A TPTP Formalization of the Unified Foundational Ontology

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

October 15, 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)) <=> (
       individual(X)))
```
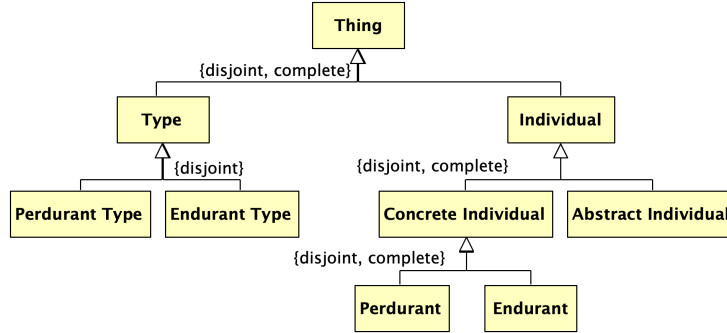
Figure 1: Partial Taxonomy of UFO – Thing.

```
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(
       concreteIndividual1) & abstractIndividual(abstractIndividual1)
       & endurant(endurant1) & perdurant(perdurant1) & endurantType(
       endurantType1) & perdurantType(perdurantType1)
49 % )).
```

2.1.2 Partial Taxonomy of Abstract Individual
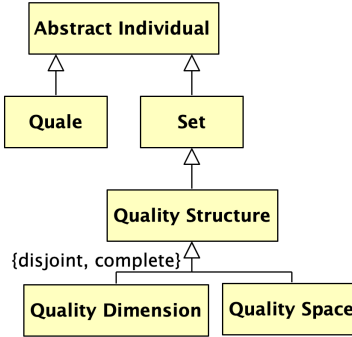
```
51 % Abstract Individual
52
```

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  % Set
66
67  fof ( ax_set_taxonomy_qualityStructure , axiom , (
68    ![X]: (qualityStructure(X) => (set_(X)))
69  )).
70
71  % Quality Structure
72
73  fof ( ax_qualityStructure_taxonomy , axiom , (
74    ![X]: ((qualityDimension(X) | qualitySpace(X)) <=> (
        qualityStructure(X)))
75  )).
76
77  fof ( ax_qualityStructure_partition , axiom , (
78    ~?[X]: (qualityDimension(X) & qualitySpace(X))
79  )).
80
81  % Abstract Individual partial taxonomy instances
82  % (tested to rule out trivial models)
83
84  % fof ( ax_abstractIndividual_instances , axiom , (
85  %    set_(set1) & quale(quale1) & qualityStructure(qualityStructure1
        ) & qualityDimension(qualityDimension1) & qualitySpace(
        qualitySpace1) & world(world1)
86  % )).
```

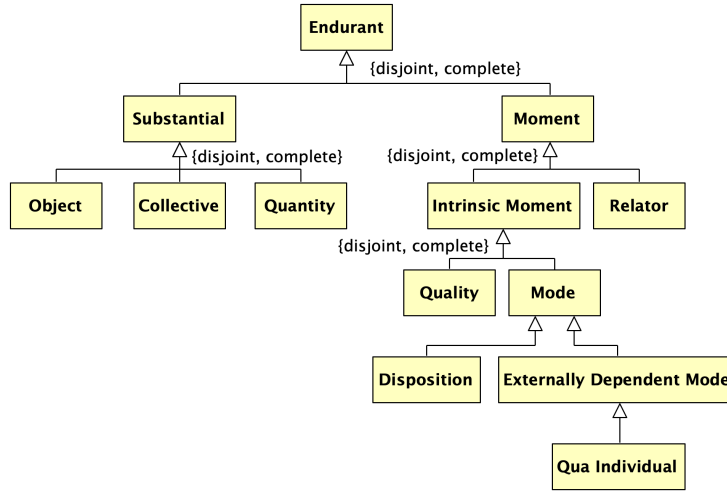### 2.1.3   Partial Taxonomy of Endurant

Figure 3: Partial Taxonomy of UFO – Endurant.

```
88  % Endurant
89
90  fof(ax_endurant_taxonomy, axiom, (
91    ![X]: ((substantial(X) | moment(X)) <=> (endurant(X)))
92  )).
93
94  fof(ax_endurant_partition, axiom, (
95    ~?[X]: (substantial(X) & moment(X))
96  )).
97
98  % Substantial
99
100 fof(ax_substantial_taxonomy, axiom, (
101   ![X]: ((object(X) | collective(X) | quantity(X)) <=> (substantial
        (X)))
102 )).
103
104 fof(ax_substantial_partition, axiom, (
105   ~?[X]: ((object(X) & collective(X)) | (object(X) & quantity(X)) |
         (collective(X) & quantity(X)))
106 )).
107
108 % Moment
109
110 fof(ax_moment_taxonomy, axiom, (
111   ![X]: ((intrinsicMoment(X) | relator(X)) <=> (moment(X)))
112 )).
113
114 fof(ax_moment_partition, axiom, (
115   ~?[X]: (intrinsicMoment(X) & relator(X))
116 )).
117
118 % Intrinsic Moment
```

