A TPTP Formalization of the Unified Foundational Ontology

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

July 27, 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

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%TAXONOMY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


fof(ax_taxonomy, axiom, (
![X]: ((type(X)|individual(X))<=>(thing(X))))).

fof(ax_taxonomy, axiom, (
    ~?[X]: (type(X)&individual(X)))).


%%%%%Taxonomy of Individuals:

fof(ax_taxonomy, axiom, (
![X]: ((concreteIndividual(X)|abstractIndividual(X))<=>(individual(
    X))))).
```

```
18
19
20 fof(ax_taxonomy, axiom, (
21 ![X]: ((perdurant(X)|endurant(X))<=>(concreteIndividual(X)))).
22
23
24 fof(ax_taxonomy, axiom, (
25 ![X]: ((quale(X)|set(X))=>(abstractIndividual(X)))).
26
27 fof(ax_taxonomy, axiom, (
28 ![X]: ((object(X)|collective(X)|quantity(X))<=>(substantial(X)))).
29
30
31 fof(ax_taxonomy, axiom, (
32 ![X]: ((substantial(X)|moment(X))<=>(endurant(X)))).
33
34 fof(ax_taxonomy, axiom, (
35 ![X]: ((intrinsicMoment(X)|relator(X))<=>(moment(X)))).
36
37 fof(ax_taxonomy, axiom, (
38 ![X]: ((quality(X)|mode(X))<=>(intrinsicMoment(X)))).
39
40 fof(ax_taxonomy, axiom, (
41 ![X]: ((externallyDependentMode(X)|disposition(X))<=>(mode(X)))).
42
43 fof(ax_taxonomy, axiom, (
44 ![X]: ((quaIndividual(X))<=>(externallyDependentMode(X)))).
45
46 fof(ax_taxonomy, axiom, (
47 ![X]: ((qualityStructure(X))<=>(set(X)))).
48
49 fof(ax_taxonomy, axiom, (
50 ![X]: ((qualityDimension(X)|qualitySpace(X))<=>(qualityStructure(X)
    )))).
51
52
53 fof(ax_taxonomy, axiom, (
54 ![X]: (world(X)=>(qualityStructure(X)))).
55
56
57 %%%%%%Disjoitness axioms:
58
59 fof(ax_taxonomy, axiom, (
60     ~?[X]: (concreteIndividual(X)&abstractIndividual(X)))).
61
62 fof(ax_taxonomy, axiom, (
63     ~?[X]: (perdurant(X)&endurant(X)))).
64
65
66 fof(ax_taxonomy, axiom, (
67     ~?[X]: (substantial(X)&moment(X)))).
68
69 fof(ax_taxonomy, axiom, (
70     ~?[X]: (quale(X)&set(X)))).
71
72
73 fof(ax_taxonomy, axiom, (
```

```
    ~?[X]: (object(X)&collective(X))).


fof(ax_taxonomy, axiom, (
    ~?[X]: (collective(X)&quantity(X))).


fof(ax_taxonomy, axiom, (
    ~?[X]: (object(X)&quantity(X))).

fof(ax_taxonomy, axiom, (
    ~?[X]: (intrinsicMoment(X)&relator(X))).


fof(ax_taxonomy, axiom, (
    ~?[X]: (quality(X)&mode(X))).

fof(ax_taxonomy, axiom, (
    ~?[X]: (externallyDependentMode(X)&disposition(X))).

fof(ax_taxonomy, axiom, (
    ~?[X]: (qualityDimension(X)&qualitySpace(X))).




%%%%%Taxonomy of Types accoording to their instances

fof(ax_taxonomy, axiom, (
![X]: ((perdurantType(X)|endurantType(X))=>(type(X)))).

fof(ax_taxonomy, axiom, (
![X]: ((substantialType(X)|momentType(X))<=>(endurantType(X)))).

fof(ax_taxonomy, axiom, (
![X]: ((objectType(X)|collectiveType(X)|quantityType(X))<=>(
    substantialType(X)))).

fof(ax_taxonomy, axiom, (
![X]: ((intrinsicMomentType(X)|relatorType(X))<=>(momentType(X))))
    .

fof(ax_taxonomy, axiom, (
![X]: ((qualityType(X)|modeType(X))<=>(intrinsicMomentType(X)))).

