

A TPTP Formalization of the Unified Foundational Ontology

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

October 12, 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)))
```

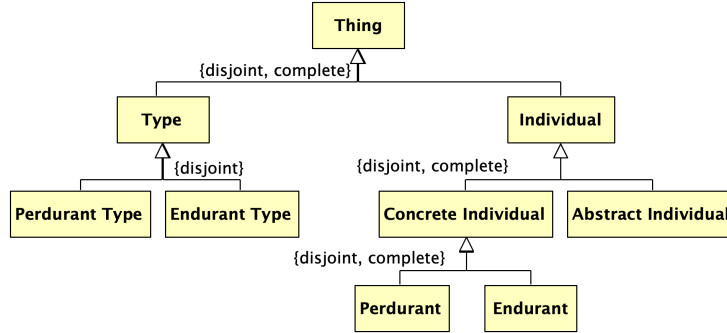


Figure 1: Partial Taxonomy of UFO – Thing.

```

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

```

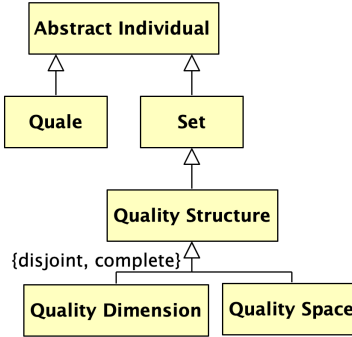


Figure 2: Partial Taxonomy of UFO – Abstract Individual.

```

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 % 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 to 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

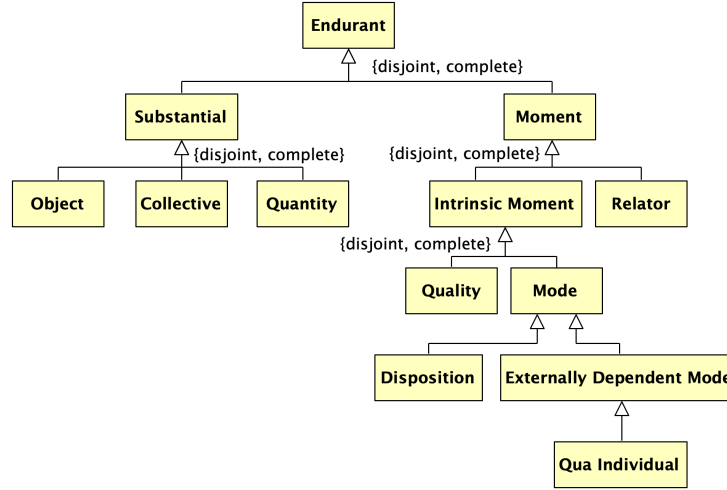


Figure 3: Partial Taxonomy of UFO – Endurant.

```

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 to 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 )).

```

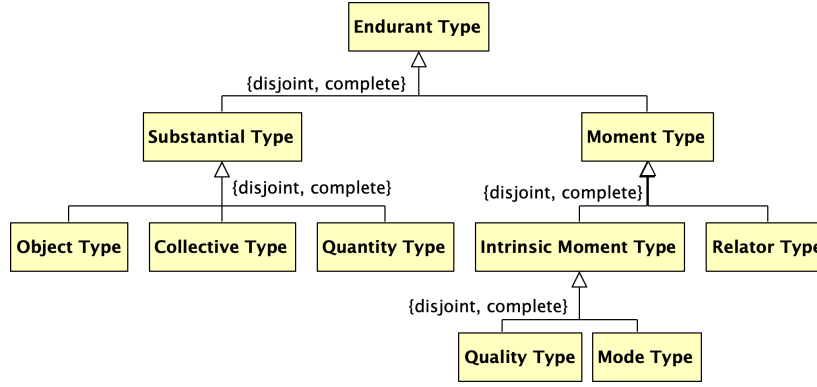


Figure 4: Partial Taxonomy of UFO – Endurant Types (by ontological nature).

```

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 to 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)
198 % )

```

194 %)) .

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

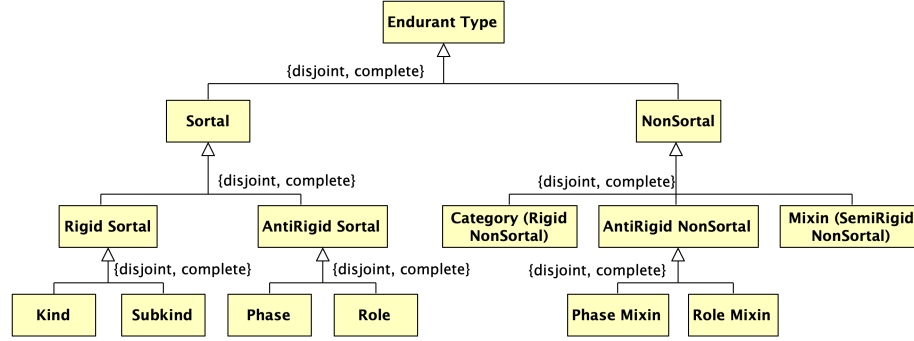


Figure 5: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```

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 to 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