4

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

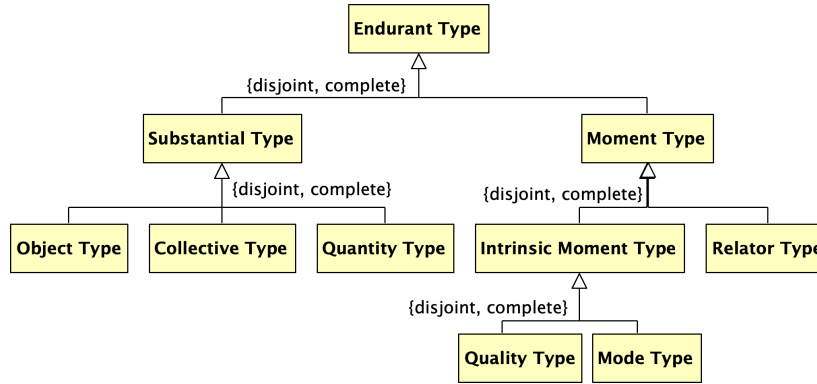### 2.1.4 Partial Taxonomy of Endurant Type (on ontological natures)

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

```
147 % Endurant Type (by ontological nature)
148
```

```
149 fof(ax_endurantType_taxonomy_nature, axiom, (
150   ![X]: ((substantialType(X) | momentType(X)) <=> (endurantType(X))
        )
151 )).
152
153 fof(ax_endurantType_partition_nature, axiom, (
154   ~?[X]: (substantialType(X) & momentType(X))
155 )).
156
157 % Substantial Type
158
159 fof(ax_substantialType_taxonomy, axiom, (
160   ![X]: ((objectType(X) | collectiveType(X) | quantityType(X)) <=>
        (substantialType(X)))
161 )).
162
163 fof(ax_substantialType_partition, axiom, (
164   ~?[X]: ((objectType(X) & collectiveType(X)) | (objectType(X) &
        quantityType(X)) | (collectiveType(X) & quantityType(X)))
165 )).
166
167 % Moment Type
168
169 fof(ax_momentType_taxonomy, axiom, (
170   ![X]: ((intrinsicMomentType(X) | relatorType(X)) <=> (momentType(
        X)))
171 )).
172
173 fof(ax_momentType_partition, axiom, (
174   ~?[X]: (intrinsicMomentType(X) & relatorType(X))
175 )).
176
177 % Intrinsic Moment Type
178
179 fof(ax_intrinsicMomentType_taxonomy, axiom, (
180   ![X]: ((qualityType(X) | modeType(X)) <=> (intrinsicMomentType(X)
        ))
181 )).
182
183 fof(ax_intrinsicMomentType_partition, axiom, (
184   ~?[X]: (qualityType(X) & modeType(X))
185 )).
186
187 % Endurant Type (by ontological nature) partial taxonomy instances
188 % (tested to rule out trivial models)
189
190 % fof(ax_endurantType_instances_natures, axiom, (
191 %   substantialType(substantialType1) & momentType(momentType1) &
        objectType(objectType1) & collectiveType(collectiveType1) &
        quantityType(quantityType1) & intrinsicMomentType(
        intrinsicMomentType1) & relatorType(relatorType1) & qualityType
        (qualityType1) & modeType(modeType1)
192 % )).
```

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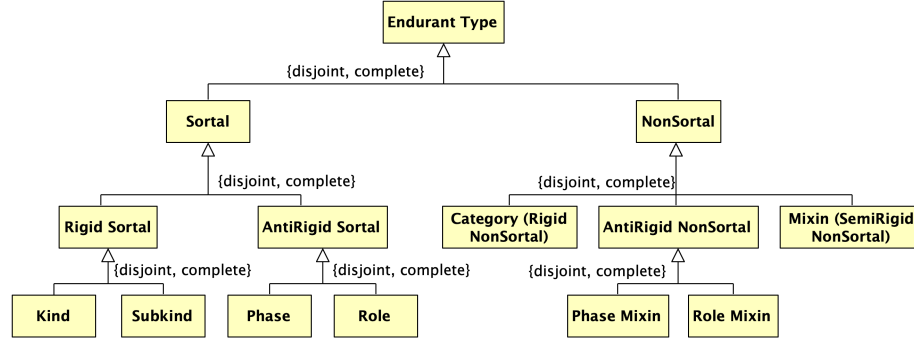

Figure 5: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```
194  % Endurant Type (by modal properties of types)
195
196  fof(ax_endurantType_taxonomy_properties, axiom, (
197    ![X]: ((sortal(X) | nonSortal(X)) <=> (endurantType(X)))
198  )).
199
200  fof(ax_endurantType_partition_properties, axiom, (
201    ~?[X]: (sortal(X) & nonSortal(X))
202  )).
203
204  % Sortal
205
206  fof(ax_sortal_taxonomy, axiom, (
207    ![X]: ((rigidSortal(X) | antiRigidSortal(X)) <=> (sortal(X)))
208  )).
209
210  fof(ax_sortal_partition, axiom, (
211    ~?[X]: (rigidSortal(X) & antiRigidSortal(X))
212  )).
213
214  % Rigid Sortal
215
216  fof(ax_rigidSortal_taxonomy, axiom, (
217    ![X]: ((kind(X) | subkind(X)) <=> (rigidSortal(X)))
218  )).
219
220  fof(ax_rigidSortal_partition, axiom, (
221    ~?[X]: (kind(X) & subkind(X))
222  )).
223
224  % Anti-Rigid Sortal
225
226  fof(ax_antiRigidSortal_taxonomy, axiom, (
227    ![X]: ((phase(X) | role(X)) <=> (antiRigidSortal(X)))
```