%Disjointness axioms


fof(ax_taxonomy, axiom, (
    ~?[X]: (perdurantType(X)&endurantType(X))).

fof(ax_taxonomy, axiom, (
    ~?[X]: (substantialType(X)&momentType(X))).

fof(ax_taxonomy, axiom, (
    ~?[X]: (objectType(X)&collectiveType(X))).

```

```
129
130 fof(ax_taxonomy, axiom, (
131     ~?[X]: (objectType(X)&quantityType(X)))).
132
133 fof(ax_taxonomy, axiom, (
134     ~?[X]: (collectiveType(X)&quantityType(X)))).
135
136 fof(ax_taxonomy, axiom, (
137         ~?[X]: (intrinsicMomentType(X)&relatorType(X)))).
138
139 fof(ax_taxonomy, axiom, (
140         ~?[X]: (qualityType(X)&modeType(X)))).
141
142
143 %%%%%Taxonomy accordint to metaproperty (sortality etc).
144
145
146 fof(ax_taxonomy, axiom, (
147 ![X]: ((sortal(X)|nonSortal(X))<=>(endurantType(X))))).
148
149 fof(ax_taxonomy, axiom, (
150 ![X]: ((rigidSortal(X)|antiRigidSortal(X))<=>(sortal(X))))).
151
152 fof(ax_taxonomy, axiom, (
153 ![X]: ((kind(X)|subkind(X))<=>(rigidSortal(X))))).
154
155 fof(ax_taxonomy, axiom, (
156 ![X]: ((phase(X)|role(X))<=>(antiRigidSortal(X))))).
157
158 fof(ax_taxonomy, axiom, (
159 ![X]: ((category(X)|antiRigidNonSortal(X)|mixin(X))<=>(nonSortal(X)
    ))))).
160
161 fof(ax_taxonomy, axiom, (
162 ![X]: ((phaseMixin(X)|roleMixin(X))<=>(antiRigidNonSortal(X))))).
163
164
165 %%%Disjointness axioms:
166
167 fof(ax_taxonomy, axiom, (
168     ~?[X]: (sortal(X)&nonSortal(X)))).
169
170 fof(ax_taxonomy, axiom, (
171     ~?[X]: (rigidSortal(X)&antiRigidSortal(X)))).
172
173 fof(ax_taxonomy, axiom, (
174         ~?[X]: (kind(X)&subKind(X)))).
175
176 fof(ax_taxonomy, axiom, (
177         ~?[X]: (phase(X)&roles(X)))).
178
179 fof(ax_taxonomy, axiom, (
180         ~?[X]: (category(X)&antiRigidNonSortal(X)))).
181
182 fof(ax_taxonomy, axiom, (
183         ~?[X]: (category(X)&mixin(X)))).
184
```

```
fof(ax_taxonomy, axiom, (
        ~?[X]: (antiRigidNonSortal(X)&mixin(X)))).

fof(ax_taxonomy, axiom, (
      ~?[X]: (phaseMixin(X)&roleMixin(X)))).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%Definition of instance of (iof), types, and individuals
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fof(ax_dtypea1, axiom, (
    ![X] : (type(X) <=> (?[W,Y] : (world(W) & (iof(Y,X,W))
    )))))).

%Note: we need to drop that individuals are not wordls, as now
    worlds are abstract.

fof(ax_dindividual_a2, axiom, (
        ![X] : (individual(X) <=> (![W] : (world(W) => ~?[Y] : (iof
    (Y,X,W)))) )
        )).

fof(ax_multilevel_a3, axiom, (
    ![X,Y,W] : (iof(X,Y,W) => ((individual(X)|type(X)) & type(Y) &
    world(W))))).

fof(ax_twolevels_a4, axiom, (
        ~?[X,Y,Z,W]: (type(X) & iof(X,Y,W) & iof(Y,Z,W)))).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
%Specialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5


fof(ax_dspecialization_a5, axiom, (
  ![T1,T2] :  (specializes(T1,T2) <=> (type(T1) & type(T2) &
                    ![W]: (world(W) => ![E]:(iof(E,T1,W) => iof(E,
    T2,W)))))))).

fof(ax_dspecialization_strict_d1, axiom, (
  ![T1,T2] :  (strictlySpecializes(T1,T2) <=> (specializes(T1,T2) &
     ~specializes(T2,T1))))).