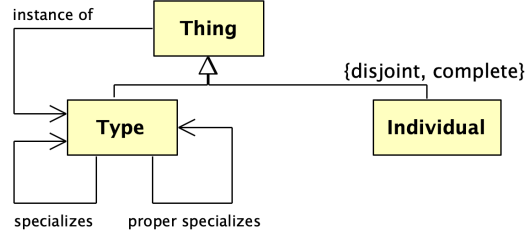


Figure 6: Types, individuals, instantiation, and specialization.

```

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 % 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 % fof(th_thingPartition_t2, conjecture, (
314 %   ~?[X]: (type_(X) & individual(X))
315 % )).
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) & ~
332     specializes(Y,X)))
333 )).
334
335 % Ax |= "th_cyclicSpecializations_t3"; conjecture commented for
336 % convenience
337
338 % fof(th_cyclicSpecializations_t3, conjecture, (
339 %   ![X,Y]: (specializes(X,Y) => (specializes(X,X) & specializes(Y,
340 %     Y)))
341 % )).
342
343 % Ax |= "th_transitiveSpecializations_t4"; conjecture commented for
344 % convenience
345
346 % fof(th_transitiveSpecializations_t4, conjecture, (
347 %   ![X,Y,Z]: ((specializes(X,Y) & specializes(Y,Z)) => (
348 %     specializes(X,Z)))
349 % )).
350
351 fof(ax_sharedSpecializations_a6, axiom, (
352   ![T1,T2]: (?[X,W]: ((iof(X,T1,W) & iof(X,T2,W) & ~specializes(T1,
353     T2) & ~specializes(T2,T1)) => (
354     (?[T3]: (specializes(T1,T3) & specializes(T2,T3) & iof(X,T3,W)
355       )))|
356     (?[T3]: (specializes(T3,T1) & specializes(T3,T2) & iof(X,T3,W)
357       )))
358 )).

```

2.1.7 Defining Rigidity and Sortality

```

354 % (tested to rule out trivial models)
355
356 % fof(ax_specializesInUse, axiom, (
357 %   endurantType(t3_1) & endurantType(t3_2) & specializes(t3_1,t3_2)
358 %   & properSpecializes(t3_1,t3_2) & specializes(t3_1,t3_1) &

```

```

    enduring(e3) & world(w3) & iof(e3,t3_1,w3)
358 %)).
359
360 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Sortality and Rigidity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
361
362 % Rigidity
363
364 % 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.
365 % TODO: verify whether it is a problem not to introduce predicates
    "world(W1) &" and "world(W2) &" before each instantiation
366
367 fof(ax_dRigid_a18, axiom, (
368     ![T]: (rigid(T) <=> (endurantType(T) & (
369         ![X]: ((?[W1]: (world(W1) & iof(X,T,W1))) => (![W2]: (world(W2)
            => iof(X,T,W2))))
370     )))
371)).
372
373 fof(ax_dAntiRigid_a19, axiom, (
374     ![T]: (antiRigid(T) <=> (endurantType(T) & (
375         ![X]: ((?[W1]: (world(W1) & iof(X,T,W1))) => (?[W2]: (world(W2)
            & ~iof(X,T,W2)))
376     ))))
377)).
378
379 fof(ax_dSemiRigid_a20, axiom, (
380     ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid
        (T)))
381)).
382
383 % Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented
    for convenience
384
385 % fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
386 %     ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid
        (T)))
387 %)).
388
389 % Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented
    for convenience
390
391 % fof(th_pairwiseDisjointRigidities_t6, conjecture, (
392 %     ~![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)
        )) | (rigid(T) & antiRigid(T)))
393 %)).
394
395 % Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture
    commented for convenience
396
397 % fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
398 %     ~![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
399 %)).
400
401 % Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8";
    conjecture commented for convenience