```
228 )).
229
230 fof(ax_antiRigidSortal_partition, axiom, (
231   ~?[X]: (phase(X) & role(X))
232 )).
233
234 % Non-Sortal
235
236 fof(ax_nonSortal_taxonomy, axiom, (
237   ![X]: ((rigidNonSortal(X) | semiRigidNonSortal(X) |
        antiRigidNonSortal(X)) <=> (nonSortal(X)))
238 )).
239
240 fof(ax_nonSortal_partition, axiom, (
241   ~?[X]: ((rigidNonSortal(X) & semiRigidNonSortal(X)) | (
        rigidNonSortal(X) & antiRigidNonSortal(X)) | (
        semiRigidNonSortal(X) & antiRigidNonSortal(X)))
242 )).
243
244 % Category
245
246 fof(ax_rigidNonSortal_taxonomy, axiom, (
247   ![X]: (rigidNonSortal(X) <=> (category(X)))
248 )).
249
250 % Mixin
251
252 fof(ax_semiRigidNonSortal_taxonomy, axiom, (
253   ![X]: (semiRigidNonSortal(X) <=> (mixin(X)))
254 )).
255
256 % Anti-Rigid Non-Sortal
257
258 fof(ax_antiRigidNonSortal_taxonomy, axiom, (
259   ![X]: ((phaseMixin(X) | roleMixin(X)) <=> (antiRigidNonSortal(X))
        )
260 )).
261
262 fof(ax_antiRigidNonSortal_partition, axiom, (
263   ~?[X]: (phaseMixin(X) & roleMixin(X))
264 )).
265
266 % Endurant Type (by modal properties of types) partial taxonomy
        instances
267 % (tested to rule out trivial models)
268
269 % fof(ax_endurantType_instances_properties, axiom, (
270 %   sortal(sortal1) & nonSortal(nonSortal1) & rigidSortal(
        rigidSortal1) & antiRigidSortal(antiRigidSortal1) & kind(kind1)
         & subkind(subkind1) & phase(phase1) & role(role1) & category(
        category1) & mixin(mixin1) & antiRigidNonSortal(
        antiRigidNonSortal1) & phaseMixin(phaseMixin1) & roleMixin(
        roleMixin1)
271 % )).
```

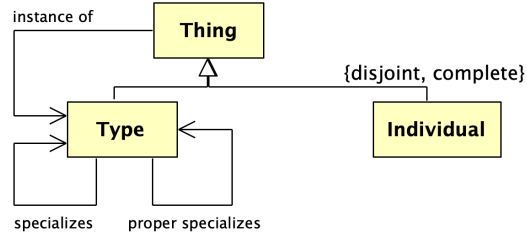### 2.1.6   Defining Types, Individuals, and Specialization

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

```
273  %%%%%%%%%% Instance of, Types, and Individuals %%%%%%%%%%
274
275  fof(ax_dIof, axiom, (
276    ![X,Y,W]: (iof(X,Y,W) => (type_(Y) & world(W)))
277  )).
278
279  fof(ax_dType_a1, axiom, (
280    ![X]: (type_(X) <=> (?[Y,W]: iof(Y,X,W)))
281  )).
282
283  fof(ax_dIndividual_a2, axiom, (
284    ![X]: (individual(X) <=> (~?[Y,W]: iof(Y,X,W)))
285  )).
286
287  % TODO: confirm whether we are including second-order types in this
            formalization
288
289  fof(ax_multiLevel_a3, axiom, (
290    ![X,Y,W]: (iof(X,Y,W) => (type_(X) | individual(X)))
291  )).
292
293  fof(ax_twoLevelConstrained_a4, axiom, (
294    ~?[X,Y,Z,W]: (type_(X) & iof(X,Y,W) & iof(Y,Z,W))
295  )).
296
297  % Instantiation relations
298  % (tested to rule out trivial models)
299
300  % fof(ax_iofInUse, axiom, (
301  %   type_(t2) & individual(i2) & world(w2) & iof(i2,t2,w2)
302  % )).
303
304  % Ax |= "th_everythingIsAThing_t1"; conjecture commented for
        convenience
305
306  % fof(th_everythingIsAThing_t1, conjecture, (
307  %   ![X]: (type_(X) | individual(X))
308  % )).
309
310  % Ax |= "th_thingPartition_t2"; conjecture commented for
        convenience
```