  % Whenever two types have a common instance, they must share a
    supertype or a subtype for this instance
  fof(ax_nondisjointSameTaxonomy_a6, axiom, (
      ![T1,T2]: (![X,W]: ((iof(X,T1,W)&iof(X,T2,W)&~specializes(T1,
    T2)&~specializes(T2,T1))=>
           (
               (?[T3]: (specializes(T1,T3)&specializes(T2,T3)&iof(X,
    T3,W)))|
```

```
               (?[T3]: (specializes(T3,T1)&specializes(T3,T2)&iof(X,T3,W)))
          )
          ))
  )).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%Definition of rigidity, anitirigidity, semirigidity, sortality.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Definition of rigid type
fof(ax_drigid_a7, axiom, (
    ![T]: (rigid(T)<=>(endurantType(T) &
                    (![X]: ((?[W]: (world(W) & iof(X,T,W))) => (![W2]: (world(W2)=>iof(X,T,W2))))))
)).

% Definition of antirigid type
fof(ax_dantirigid_a8, axiom, (
    ![T]: (antiRigid(T)<=>(endurantType(T) &
                    (![X]: ((?[W]: (world(W) & iof(X,T,W))) => (?[W2]: (world(W2) & ~iof(X,T,W2)))))))).

% Implicit definition of semirigid type
fof(ax_semirigid_a9, axiom, (
    ![T]: (semiRigid(T)<=>(endurantType(T) &
                    ~antiRigid(T) & ~rigid(T)))
)).


%%%%%%%%%%%%%%%%%%%%%% Definition of sortality

% Every *individual* necessarily instantiates a kind  // imply kinds are rigid!

fof(ax_individualKindMin_a10_revised_to_endurants, axiom, (
    ![X] : (endurant(X) => ?[K]:(kind(K) & ![W]: (world(W)=>iof(X,K,W))))
    )).

% Every thing instantiates at most one kind (whenever it instantiates a kind it does not
% possible instantiate a different one

fof(ax_individualKindMax_a11, axiom, (
    ![X,K,W] : ( ( kind(K) & iof(X,K,W)) =>
                 (~?[Z,W2]: (~(Z=K) & kind(Z) & iof(X,Z,W2))) )
    )).

% Sortals definition, sortals are those types whose instances instantiate the same kind

fof(ax_dsortal_a12, axiom, (
```

```
    ![T] : (sortal(T) <=> (endurantType(T) &
                      (?[K] : (kind(K) & ![X,W]: (world(W)=>(iof(X,T,
    W) => iof(X,K,W) ))))))
    )).

% A non-sortal is a type that is not a sortal

fof(ax_dnonsortal_a13, axiom, (
    ![T] : (nonSortal(T) <=> (endurantType(T) & ~sortal(T)) )
    )).


%%%%%%%%%%%%%%%%%%%%%%% Definitions


fof(ax_rigidSortal, axiom, (
            ![T]: (rigidSortal(T)<=>(rigid(T)&sortal(T)))
        )).

fof(ax_rigidNonSortal, axiom, (
    ![T]: (rigidNonSortal(T)<=>(rigid(T) & ~sortal(T)))
                )).
fof(ax_antiRigidSortal, axiom, (
                      ![T]: (antiRigidSortal(T)<=>(antiRigid(T)&
    sortal(T)))
                                  )).

fof(ax_antiRigidNonSortal, axiom, (
            ![T]: (antiRigidNonSortal(T)<=>(antiRigid(T)&~sortal(T)
    )))).

fof(ax_semiRigidNonSortal, axiom, (
                         ![T]: (semiRigidNonSortal(T)<=>(semiRigid(T
    )&~sortal(T))))).



