
987 % TODO: introduce continuity of existence with perdurants never
      ceasing to exist
988
989 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Moments and Inherence %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```

990
991 % Inherence
992
993 fof(ax_inherenceImpliesExistentialDependence_a67, axiom, (
994   ![M,X]: (inheresIn(M,X) => existentiallyDependsOn(M,X))
995 )).
996
997 fof(ax_thingsInvolvedInInherence_a68, axiom, (
998   ![M,X]: (inheresIn(M,X) => (moment(M) & (type_(X) | enduring(X)))
999   )
1000 )).
1001 % TODO: add definition (d5) for the "bearer" axiom
1002
1003 fof(ax_irreflexiveInherence, axiom, (
1004   ![X]: (~inheresIn(X,X))
1005 )).
1006
1007 fof(ax_asymmetricInherence, axiom, (
1008   ![X,Y]: (inheresIn(X,Y) => ~inheresIn(Y,X))
1009 )).
1010
1011 fof(ax_intransitiveInherence, axiom, (
1012   ![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(Y,Z)) => ~inheresIn(X,Z))
1013 )).
1014
1015 fof(ax_uniqueInherence_a69, axiom, (
1016   ![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(X,Z)) => (Y=Z))
1017 )).
1018
1019 % Moments
1020
1021 fof(ax_dMomentOf_d6, axiom, (
1022   ![M,X]: (momentOf(M,X) <=> (inheresIn(M,X) | (
1023     ?[M2]: (inheresIn(M,M2) & momentOf(M2,X))
1024   )))
1025 )).
1026
1027 fof(ax_dUltimateBearerOf_d7, axiom, (
1028   ![B,M]: (ultimateBearerOf(B,M) <=> (~moment(B) & momentOf(M,B)))
1029 )).
1030
1031 fof(ax_everyMomentHasUniqueUltimateBearer_a70, axiom, (
1032   ![M]: (moment(M) => (?[B]: (ultimateBearerOf(B,M) & (
1033     ![B2]: (ultimateBearerOf(B2,M) <=> (B=B2))
1034   ))))
1035 )).
1036
1037 fof(ax_noMomentOfCycles, axiom, (
1038   ~?[M]: momentOf(M,M)
1039 )).
1040
1041 % Ax |= "th_irreflexiveInherence_t28"; conjecture commented for
    convenience
1042
1043 % fof(th_irreflexiveInherence_t28, conjecture, (
1044 %   ~?[X]: (inheresIn(X,X))

```

```

1045 % )).
1046
1047 % Ax |= "th_asymmetricInherence_t29"; conjecture commented for
      convenience
1048
1049 % fof(th_asymmetricInherence_t29, conjecture, (
1050 %   ~?[X,Y]: (inheresIn(X,Y) & inheresIn(Y,X))
1051 % )).
1052
1053 % Ax |= "th_antiTransitiveInherence_t30"; conjecture commented for
      convenience
1054
1055 % fof(th_antiTransitiveInherence_t30, conjecture, (

```

#### 2.1.14 Relators

```

1057 % )).
1058
1059 % TODO: add instances
1060
1061 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Relators %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1062
1063 % External Dependence and Externally Dependent Modes
1064
1065 fof(ax_externallyDependsOn_a71, axiom, (
1066   ~?[M,X]: (externallyDependsOn(M,X) <=> (existentiallyDependsOn(M,
      X) & (![Y]: (inheresIn(M,Y) => existentiallyIndependentOf(X,Y))
      )))
1067 )).
1068
1069 fof(ax_dExternallyDependentMode_a72, axiom, (
1070   ![M]: (externallyDependentMode(M) <=> (mode(M) & (?[X]: (
      externallyDependsOn(M,X)))))
1071 )).
1072
1073 % Founded by
1074
1075 fof(ax_foundedByInvolvedThings_a73, axiom, (
1076   ![M,P]: (foundedBy(M,P) <=> ((externallyDependentMode(M) |
      relator(M)) & perdurant(P)))
1077 )).
1078
1079 fof(ax_relationalModesHaveAFoundationEvent_a74, axiom, (
1080   ![M]: ((externallyDependentMode(M) | relator(M)) => (?[P]: (
      foundedBy(M,P))))
1081 )).
1082
1083 fof(ax_uniqueFoundationEvents_a74, axiom, (
1084   ![M,P1,P2]: ((foundedBy(M,P1) & foundedBy(M,P2)) => (P1=P2))
1085 )).
1086
1087 % TODO: add definition (d8) for the "foundationOf" axiom
1088
1089 % Qua Individual
1090
1091 fof(ax_dQuaIndividualOf_a75, axiom, (
1092   ![X,Y]: (quaIndividualOf(X,Y) <=> (![Z]: (overlap(Z,X) <=> (

```