```
311
312 % fof(th_thingPartition_t2, conjecture, (
313 %    ~?[X]: (type_(X) & individual(X))
314 % )).
315
316 %%%%%%% Specialization and Proper Specialization %%%%%%%
317
318 fof(ax_dSpecializes, axiom, (
319   ![X,Y]: (specializes(X,Y) => (type_(X) & type_(Y)))
320 )).
321
322 fof(ax_specialization_a5, axiom, (
323   ![T1,T2]: (specializes(T1,T2) <=> (
324     type_(T1) & type_(T2) & ![W]: (world(W) => ![E]: (iof(E,T1,W)
        => iof(E,T2,W)))
325   ))
326 )).
327
328 fof(ax_properSpecializes_d1, axiom, (
329   ![X,Y]: (properSpecializes(X,Y) <=> (specializes(X,Y) & ~
        specializes(Y,X)))
330 )).
331
332 % Ax |= "th_cyclicSpecializations_t3"; conjecture commented for
      convenience
333
334 % fof(th_cyclicSpecializations_t3, conjecture, (
335 %   ![X,Y]: (specializes(X,Y) => (specializes(X,X) & specializes(Y,
      Y)))
336 % )).
337
338 % Ax |= "th_transitiveSpecializations_t4"; conjecture commented for
       convenience
339
340 % fof(th_transitiveSpecializations_t4, conjecture, (
341 %   ![X,Y,Z]: ((specializes(X,Y) & specializes(Y,Z)) => (
      specializes(X,Z)))
342 % )).
343
344 fof(ax_sharedSpecializations_a6, axiom, (
345   ![T1,T2]: (?[X,W]: ((iof(X,T1,W) & iof(X,T2,W) & ~specializes(T1,
      T2) & ~specializes(T2,T1)) => (
346       (?[T3]: (specializes(T1,T3) & specializes(T2,T3) & iof(X,T3,W
      )))|
347       (?[T3]: (specializes(T3,T1) & specializes(T3,T2) & iof(X,T3,W
      )))
348   )))
349 )).
350
351 % Specialization relations
352 % (tested to rule out trivial models)
353
354 % fof(ax_specializesInUse, axiom, (
355 %   endurantType(t3_1) & endurantType(t3_2) & specializes(t3_1,t3_2
      ) & properSpecializes(t3_1,t3_2) & specializes(t3_1,t3_1) &
      endurant(e3) & world(w3) & iof(e3,t3_1,w3)
356 % )).
```

2.1.7 Defining Rigidity and Sortality

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

```
402 %    ~![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
403 % )).
404
405 % Rigidity properties
406 % (tested to rule out trivial models)
407
408 % fof(ax_rigidityInUse, axiom, (
409 %    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)
410 % )).
411
412 % Sortality
413
414 fof(ax_endurantsKind_a21, axiom, (
415   ![E]: (endurant(E) => (
416     ?[U]: (kind(U) & (![W]: (world(W) => iof(E,U,W)))))
417   ))
418 )).
419
420 fof(ax_uniqueKind_a22, axiom, (
421   ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
422     ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
423   ))
424 )).
425
426 % Changing "ax_dSortal_a23" from the form it was defined in the
        paper to "sortals are endurant types that specialize some
        ultimate sortal" seem to express the same concept while
        speeding up the execution of SPASS considerably
427
428 % fof(ax_dSortal_a23, axiom, (
429 %    ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,
        W]: (iof(E,S,W) => iof(E,U,W)))))))
430 % )).
431
432 fof(ax_dSortal_a23, axiom, (
433   ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) &
        specializes(S,U)))))
434 )).
435
436 % If we have the taxonomy's axiomatization, then a24 becomes a
        theorem
437 % Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24";
        conjecture commented for convenience
438
439 % fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
440 %    ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
441 % )).
442
443 % Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience
444
445 % fof(th_kindsAreRigid_t9, conjecture, (
446 %    ![U]: ((kind(U)) => (rigid(U)))
447 % )).
448
449 % Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented
```

```
          for convenience
450
451 % fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
452 %    ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
453 %      ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2
      ,W2)))
454 %    )
455 % )).
456
457 % Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented
      for convenience
458
459 % fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
460 %    ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
461 %      ~?[T]: (specializes(T,K1) & specializes(T,K2)))
462 %    )
463 % )).
464
465 % Ax |= "th_kindsAreSortal_t12"; conjecture commented for
      convenience
466
467 % fof(th_kindsAreSortal_t12, conjecture, (
468 %    ![K]: ((kind(K)) => (sortal(K)))
469 % )).
470
471 % Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for
      convenience
472
473 % fof(th_sortalSpecializeKinds_t13, conjecture, (
474 %    ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
475 % )).
476
477 % Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented
       for convenience
478
479 % fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
480 %    ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) &
      specializes(S,U) & specializes(S,U2) & ~(U=U2))))
481 % )).
482
483 % Sortality properties
484 % (tested to rule out trivial models)
485
486 % fof(ax_sortalityInUse, axiom, (
487 %    endurant(e5_1) & endurant(e5_2) & world(w5) & kind(k5_1) & kind
      (k5_2) & iof(e5_1,k5_1,w5) & iof(e5_1,k5_1,w5) & ~(k5_1=k5_2)
488 % )).
489
490 % Sortality + Rigidity
491
492 fof(ax_rigidSortalsAreRigidAndSortal_xx, axiom, (
493   ![T]: ((rigidSortal(T)) <=> (rigid(T) & sortal(T)))
494 )).
495
496 fof(ax_antiRigidSortalsAreAntiRigidAndSortal_xx, axiom, (
497   ![T]: ((antiRigidSortal(T)) <=> (antiRigid(T) & sortal(T)))
498 )).
```

