

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

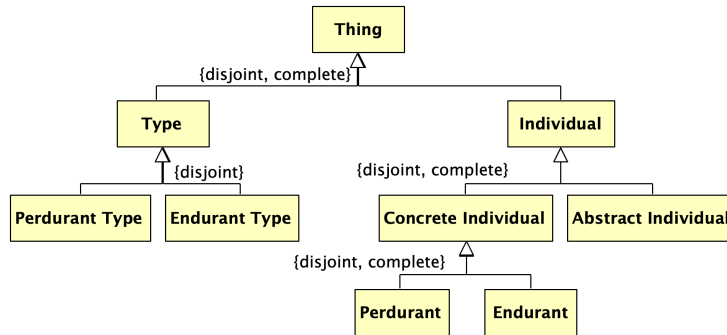


Figure 1: Partial Taxonomy of UFO – Thing.

```

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)))
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 % )).
53
54 % Abstract Individual
55
56 fof(ax_abstractIndividual_taxonomy_quale, axiom, (
57   ![X]: (quale(X) => (abstractIndividual(X)))
58 )).
59
60 fof(ax_abstractIndividual_taxonomy_set, axiom, (

```

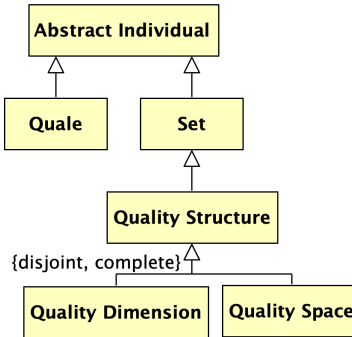


Figure 2: Partial Taxonomy of UFO – Abstract Individual.

```

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)
92  % )).
93
94  % Endurant
95
96  fof(ax_endurant_taxonomy, axiom, (

```

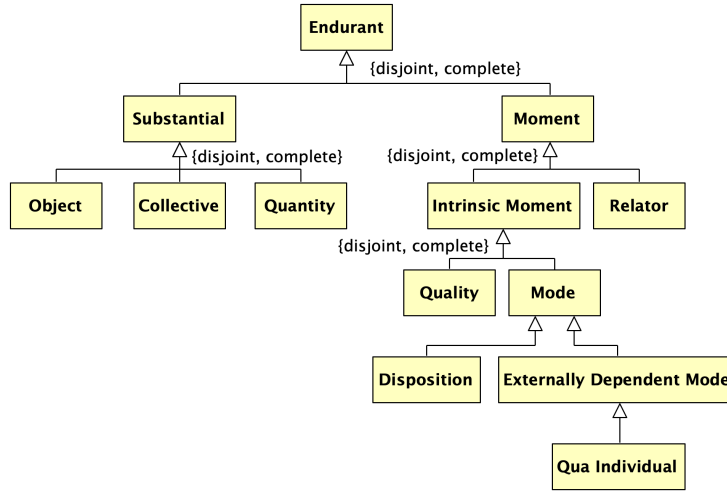


Figure 3: Partial Taxonomy of UFO – Endurant.

```

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, (
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)))

```

```

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 % ))).