```

1093     externallyDependentMode(Z) & inheresIn(Z,Y) & (![P]: (foundedBy
1094       (X,P) => foundedBy(Z,P)))
1095   ))).
1096
1097 % Ax |= "
1098     th_thePartsOfAQuaIndividualShareTheFoundationOfTheWhole_t31";
1099     conjecture commented for convenience
1100
1101 % fof(th_thePartsOfAQuaIndividualShareTheFoundationOfTheWhole_t31,
1102     conjecture, (
1103     %[X,Y,Z]: ((quaIndividual(X) & partOf(Z,X)) => (![P]: (
1104       foundedBy(Z,P) => foundedBy(X,P))))
1105   %)).
1106
1107 fof(ax_dQuaIndividual_a76, axiom, (
1108     %[X]: (quaIndividual(X) <=> ?[Y]: (quaIndividualOf(X,Y)))
1109   %)).
1110
1111 % Qua Individual is already defined as a subtype of Externally
1112     Dependent Mode in the taxonomy; skipping (a78)
1113
1114 % Skipping (a79); already defined in (a74)
1115
1116 fof(ax_thePartsOfARelatorShareTheFoundationOfTheWhole_a80, axiom, (
1117     %[X,Y,Z]: ((relator(X) & partOf(Z,X)) => (![P]: (foundedBy(Z,P)
1118       => foundedBy(X,P))))
1119   %)).
1120
1121 fof(ax_dRelator_a81, axiom, (
1122     %[R]: (relator(R) <=> (
1123     (?[X]: (properPartOf(X,R))
1124     & (![Y,Z]: ((properPartOf(Y,R) & properPartOf(Z,R)) => (
1125       quaIndividual(Y) & quaIndividual(Z) & existentiallyDependsOn(Y,
1126       Z) & existentiallyDependsOn(Z,Y) & (![P]: (foundedBy(Y,P) <=>
1127       foundedBy(Z,P)))))
1128     & (![Y2,Z2]: ((properPartOf(Y2,R) & quaIndividual(Z2) &
1129       existentiallyDependsOn(Y2,Z2) & existentiallyDependsOn(Z2,Y2) &
1130       (![P2]: (foundedBy(Y2,P2) <=> foundedBy(Z2,P2)))) => (
1131       properPartOf(Z2,R))))
1132   %)).
1133
1134 % Ax |= "th_relatorsImplyTheExistenceOfAtLeastTwoQuaIndividuals_t32
1135     "; conjecture commented for convenience
1136
1137 % fof(th_relatorsImplyTheExistenceOfAtLeastTwoQuaIndividuals_t32,
1138     conjecture, (
1139     %[R]: (relator(R) => (?[Q1,X,Q2,Y]: (quaIndividualOf(Q1,X) &
1140       quaIndividualOf(Q2,Y) & ~(Q1=Q2))))
1141   %)).
1142
1143 fof(ax_dMediates_a82, axiom, (
1144     %[R,E]: (mediates(R,E) <=> (relator(R) & endurant(E) & (?[Q]: (
1145       quaIndividualOf(Q,E) & partOf(Q,R))))
1146   %)).
1147
1148

```

```

1133 % Ax |= "th_relatorsMediateAtLeastTwoThings_t33"; conjecture
      commented for convenience
1134
1135 % fof(th_relatorsMediateAtLeastTwoThings_t33, conjecture, (
1136 %   ![R]: (relator(R) => (?[E1,E2]: (~ (E1=E2) & mediates(R,E1) &
      mediates(R,E2))))
1137 % )).

```

### 2.1.15 Characterization

```

1139 % TODO: add definition (d9) for the "relator bearer" axiom
1140
1141 % TODO: add instances
1142
1143 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Characterization %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1144
1145 fof(ax_endurantTypeCharacterizationByMomentTypes_a83, axiom, (
1146   ![ET,MT]: (characterizes(MT,ET) => (
1147     endurantType(ET)
1148     & momentType(M)
1149     & (![E,W]: (iof(E,ET,W) => (?[M]: (iof(M,MT,W) & inheresIn(M,E)
      ))))
1150     & (![M2,W2]: (iof(M2,MT,W2) => (?[E2]: (iof(E2,ET,W2) &
      inheresIn(M2,E2))))))
1151   ))
1152 ))).
1153
1154 % Ax |= "
      th_qualitiesInheresInAUniqueEndurantConnectThroughCharacteization_a84
      "; conjecture commented for convenience
1155
1156 % fof(
      th_qualitiesInheresInAUniqueEndurantConnectThroughCharacteization_a84
      , conjecture, (

```

### 2.1.16 Qualities and Quality Structures

```

1162 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Qualities and Quality Structures %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1163
1164 % Skipping (a85); previously introduced in the taxonomy
1165 % Skipping (a86); previously introduced in the taxonomy
1166 % Skipping (a87); previously introduced in the taxonomy
1167
1168 % ZFC Set Theory
1169
1170 % TODO: we seem to require
1171
1172 % Quality Structures
1173
1174 fof(ax_dQualityStructure_d10, axiom, (
1175   ![QS]: (qualityStructure(QS) <=> (?[QT]: (qualityType(QT) &
      associatedWith(QS,QT))))
1176 ))).
1177
1178 fof(ax_dQualityStructure_d10, axiom, (
1179   ![QS]: (qualityStructure(QS) <=> (?[QT]: (qualityType(QT) &
      associatedWith(QS,QT))))
1180 ))).

```

### 2.1.17 Endurants and Perdurants

```
1182 %%%%%%%%%%%%%%%%%%%%%%%%% Endurants and Perdurants %%%%%%%%%%%%%%%%%%%%%%%%%
1183
1184 fof(ax_manifestedInInvolvedThings_a104, axiom, (
1185   ![E,P]: (manifestedIn(E,P) => (endurant(E) & perdurant(P)))
1186 )).
1187
1188 fof(ax_lifeOfInvolvedThings_a105, axiom, (
1189   ![E,P]: (lifeOf(E,P) => (
1190     endurant(E)
1191     & (![P2]: (overlap(P2,P) <=> (perdurant(P2) & manifestedIn(P2,X
1192   )))
1193 )).
1194
1195 % TODO: review ax_lifeOfInvolvedThings_a105 and its translation of
1196         the small sigma predicate schema in (a105)
1197
1198 fof(ax_meetsInvolvedThings_a106, axiom, (
1199   ![P1,P2]: (meets(P1,P2) => (perdurant(P1) & perdurant(P2)))
1200 )).
1201 % TODO: add instances
```