```
499
500 fof(ax_rigidNonSortalsAreRigidAndNonSortal_xx, axiom, (
501   ![T]: ((rigidNonSortal(T)) <=> (rigid(T) & nonSortal(T)))
502 )).
503
504 fof(ax_antiRigidNonSortalsAreAntiRigidAndNonSortal_xx, axiom, (
505   ![T]: ((antiRigidNonSortal(T)) <=> (antiRigid(T) & nonSortal(T)))
506 )).
507
508 fof(ax_semiRigidNonSortalsAreSemiRigidAndNonSortal_xx, axiom, (
509   ![T]: ((semiRigidNonSortal(T)) <=> (semiRigid(T) & nonSortal(T)))
510 )).
511
512 % If we have the taxonomy's axiomatization, then a25 becomes a
         theorem
513 % Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented
       for convenience
514
515 % fof(th_kindAndSubkindAreDisjoint_a25, conjecture, (
516 %   ~?[T]: (kind(T) & subkind(T))
517 % )).
518
519 % If we have the taxonomy's axiomatization, then a26 becomes a
         theorem
520 % Ax |= "th_kindAndSubkindAreRigidSortals_a26"; conjecture
       commented for convenience
521
522 % fof(th_kindAndSubkindAreRigidSortals_a26, conjecture, (
523 %   ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))
524 % )).
525
526 % If we have the taxonomy's axiomatization, then a27 becomes a
         theorem
527 % Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for
       convenience
528
529 % fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
530 %   ~?[T]: (phase(T) & role(T))
531 % )).
532
533 % If we have the taxonomy's axiomatization, then a28 becomes a
         theorem
534 % Ax |= "th_phaseAndRoleAreAntiRigidSortals_a28"; conjecture
       commented for convenience
535
536 % fof(th_phaseAndRoleAreAntiRigidSortals_a28, conjecture, (
537 %   ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
538 % )).
539
540 % Skipping (a29) because we leave the concept of semi-rigid sortals
         out of this ontology.
541
542 % If we have the taxonomy's axiomatization, then a30 becomes a
         theorem
543 % Ax |= "th_categoriesAreRigidNonSortals_a30"; conjecture commented
         for convenience
544
```

```
545 % fof(th_categoriesAreRigidNonSortals_a30, conjecture, (
546 %    ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))
547 % )).
548
549 % If we have the taxonomy's axiomatization, then a31 becomes a
         theorem
550 % Ax |= "th_mixinsAreSemiRigidNonSortals_a31"; conjecture commented
          for convenience
551
552 % fof(th_mixinsAreSemiRigidNonSortals_a31, conjecture, (
553 %    ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
554 % )).
555
556 % If we have the taxonomy's axiomatization, then a32 becomes a
         theorem
557 % Ax |= "th_phaseMixinAndRoleMixinAreDisjoint_a32"; conjecture
         commented for convenience
558
559 % fof(th_phaseMixinAndRoleMixinAreDisjoint_a32, conjecture, (
560 %    ~?[T]: (phaseMixin(T) & roleMixin(T))
561 % )).
562
563 % If we have the taxonomy's axiomatization, then a33 becomes a
         theorem
564 % Ax |= "ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a33";
         conjecture commented for convenience
565
566 % fof(th_phaseMixinAndRoleMixinAreAntiRigidSortals_a33, conjecture,
          (
567 %    ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) &
         nonSortal(T)))
568 % )).
569
570 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture
         commented for convenience
571
572 % fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
573 %    ~?[T]: (endurantType(T) & (
574 %      (
575 %        (kind(T) & subkind(T))
576 %        | (kind(T) & phase(T))
577 %        | (kind(T) & role(T))
578 %        | (kind(T) & category(T))
579 %        | (kind(T) & mixin(T))
580 %        | (kind(T) & phaseMixin(T))
581 %        | (kind(T) & roleMixin(T))
582 %      ) | (
583 %        (subkind(T) & phase(T))
584 %        | (subkind(T) & role(T))
585 %        | (subkind(T) & category(T))
586 %        | (subkind(T) & mixin(T))
587 %        | (subkind(T) & phaseMixin(T))
588 %        | (subkind(T) & roleMixin(T))
589 %      ) | (
590 %        (phase(T) & role(T))
591 %        | (phase(T) & category(T))
592 %        | (phase(T) & mixin(T))
```

```
593 %         | (phase(T) & phaseMixin(T))
594 %         | (phase(T) & roleMixin(T))
595 %      ) | (
596 %        (role(T) & category(T))
597 %         | (role(T) & mixin(T))
598 %         | (role(T) & phaseMixin(T))
599 %         | (role(T) & roleMixin(T))
600 %      ) | (
601 %        (category(T) & mixin(T))
602 %         | (category(T) & phaseMixin(T))
603 %         | (category(T) & roleMixin(T))
604 %      ) | (
605 %        (mixin(T) & phaseMixin(T))
606 %         | (mixin(T) & roleMixin(T))
607 %      ) | (
608 %        (phaseMixin(T) & roleMixin(T))
609 %      )
610 %   ))
611 % )).
612
613 % Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
       "; conjecture commented for convenience
614
615 % fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19 ,
       conjecture , (
616 %   ![T]: (endurantType(T) => (
617 %     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
       mixin(T) | phaseMixin(T) | roleMixin(T)
618 %   ))
619 % )).
620
621 % Sortality and rigidity properties combined
622 % (tested to rule out trivial models)
623
624 % fof(ax_sortalityAndRigidityInUse , axiom , (
625 %   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)
626 % )).
```

2.1.8   Defining Endurant Types