```

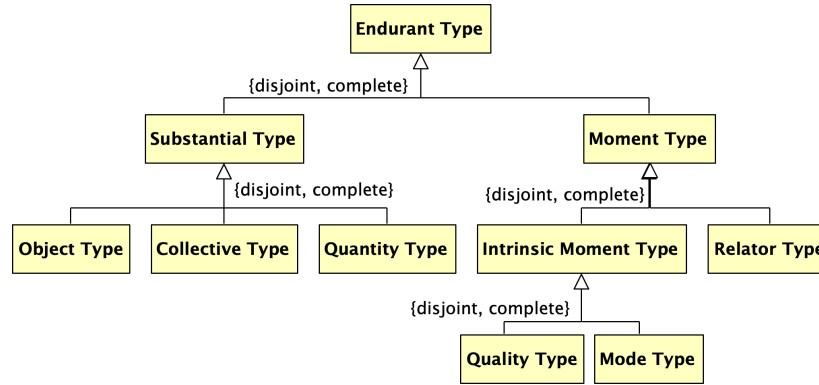


Figure 4: Partial Taxonomy of UFO – Endurant Types (by ontological nature).

```

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 fof(ax_endurantType_partition_nature, axiom, (
156   ~?[X]: (substantialType(X) & momentType(X))
157 )).
158
159 % Substantial Type
160
161 fof(ax_substantialType_taxonomy, axiom, (
162   ![X]: ((objectType(X) | collectiveType(X) | quantityType(X)) <=>
163     (substantialType(X)))
164 )).
165
166 fof(ax_substantialType_partition, axiom, (
167   ~?[X]: ((objectType(X) & collectiveType(X)) | (objectType(X) &
168     quantityType(X)) | (collectiveType(X) & quantityType(X)))
169 )).
170
171 % Moment Type
172
173 fof(ax_momentType_taxonomy, axiom, (
174   ![X]: ((intrinsicMomentType(X) | relatorType(X)) <=> (momentType(X)))
175 )).
176
177 fof(ax_momentType_partition, axiom, (
178   ~?[X]: (intrinsicMomentType(X) & relatorType(X))
179 )).
180
181 % Intrinsic Moment Type
182
183 fof(ax_intrinsicMomentType_taxonomy, axiom, (
184   ![X]: ((qualityType(X) | modeType(X)) <=> (intrinsicMomentType(X)))
185 )).
186
187 fof(ax_intrinsicMomentType_partition, axiom, (
188   ~?[X]: (qualityType(X) & modeType(X))
189 )).
190
191 % Endurant Type (by ontological nature) partial taxonomy instances
192 % (tested rule out trivial models)
193
194 % fof(ax_endurantType_instances_natures, axiom, (
195   % substantialType(substantialType1) & momentType(momentType1) &
196   % objectType(objectType1) & collectiveType(collectiveType1) &
197   % quantityType(quantityType1) & intrinsicMomentType(
198   %   intrinsicMomentType1) & relatorType(relatorType1) & qualityType
199   %   (qualityType1) & modeType(modeType1) &
200   %   externallyDependentModeType(externallyDependentModeType1) &
201   %   quaIndividualType(quaIndividualType1)
202 % )).
203
204 % Endurant Type (by modal properties of types)
205
206 fof(ax_endurantType_taxonomy_properties, axiom, (
207   ![X]: ((sortal(X) | nonSortal(X)) <=> (endurantType(X)))
208 )).
209
210 fof(ax_endurantType_partition_properties, axiom, (

```

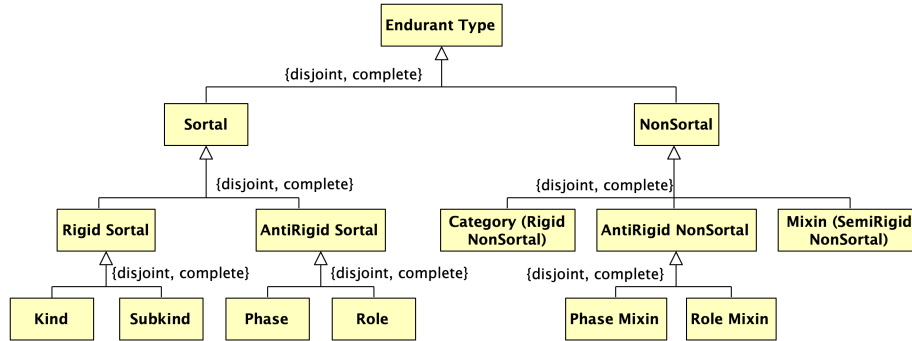


Figure 5: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```

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)))

```

```

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

```

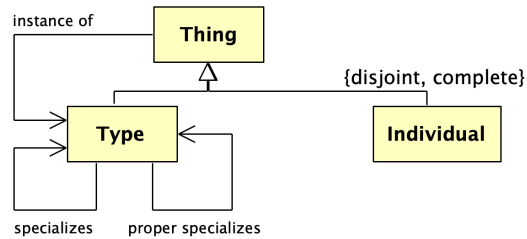


Figure 6: Types, individuals, instantiation, and specialization.