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Taxonomy of endurant types according to the ontological nature of
    their instances

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Endurant types are all those types whose instances are endurants

fof(ax_dendurantType_a36, axiom, (
      ![T]: (endurantType(T) <=> (type(T) & (![X,W]: (iof(X,T,W) =>
     endurant(X)))))
      )).

% Substantial types are all those types whose instances are
    substantials

fof(ax_dsubstantialType_a36, axiom, (
          ![T]: (substantialType(T)<=> (type(T) & (![X,W]: (iof(X,T
    ,W)=>substantial(X)))))
```

```
      )).

% Moment types are all those types whose instances are moments

fof(ax_dmomentType_a36, axiom, (
      ![T]: (momentType(T) <=> (type(T) & (![X,W]: (iof(X,T,W)=>
    moment(X))))))
      )).

% Relator types are all those types whose instances are relators

  fof(ax_drelatorType_a36, axiom, (
          ![T]: (relatorType(T)<=> (type(T) & (![X,W]: (iof(X,T,W)
    =>relator(X)))))
      )).

% Mode types are all those types whose instances are modes

      fof(ax_dmodeType_a36, axiom, (
          ![T]: (modeType(T)<=> (type(T) & (![X,W]: (iof(X,T,W)=>
    mode(X))))))
      )).

% Quality types are all those types whose instances are qualities

      fof(ax_dqualityType_a36, axiom, (
          ![T]: (qualityType(T) <=> (type(T) & (![X,W]: (iof(X,T,W)
    =>quality(X)))))
      )).


%%% Kinds are specialized according to the ontological nature of
    their instances

      % Substantial kinds are those kinds whose instances are
    substantials
      fof(ax_dsubstantialKind_a37, axiom, (
          ![T]: (substantialKind(T)<=> (substantialType(T) & kind(T
    )))
      )).

      % Relator kinds are those kinds whose instances are relators
      fof(ax_drelatorKind_a37, axiom, (
          ![T]: (relatorKind(T) <=> (relatorType(T) & kind(T)))
      )).

      % Mode kinds are those kinds whose instances are modes
      fof(ax_dmodeKind_a37, axiom, (
          ![T]: (modeKind(T)<=> (modeType(T) & kind(T)))
      )).

      % Quality kinds are those kinds whose instances are relators
      fof(ax_dqualityKind_a37, axiom, (
          ![T]: (qualityKind(T)<=> (qualityType(T) & kind(T)))
      )).

```

```
      % every endurant is instance of one of the specific endurant kinds
      fof(ax_everyEndurantInstantiatesSpecificKind_a38, axiom, (
          ![X]: (endurant(X) => (?[W,K]: ((substantialkind(K)|relatorkind(K)|modekind(K)|qualitykind(K))& iof(X,K,W))))
      )).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Mereology

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



fof(ax_part_arguments, axiom, (![X,Y]: (part(X,Y) => (concreteIndividual(X) & concreteIndividual(Y))))).

fof(ax_part_rifl, axiom, (![X]: (concreteIndividual(X) => part(X,X)))).

fof(ax_part_antisymm, axiom, (![X,Y]: ((part(X,Y) & part(Y,X)) => (Y=X)))).

fof(ax_part_tran, axiom, (![X,Y,Z]: ((part(X,Y) & part(Y,Z)) => part(X,Z)))).

fof(ax_part_overlappin, axiom, (![X,Y]: (overlap(X,Y) <=> ?[Z]:(part(Z,X) & part(Z,Y))))).

fof(ax_part_strong_supp, axiom, (![X,Y]: ((concreteIndividual(Y) & concreteIndividual(X) & ~part(Y,X)) => ?[Z]: (part(Z,Y) & ~overlap(Z,X))))).

fof(ax_part_proper_part, axiom, (![X,Y]: (properPart(Y,X) <=> (part(Y,X) & ~part(X,Y))))).

fof(ax_part_sum, axiom, (![Z,X,Y]: (sum(Z,X,Y) <=> ![W]:((overlap(W,Z) <=> (overlap(W,X) | overlap(W,X))))))).

%Check how much fusion  and existence of sum is needed.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
%Composition
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%Function as relations, "x function as y"
```

```
fof(ax_function, axiom, (![X,Y]: (function(X,Y) => (endurant(X) & type(Y))))).