```
628 %%%%%%%%%%%%%%%%%%%% Types Definition %%%%%%%%%%%%%%%%%%%%
629
630 % Defining the taxonomy of types of ontological natures through the
         categorization of the taxonomy of concrete individuals
631
632 fof(ax_perdurantTypeDefinition_a44 , axiom , (
633   ![T]: (perdurantType(T) <=> (
634     type_(T) & (![P,W]: ((world(W) & iof(P,T,W)) => (perdurant(P)))
       )
635   ))
636 )).
637
638 fof(ax_endurantTypeDefinition_a44 , axiom , (
639   ![T]: (endurantType(T) <=> (
640     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (endurant(E))))
641   ))
642 )).
```

16

```
643
644 fof(ax_substantialTypeDefinition_a44 , axiom , (
645   ![T]: (substantialType(T) <=> (
646     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (substantial(E)
          )))
647   ))
648 )).
649
650 fof(ax_momentTypeDefinition_a44 , axiom , (
651   ![T]: (momentType(T) <=> (
652     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (moment(E))))
653   ))
654 )).
655
656 fof(ax_objectTypeDefinition_a44 , axiom , (
657   ![T]: (objectType(T) <=> (
658     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (object(E))))
659   ))
660 )).
661
662 fof(ax_collectiveTypeDefinition_a44 , axiom , (
663   ![T]: (collectiveType(T) <=> (
664     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (collective(E))
          ))
665   ))
666 )).
667
668 fof(ax_quantityTypeDefinition_a44 , axiom , (
669   ![T]: (quantityType(T) <=> (
670     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quantity(E))))
671   ))
672 )).
673
674 fof(ax_intrinsicMomentTypeDefinition_a44 , axiom , (
675   ![T]: (intrinsicMomentType(T) <=> (
676     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (
          intrinsicMoment(E))))
677   ))
678 )).
679
680 fof(ax_relatorTypeDefinition_a44 , axiom , (
681   ![T]: (relatorType(T) <=> (
682     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (relator(E))))
683   ))
684 )).
685
686 fof(ax_qualityTypeDefinition_a44 , axiom , (
687   ![T]: (qualityType(T) <=> (
688     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (quality(E))))
689   ))
690 )).
691
692 fof(ax_modeTypeDefinition_a44 , axiom , (
693   ![T]: (modeType(T) <=> (
694     type_(T) & (![E,W]: ((world(W) & iof(E,T,W)) => (mode(E))))
695   ))
696 )).
```

```
697
698 % Types Definition
699 % (tested to rule out trivial models)
700 % TODO: investigate why we cannot list four different endurant
        types (it may have something to do with "intrinsicMoment" and "
        intrinsicMomentType")
701
702 % fof(ax_typesDefinitionsInstances, axiom, (
703 %   objectType(ot7) & collectiveType(ct7) & modeType(mt7)
704 % )).
705
706 % Ax |= "th_leafCategoriesArePairwiseDisjoint_t21"; conjecture
        commented for convenience
707 % Having the previously defined taxonomy, this should be quite
        trivial
708
709 % fof(th_leafCategoriesArePairwiseDisjoint_t21, conjecture, (
710 %   ~?[T]: (type_(T) & (
711 %     (
712 %       (objectType(T) & collectiveType(T)) | (objectType(T) &
        quantityType(T)) | (objectType(T) & modeType(T)) | (objectType(
        T) & qualityType(T)) | (objectType(T) & relatorType(T)) | (
        objectType(T) & perdurantType(T))
713 %     ) | (
714 %       (collectiveType(T) & quantityType(T)) | (collectiveType(T)
        & modeType(T)) | (collectiveType(T) & qualityType(T)) | (
        collectiveType(T) & relatorType(T)) | (collectiveType(T) &
        perdurantType(T))
715 %     ) | (
716 %       (quantityType(T) & modeType(T)) | (quantityType(T) &
        qualityType(T)) | (quantityType(T) & relatorType(T)) | (
        quantityType(T) & perdurantType(T))
717 %     ) | (
718 %       (modeType(T) & qualityType(T)) | (modeType(T) & relatorType
        (T)) | (modeType(T) & perdurantType(T))
719 %     ) | (
720 %       (qualityType(T) & relatorType(T)) | (qualityType(T) &
        perdurantType(T))
721 %     ) | (
722 %       relatorType(T) & perdurantType(T)
723 %     )
724 %   ))
725 % )).
726
727 % Ultimate Sortals Definitions (by ontological nature)
728
729 fof(ax_objectKindDefinition_a45, axiom, (
730   ![T]: (objectKind(T) <=> (objectType(T) & kind(T)))
731 )).
732
733 fof(ax_collectiveKindDefinition_a45, axiom, (
734   ![T]: (collectiveKind(T) <=> (collectiveType(T) & kind(T)))
735 )).
736
737 fof(ax_quantityKindDefinition_a45, axiom, (
738   ![T]: (quantityKind(T) <=> (quantityType(T) & kind(T)))
739 )).
```