```

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 % fof(ax_specializesInUse, axiom, (
332 %   type(t3_1) & type(t3_2) & specializes(t3_1,t3_2) &
333 %   properSpecializes(t3_1,t3_2) & specializes(t3_1,t3_1)
334 % )).
335 % Ax |= "th_cyclicSpecializations_t3"; conjecture commented for
336 % convenience
337 % fof(th_cyclicSpecializations_t3, conjecture, (
338 %   ![X,Y]: (specializes(X,Y) => (specializes(X,X) & specializes(Y,
339 %   Y)))
340 % )).
341 % Ax |= "th_transitiveSpecializations_t4"; conjecture commented for
342 % convenience
343 % fof(th_transitiveSpecializations_t4, conjecture, (
344 %   ![X,Y,Z]: ((specializes(X,Y) & specializes(Y,Z)) => (
345 %   specializes(X,Z)))
346 % )).
347 fof(ax_sharedSpecializations_a6, axiom, (
348   ![T1,T2]: (?[X,W]: ((iof(X,T1,W) & iof(X,T2,W) & ~specializes(T1,
349   T2) & ~specializes(T2,T1)) => (
350   (?[T3]: (specializes(T1,T3) & specializes(T2,T3) & iof(X,T3,W)
351   )))|
352   (?[T3]: (specializes(T3,T1) & specializes(T3,T2) & iof(X,T3,W)
353   )))
354   )))
355 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Sortality and Rigidity %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
356 % TODO: I don't find we need to attach the "rigid(T)" predicate to
357 % the "endurant(T)" predicate like the paper does, so let's
358 % review this idea.
359 % TODO: verify whether it is a problem not to introduce predicates
360 % "world(W1) &" and "world(W2) &" before each instantiation
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, (

```

```

372      ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid
373      (T)))
374    )).
375    % Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented
376    for convenience
377    % fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
378    %      ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid
379    %      (T)))
380    %    )).
381    % Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented
382    for convenience
383    % fof(th_pairwiseDisjointRigidities_t6, conjecture, (
384    %      ![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)
385    %      )) | (rigid(T) & antiRigid(T)))
386    %    )).
387    % Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture
388    commented for convenience
389    % fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
390    %      ![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
391    %    )).
392    % Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8";
393    conjecture commented for convenience
394    % fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture,
395    %      (
396    %      ![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
397    %    )).
398    % fof(ax_endurantsKind_a21, axiom, (
399    %      ![E]: (endurant(E) => (
400    %        ?[U]: (kind(U) & (![W]: (world(W) & iof(E,U,W))))
401    %      ))
402    %    )).
403    % fof(ax_uniqueKind_a22, axiom, (
404    %      ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
405    %        ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
406    %      ))
407    %    )).
408    % Changing "ax_dSortal_a23" from the form it was defined in the
409    paper to "sortals are endurant types that specialize some
410    ultimate sortal" seem to express the same concept while
411    speeding up the execution of SPASS considerably
412    % fof(ax_dSortal_a23, axiom, (
413    %      ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,
414    %      W]: (iof(E,S,W) => iof(E,U,W)))))))
415    %    )).
416

```

```

417 fof(ax_dSortal_a23, axiom, (
418   ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) &
      specializes(S,U))))))
419 ))).
420
421 % If we have the taxonomy's axiomatization, then a24 becomes a
      theorem
422 % Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24";
      conjecture commented for convenience
423
424 % fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
425 %   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
426 % )).
427
428 % Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience
429
430 % fof(th_kindsAreRigid_t9, conjecture, (
431 %   ![U]: ((kind(U)) => (rigid(U)))
432 % )).
433
434 % Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented
      for convenience
435
436 % fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
437 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
438 %     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2
      ,W2)))
439 %   )
440 % )).
441
442 % Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented
      for convenience
443
444 % fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
445 %   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
446 %     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
447 %   )
448 % )).
449
450 % Ax |= "th_kindsAreSortal_t12"; conjecture commented for
      convenience
451
452 % fof(th_kindsAreSortal_t12, conjecture, (
453 %   ![K]: ((kind(K)) => (sortal(K)))
454 % )).
455
456 % Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for
      convenience
457
458 % fof(th_sortalSpecializeKinds_t13, conjecture, (
459 %   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
460 % )).
461
462 % Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented
      for convenience
463
464 % fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (

```

```
465 %      ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) &
466 %      specializes(S,U) & specializes(S,U2) & ~(U=U2))))
% ))).
```