```

```

402
403 % fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture,
404 %   (~![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
405 %   )).
406
407 % Rigidity properties
408 % (tested to rule out trivial models)
409
410 % fof(ax_rigidityInUse, axiom, (
411 %   endurantType(t4_1) & endurantType(t4_2) & endurantType(t4_3) &
412 %   rigid(t4_1) & semiRigid(t4_2) & antiRigid(t4_3) &
413 %   properSpecializes(t4_1,t4_2) & properSpecializes(t4_3,t4_1)
414 %   )).
415
416 % Sortality
417
418 fof(ax_endurantsKind_a21, axiom, (
419   ![E]: (endurant(E) => (
420     ?[U]: (kind(U) & (![W]: (world(W) & iof(E,U,W))))
421   ))).
422
423 fof(ax_instances, axiom, (
424   endurant(ex)
425 )).
426
427 fof(ax_uniqueKind_a22, axiom, (
428   ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
429     ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
430   )))
431
432 % Changing "ax_dSortal_a23" from the form it was defined in the
433 % paper to "sortals are endurant types that specialize some
434 % ultimate sortal" seem to express the same concept while
435 % speeding up the execution of SPASS considerably
436
437 fof(ax_dSortal_a23, axiom, (
438   ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,
439     W]: (iof(E,S,W) => iof(E,U,W))))))
440 %   )).
441
442 fof(ax_dSortal_a23, axiom, (
443   ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) &
444     specializes(S,U))))
445 %   )).
446
447 % If we have the taxonomy's axiomatization, then a24 becomes a
448 % theorem
449 % Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24";
450 % conjecture commented for convenience
451
452 fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
453 %   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
454 %   )).
455
456

```

```

449 % Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience
450
451 % fof(th_kindsAreRigid_t9, conjecture, (
452 %   ![U]: ((kind(U)) => (rigid(U)))
453 % )).
454
455 % Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented
    for convenience
456
457 % fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
458 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
459 %     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2
    ,W2)))
460 %   )
461 % )).
462
463 % Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented
    for convenience
464
465 % fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
466 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
467 %     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
468 %   )
469 % )).
470
471 % Ax |= "th_kindsAreSortal_t12"; conjecture commented for
    convenience
472
473 % fof(th_kindsAreSortal_t12, conjecture, (
474 %   ![K]: ((kind(K)) => (sortal(K)))
475 % )).
476
477 % Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for
    convenience
478
479 % fof(th_sortalSpecializeKinds_t13, conjecture, (
480 %   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
481 % )).
482
483 % Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented
    for convenience
484
485 % fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
486 %   ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) &
    specializes(S,U) & specializes(S,U2) & ~(U=U2))))
487 % )).
488
489 % Sortality properties
490 % (tested to rule out trivial models)
491
492 fof(ax_sortalityInUse, axiom, (
493   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)
494 )).
495
496 % Sortality + Rigidity

```

```

497
498 fof(ax_rigidSortalsAreRigidAndSortal_xx, axiom, (
499   ![T]: ((rigidSortal(T)) <=> (rigid(T) & sortal(T)))
500 )).
501
502 fof(ax_antiRigidSortalsAreAntiRigidAndSortal_xx, axiom, (
503   ![T]: ((antiRigidSortal(T)) <=> (antiRigid(T) & sortal(T)))
504 )).
505
506 fof(ax_rigidNonSortalsAreRigidAndNonSortal_xx, axiom, (
507   ![T]: ((rigidNonSortal(T)) <=> (rigid(T) & nonSortal(T)))
508 )).
509
510 fof(ax_antiRigidNonSortalsAreAntiRigidAndNonSortal_xx, axiom, (
511   ![T]: ((antiRigidNonSortal(T)) <=> (antiRigid(T) & nonSortal(T)))
512 )).
513
514 fof(ax_semiRigidNonSortalsAreSemiRigidAndNonSortal_xx, axiom, (
515   ![T]: ((semiRigidNonSortal(T)) <=> (semiRigid(T) & nonSortal(T)))
516 )).
517
518 % If we have the taxonomy's axiomatization, then a25 becomes a
    theorem
519 % Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented
    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
    theorem
526 % Ax |= "th_kindAndSubkindAreRigidSortals_a26"; conjecture
    commented for convenience
527
528 % fof(th_kindAndSubkindAreRigidSortals_a26, conjecture, (
529 %   ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))
530 % )).
531
532 % If we have the taxonomy's axiomatization, then a27 becomes a
    theorem
533 % Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for
    convenience
534
535 % fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
536 %   ~?[T]: (phase(T) & role(T))
537 % )).
538
539 % If we have the taxonomy's axiomatization, then a28 becomes a
    theorem
540 % Ax |= "th_phaseAndRoleAreAntiRigidSortals_a28"; conjecture
    commented for convenience
541
542 % fof(th_phaseAndRoleAreAntiRigidSortals_a28, conjecture, (
543 %   ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
544 % )).
545