```
740
741  fof(ax_modeKindDefinition_a45, axiom, (
742    ![T]: (modeKind(T) <=> (modeType(T) & kind(T)))
743  )).
744
745  fof(ax_qualityKindDefinition_a45, axiom, (
746    ![T]: (qualityKind(T) <=> (qualityType(T) & kind(T)))
747  )).
748
749  fof(ax_relatorKindDefinition_a45, axiom, (
750    ![T]: (relatorKind(T) <=> (relatorType(T) & kind(T)))
751  )).
752
753  % Ultimate sortals (by ontological nature) instances
754  % (tested to rule out trivial models)
755  % TODO: investigate why we cannot list all different types of
         ultimate sortals at once
756
757  % fof(ax_typesDefinitionsInstances, axiom, (
758  %   objectKind(ok9) & collectiveKind(ck9) & quantityKind(quank9) &
         relatorKind(rk9) & modeKind(mk9) & qualityKind(qualk9)
759  % )).
760
761  % Skipping (t22) because (a21) makes it trivial
762
763  % Ax |= "th_endurantsInstantiateEndurantKindsOfSomeNature_a46";
         conjecture commented for convenience
764  % This axiom is actually a theorem in this version of the
         axiomatization
765
766  % fof(th_endurantsInstantiateEndurantKindsOfSomeNature_a46,
         conjecture, (
767  %   ![E]: (endurant(E) => (
768  %     ?[K,W]: ((objectKind(K) | collectiveKind(K) | quantityKind(K)
         | modeKind(K) | qualityKind(K) | relatorKind(K))
769  %     & iof(E,K,W))
770  %   ))
771  % )).
772
773  % Ax |= "th_endurantSortalsCompleteness_t23"; conjecture commented
         for convenience
774  % Thanks to the taxonomy, we already have "sortal(T) =>
         endurantType(T)", but I leave it like this to be consistent
         with the paper
775
776  % fof(th_endurantSortalsCompleteness_t23, conjecture, (
777  %   ![T]: ((endurantType(T) & sortal(T)) => (objectKind(T) |
         collectiveKind(T) | quantityKind(T) | qualityKind(T) | modeKind
         (T) | relatorKind(T) | phase(T) | role(T)))
778  % )).
779
780  % Ax |= "th_objectTypesSpecializeAKindOfSameNature_t24"; conjecture
          commented for convenience
781
782  % fof(th_objectTypesSpecializeAKindOfSameNature_t24, conjecture, (
783  %   ![T]: ((objectType(T) & sortal(T)) <=> (?[K]: (objectKind(K) &
         specializes(T,K))))
```

```
784 % )).
785
786 % Ax |= "th_collectiveTypesSpecializeAKindOfSameNature_t24";
       conjecture commented for convenience
787
788 % fof(th_collectiveTypesSpecializeAKindOfSameNature_t24, conjecture
       , (
789 %   ![T]: ((collectiveType(T) & sortal(T)) <=> (?[K]: (
       collectiveKind(K) & specializes(T,K))))
790 % )).
791
792 % Ax |= "th_quantityTypesSpecializeAKindOfSameNature_t24";
       conjecture commented for convenience
793
794 % fof(th_quantityTypesSpecializeAKindOfSameNature_t24, conjecture,
       (
795 %   ![T]: ((quantityType(T) & sortal(T)) <=> (?[K]: (quantityKind(K
       ) & specializes(T,K))))
796 % )).
797
798 % Ax |= "th_modeTypesSpecializeAKindOfSameNature_t24"; conjecture
       commented for convenience
799
800 % fof(th_modeTypesSpecializeAKindOfSameNature_t24, conjecture, (
801 %   ![T]: ((modeType(T) & sortal(T)) <=> (?[K]: (modeKind(K) &
       specializes(T,K))))
802 % )).
803
804 % Ax |= "th_qualityTypesSpecializeAKindOfSameNature_t24";
       conjecture commented for convenience
805
806 % fof(th_qualityTypesSpecializeAKindOfSameNature_t24, conjecture, (
807 %   ![T]: ((qualityType(T) & sortal(T)) <=> (?[K]: (qualityKind(K)
       & specializes(T,K))))
808 % )).
809
810 % Ax |= "th_relatorTypesSpecializeAKindOfSameNature_t24";
       conjecture commented for convenience
811
812 % fof(th_relatorTypesSpecializeAKindOfSameNature_t24, conjecture, (
813 %   ![T]: ((relatorType(T) & sortal(T)) <=> (?[K]: (relatorKind(K)
       & specializes(T,K))))
814 % )).
815
816 % Ax |= "th_sortalLeafCategoriesAreDisjoint_t25"; conjecture
       commented for convenience
817
818 % fof(th_sortalLeafCategoriesAreDisjoint_t25, conjecture, (
819 %   ![T]: (objectKind(T) => (~(collectiveKind(T) | quantityKind(T)
       | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T) |
       mixin(T) | phaseMixin(T) | roleMixin(T))))
820 %   & ![T]: (collectiveKind(T) => (~(objectKind(T) | quantityKind(T
       ) | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T)
       | mixin(T) | phaseMixin(T) | roleMixin(T))))
821 %   & ![T]: (quantityKind(T) => (~(objectKind(T) | collectiveKind(T
       ) | modeKind(T) | qualityKind(T) | relatorKind(T) | category(T)
       | mixin(T) | phaseMixin(T) | roleMixin(T))))
```

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

2.1.9   Mereology

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