%Generic functional independence

fof(ax_gfd_a47, axiom, (![X1,Y1,W]: (gfd(X1,Y1,W) <=> (![X]: ((iof(X,X1,W) & function(X,X1))
                                  => (?[Y]: (~(Y=X) & iof(Y,Y1,W) & function(Y,Y1)))))))).

%Individual functional dependence

fof(ax_ifd_a48, axiom, (![X,X1,Y,Y1,W]: (ifd(X,X1,Y,Y1,W) <=> (gfd(X1,Y1,W) & iof(X,X1,W) & iof(Y,Y1,W) &
                                (function(X,X1) => function(Y,Y1)))))).


%Component of

fof(ax_ifd_a49, axiom, (![X,X1,Y,Y1,W]: (componentOf(X,X1,Y,Y1,W) <=> (properPart(X,Y) & ifd(X,X1,Y,Y1,W))))).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%Constitution
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%ConstitutedBy

fof(ax_constituted_by_a58, axiom, (![X,Y,W]: (constitutedBy(X,Y,W) <=>
      ((endurant(X) <=> endurant(Y)) & (perdurant(X) <=> perdurant(Y)) & world(W))))).

fof(ax_constituted_by_a59, axiom, (![X,Y,X1,Y1,W]: ((constitutedBy(X,Y,W) & iof(X,X1,W) & iof(Y,Y1,W) & kind(X1) & kind(Y1)) => ~(X1=Y1)))).


%%%%Generic constutional dependence (GCD).

fof(ax_gcd_a60, axiom, (![X1,Y1]: ((gcd(X1,Y1) <=> (type(X1) & type(Y1) & ![X,W]:(iof(X,X1,W) =>
                        (?[Y]:(iof(Y,Y1,W) & constitutedBy(X,Y,W)))))))).

%%%%Constitution

```

```
fof(ax_constitution_a61, axiom, (![X,Y,X1,Y1,W]: ((constitution(X,X1,Y,Y1,W)
                  <=> (iof(X,X1,W) & iof(Y,Y1,W) & gcd(X1,Y1) & consitutedBy(X,Y,W)))))).


fof(ax_constitution_perdurants_a62, axiom, (![X,Y,W]: ((perdurant(X) & constitutedBy(X,Y,W)) =>
                    (![W1]: (exists(X,W1) => constitutedBy(X,Y,W1)))))).

fof(ax_constitution_a63, axiom, (![X,Y,W]: (constitutedBy(X,Y,W) => ~(constitutedBy(Y,X,W))))).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%Existence, existential dependence, existential independence
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Existence axiom.

fof(ax_existence, axiom, (![X,W]: (ex(X,W) => (thing(X) & world(W))))).

%existential dependence and independence

fof(ax_existential_dependence, axiom, (![X,Y]: (ed(X,Y) <=> ![W]:(ex(X,W) => ex(Y,W))))).

fof(ax_existential_independence, axiom, (![X,Y]: (ind(X,Y) <=> (~ed(X,Y) & ~ed(Y,X))))).




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Inherence
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


  fof(ax_inherence_type, axiom, (![X,Y]: (inheresIn(X,Y) => (moment(X) & (type(X) | concreteIndividual(Y)))))).

      fof(ax_inherence_ed, axiom, (![X,Y]: (inheresIn(X,Y) => ed(X,Y)))).

      fof(ax_inherence_irrifl, axiom, (![X]: ~inheresIn(X,X))).

      fof(ax_inherence_asymm, axiom, (![X,Y]: (inheresIn(X,Y) => ~inheresIn(Y,X)))).
```

```

        fof(ax_inherence_intrans, axiom, (![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(Y,Z)) => ~inheresIn(X,Z)))).

        fof(ax_inherence_unic, axiom, (![X,Y,Z]: ((inheresIn(X,Y) & inheresIn(X,Z)) => Y=Z))).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Moments
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fof(ax_momentOf, axiom, (![M,X]: (momentOf(M,X) <=> (inheresIn(M,X) | (?[Y]:(ineheresIn(M,Y) & momentOf(Y,X))))))).