```

```

546 % Skipping (a29) because we leave the concept of semi-rigid sortals
      out of this ontology.
547
548 % If we have the taxonomy's axiomatization, then a30 becomes a
      theorem
549 % Ax |= "th_categoriesAreRigidNonSortals_a30"; conjecture commented
      for convenience
550
551 % fof(th_categoriesAreRigidNonSortals_a30, conjecture, (
552 %   ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))
553 % )).
554
555 % If we have the taxonomy's axiomatization, then a31 becomes a
      theorem
556 % Ax |= "th_mixinsAreSemiRigidNonSortals_a31"; conjecture commented
      for convenience
557
558 % fof(th_mixinsAreSemiRigidNonSortals_a31, conjecture, (
559 %   ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
560 % )).
561
562 % If we have the taxonomy's axiomatization, then a32 becomes a
      theorem
563 % Ax |= "th_phaseMixinAndRoleMixinAreDisjoint_a32"; conjecture
      commented for convenience
564
565 % fof(th_phaseMixinAndRoleMixinAreDisjoint_a32, conjecture, (
566 %   ~?[T]: (phaseMixin(T) & roleMixin(T))
567 % )).
568
569 % If we have the taxonomy's axiomatization, then a33 becomes a
      theorem
570 % Ax |= "ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a33";
      conjecture commented for convenience
571
572 % fof(th_phaseMixinAndRoleMixinAreAntiRigidSortals_a33, conjecture,
      (
573 %   ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) &
      nonSortal(T)))
574 % )).
575
576 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture
      commented for convenience
577
578 % fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
579 %   ~?[T]: (endurantType(T) & (
580 %     (
581 %       (kind(T) & subkind(T))
582 %       | (kind(T) & phase(T))
583 %       | (kind(T) & role(T))
584 %       | (kind(T) & category(T))
585 %       | (kind(T) & mixin(T))
586 %       | (kind(T) & phaseMixin(T))
587 %       | (kind(T) & roleMixin(T))
588 %     ) | (
589 %       (subkind(T) & phase(T))
590 %       | (subkind(T) & role(T))

```

```

591 %      | (subkind(T) & category(T))
592 %      | (subkind(T) & mixin(T))
593 %      | (subkind(T) & phaseMixin(T))
594 %      | (subkind(T) & roleMixin(T))
595 %      ) | (
596 %      (phase(T) & role(T))
597 %      | (phase(T) & category(T))
598 %      | (phase(T) & mixin(T))
599 %      | (phase(T) & phaseMixin(T))

```

2.1.8 Defining Endurant Types

```

601 %      ) | (
602 %      (role(T) & category(T))
603 %      | (role(T) & mixin(T))
604 %      | (role(T) & phaseMixin(T))
605 %      | (role(T) & roleMixin(T))
606 %      ) | (
607 %      (category(T) & mixin(T))
608 %      | (category(T) & phaseMixin(T))
609 %      | (category(T) & roleMixin(T))
610 %      ) | (
611 %      (mixin(T) & phaseMixin(T))
612 %      | (mixin(T) & roleMixin(T))
613 %      ) | (
614 %      (phaseMixin(T) & roleMixin(T))
615 %      )
616 %    ))
617 % )).
618
619 % Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
        "; conjecture commented for convenience
620
621 % fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19,
        conjecture, (
622 %   ![T]: (endurantType(T) => (
623 %     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
624 %     mixin(T) | phaseMixin(T) | roleMixin(T)
625 %   ))
626 % )).
627 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Endurant Types Definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
628
629 % fof(ax_endurantTypeDefinition_xx, axiom, (
630 %   ![T]: (endurantType(T) <=> (
631 %     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (endurant(E))
632 %   ))
633 % )).
634
635 % fof(ax_substantialTypeDefinition_xx, axiom, (
636 %   ![T]: (substantialType(T) <=> (
637 %     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (substantial(
638 %       E))))
639 % )).
640
641 % fof(ax_momentTypeDefinition_xx, axiom, (

```



```

642 %      ![T]: (momentType(T) <=> (
643 %          type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (moment(E))))
644 %      ))
645 % )).
646
647 % fof(ax_objectTypeDefinition_xx, axiom, (
648 %      ![T]: (objectType(T) <=> (
649 %          type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (object(E))))
650 %      ))
651 % )).
652
653 % fof(ax_collectiveTypeDefinition_xx, axiom, (
654 %      ![T]: (collectiveType(T) <=> (
655 %          type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (collective(E
656 %              )))
657 %      ))
658 % )).
659
660 % fof(ax_quantityTypeDefinition_xx, axiom, (
661 %      ![T]: (quantityType(T) <=> (
662 %          type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quantity(E))
663 %              ))
664 %      ))
665 % )).
666
667 % fof(ax_intrinsicMomentTypeDefinition_xx, axiom, (
668 %      ![T]: (intrinsicMomentType(T) <=> (
669 %          type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (
670 %              intrinsicMoment(E))))
671 %      ))
672 % )).
673
674 % fof(ax_relatorTypeDefinition_xx, axiom, (
675 %      ![T]: (relatorType(T) <=> (
676 %          type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (relator(E)))
677 %      ))
678 % )).

```