```
857  fof(ax_transitiveParthood_a49, axiom, (
858    ![X,Y,Z]: ((partOf(X,Y) & partOf(Y,Z)) => (partOf(X,Z)))
859  )).
860
861  fof(ax_overlappingWholes_a50, axiom, (
862    ![X,Y]: ((overlap(X,Y)) <=> (?[Z]: (partOf(Z,X) & partOf(Z,Y))))
863  )).
864
865  fof(ax_strongSupplementation_a51, axiom, (
866    ![X,Y]: (~partOf(X,Y) <=> ?[Z]: (partOf(Z,X) & ~overlap(Z,Y)))
867  )).
868
869  fof(ax_properPart_a52, axiom, (
870    ![X,Y]: (~properPartOf(X,Y) <=> (partOf(X,Y) & ~partOf(Y,X)))
871  )).
872
873  fof(ax_binarySum_a53, axiom, (
874    ![X,Y,Z]: (sum(Z,X,Y) <=> ![W]: (overlap(W,Z) <=> (overlap(W,X) |
          overlap(W,Y))))
875  )).
876
877  fof(ax_binarySum_a53, axiom, (
878    ![X,Y,Z]: (sum(Z,X,Y) <=> ![W]: (overlap(W,Z) <=> (overlap(W,X) |
          overlap(W,Y))))
879  )).
880
881  % TODO: check whether it is necessary to introduce fusion and
          existence of sums, and how to do it
882
883  % Mereology in use
884  % (tested to rule out trivial models)
885
886  % fof(ax_mereologyInUse, axiom, (
887  %   concreteIndividual(ci10_1) & concreteIndividual(ci10_2) &
          concreteIndividual(ci10_3) & concreteIndividual(ci10_4) &
          concreteIndividual(ci10_5) & ~(ci10_1=ci10_2) & ~(ci10_2=ci10_3
          ) & ~(ci10_3=ci10_4) & ~(ci10_4=ci10_5) & properPart(ci10_1,
          ci10_2) & properPart(ci10_3,ci10_4) & sum(ci10_5,ci10_3,ci10_4)
888  % )).
```

2.1.10   Composition

```
890  %%%%%%%%%%%%%%%%%%%%%%% Composition %%%%%%%%%%%%%%%%%%%%%%%
891
892  % TODO: review why we need to constrain functions to hold between
          endurants and types only (not even "endurant types")
893
894  fof(ax_function, axiom,  (
895    ![X,Y]: (function(X,Y) => (endurant(X) & type_(Y)))
896  )).
897
898  fof(ax_genericFunctionalDependence_a55, axiom, (
899    ![T1,T2,W]: (gfd(T1,T2,W) <=>
900      ![E1]: ((iof(T1,E1,W) & function(T1,E1)) => ?[E2]: (~(E1=E2) &
          iof(T2,E2,W) & function(T2,E2))))
901  )).
902
903  fof(ax_individualFunctionalDependence_a56, axiom, (
```

```
904   ![E1,T1,E2,T2,W]: (ifd(E1,T1,E2,T2,W) <=> (
905     gfd(T1,T2,W) & iof(E1,T1,W) & iof(E2,T2,W) & (function(E1,T1)
         => function(E2,T2))
906   ))
907 )).
908
909 fof(ax_componentOf_a57, axiom, (
910   ![E1,T1,E2,T2,W]: (componentOf(E1,T1,E2,T2,W) <=> (properPartOf(
        E1,E2) & ifd(E1,T1,E2,T2,W)))
911 )).
912
913 % Composition in use
914 % (tested to rule out trivial models)
915
916 % fof(ax_compositionInUse, axiom, (
917 %    componentOf(e11_1,t11_1,e11_2,t11_2,w11) & ~(e11_1=e11_2) & ~(
        e11_1=t11_1) & ~(e11_2=t11_2) & ~(e11_1=t11_2) & ~(e11_2=t11_1)
         & ~(t11_1=t11_2)
918 % )).
```

2.1.11   Constitution

```
920 %%%%%%%%%%%%%%%%%%%%% Constitution %%%%%%%%%%%%%%%%%%%%%
921
922 fof(ax_constitutedByInvolvedNatures_a58, axiom, (
923   ![X,Y,W]: (constitutedBy(X,Y,W) => ((endurant(X) <=> endurant(Y))
         & (perdurant(X) <=> perdurant(Y)) & world(W)))
924 )).
925
926 fof(ax_constitutedByDifferentKinds_a59, axiom, (
927   ![E1,E2,T1,T2,W]: ((constitutedBy(E1,E2,W) & iof(E1,T1,W) & iof(
        E2,T2,W) & kind(T1) & kind(T2)) => (~(T1=T2)))
928 )).
929
930 % Ax |= "th_noSelfConstitution_t27"; conjecture commented for
        convenience
931
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]: (iof(E1,T1,W) => (
939       ?[E2]: (constitutedBy(E1,E2,W) & iof(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     iof(E1,T1,W) & iof(E2,T2,W) & genericConstitutionalDependence(
        T1,T2) & constitutedBy(E1,E2,W)
947   ))
948 )).
949
950 % TODO: add axiom (a62) once the predicate "exists(X,W)" is
        introduced.
```

```
951
952 % Constitution in use
953 % (tested to rule out trivial models)
954
955 % fof(ax_constitutionInUse, axiom, (
956 %   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)
957 % )).
```

2.1.12   Existential Dependence