fof(ax_ulitmate_bearer, axiom, (![B,M]: (ultimateBearerOf(B,M) <=> (~moment(B) & momentOf(B,M))))).

fof(ax_ulitmate_bearer_existence, axiom, (![M]: (moment(M) => (?[B]: (ulimateBererOf(B,M)))))).

fof(ax_ulitmate_bearer_unicity, axiom, (![M,B,B1]: ((ulimateBererOf(B,M) & ultimateBearerOf(B1,M)) => (B=B1)))).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Relators
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Externally dependent (we avoid introducing here the function "bearer of a moment", it is however unique)


fof(ax_externally_dependent, axiom, (![X,Y]: (externallyDependent(X,Y) <=>
                                        (ed(X,Y) & (![Z]: (inheresIn(X,Z) => ind(Y,Z))))))).

%Externally dependent modes

fof(ax_externally_dependent_mode, axiom, (![X]: (externallyDependentMode(X) <=> (mode(X) & (?[Y]: (externallyDependent(X,Y))))))).


%Founded by

fof(ax_founded_by, axiom, (![X,Y]: (foundedBy(X,Y) => ((externallyDependentMode(X) | relator(X)) & perdurant(Y))))).

fof(ax_foundation_existence, axiom, (![X]: (externallyDependentMode(X) => (?[Y]: (foundedBy(X,Y)))))).
```

```
548
549 fof(ax_foundation_unicity, axiom, (![X,Y,Z]: ((foundedBy(X,Y) &
        foundedBy(X,Z)) => (Y=Z)))).
550
551 %Qua individual of
552
553 fof(ax_qua_individual_of, axiom, (![X,Y]: (quaIndividualOf(X,Y) <=>
554                                       (![Z]:(overlap(Z,X) <=>
555                                             (
        externallyDependentMode(Z) & inheresIn(Z,Y) &
556                                                (![P]: (foundedBy(X,P)
        => foundedBy(Z,P)))))))).
557
558 fof(ax_qua_individual_of_unicity, axiom, (![X,Y,Y1]: ((
        quaIndividualOf(X,Y) & quaIndividualOf(X,Y1)) => (Y=Y1)))).
559
560 %Qua individual
561
562 fof(ax_qua_individual_def, axiom, (![X]: ((quaIndividual(X) <=> (?[
        Y]: (quaIndividualOf(X,Y))))))).
563
564
565
566
567
568
569
570 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
571
572 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
573
574 %Instances
575
576 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
577 %Taxonomy of types 1.
578
579 %fof(ax_instance, axiom, perdurantType(pt1)).
580 %fof(ax_instance, axiom, axiom,objectType(ot1)).
581 %fof(ax_instance, axiom, collectiveType(ct1)).
582 %fof(ax_instance, axiom, quantityType(qnt1)).
583 %fof(ax_instance, axiom, relatorType(rt1)).
584 %fof(ax_instance, axiom, qualityType(qlt1)).
585 %fof(ax_instance, axiom, modeType(mot1)).
586
587 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
588 %Taxonomy of types 2
589
590 %fof(ax_instance, axiom,kind(k1)).
591 %fof(ax_instance, axiom, subKind(sk1)).
592 %fof(ax_instance, axiom, phase(phase1)).
593 %fof(ax_instance, axiom, role(role1)).
594 %fof(ax_instance, axiom, category(cat1)).
595 %fof(ax_instance, axiom, phaseMixin(pm1)).
596 %fof(ax_instance, axiom, roleMixin(rmi1)).
597 %fof(ax_instance, axiom, mixin(mix1)).
598
599 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Taxnomy of individuals


%fof(ax_instance, axiom, perdurant(p1)).
%fof(ax_instance, axiom, object(o1)).
%fof(ax_instance, axiom, collective(c1)).
%fof(ax_instance, axiom, quantity(quan1)).
%fof(ax_instance, axiom, relator(r1)).
%fof(ax_instance, axiom, quality(qual1)).
%fof(ax_instance, axiom, quaIndividual(qi1)).
%fof(ax_instance, axiom, disposition(d1)).
%fof(ax_instance, axiom, qualityDimension(qd1)).
%fof(ax_instance, axiom, qualitySpace(qs1)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```