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

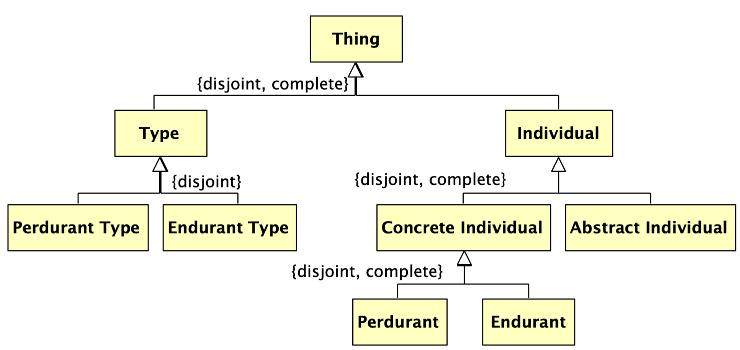

Figure 1: Partial Taxonomy of UFO – Thing.

```
% Thing

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

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

% Individual

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

fof(ax_individual_partition, axiom, (
  ~?[X]: (concreteIndividual(X) & abstractIndividual(X))
)).

% Concrete Individual

fof(ax_concreteIndividual_taxonomy, axiom, (
  ![X]: ((endurant(X) | perdurant(X)) <=> (concreteIndividual(X)))
)).

fof(ax_concreteIndividual_partition, axiom, (
  ~?[X]: (endurant(X) & perdurant(X))
)).

% Type

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

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

% Thing partial taxonomy instances
% (tested rule out trivial models)

% fof(ax_thing_instances, axiom, (
%   type(type1) & individual(individual1) & concreteIndividual(
    concreteIndividual1) & abstractIndividual(abstractIndividual1)
    & endurant(endurant1) & perdurant(perdurant1) & endurantType(
    endurantType1) & perdurantType(perdurantType1)
% )).
```

```
% Abstract Individual

fof(ax_abstractIndividual_taxonomy_quale, axiom, (
  ![X]: (quale(X) => (abstractIndividual(X)))
)).

fof(ax_abstractIndividual_taxonomy_set, axiom, (
```

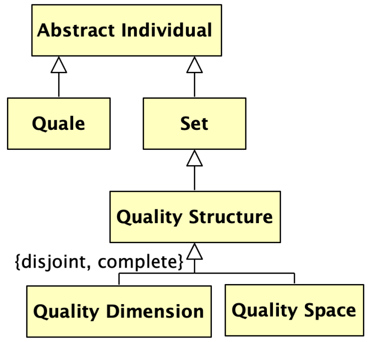

Figure 2: Partial Taxonomy of UFO – Abstract Individual.

```
  ![X]: (set(X) => (abstractIndividual(X)))
)).

% Set

fof(ax_set_taxonomy_qualityStructure, axiom, (
  ![X]: (qualityStructure(X) => (set(X)))
)).

% Quality Structure

fof(ax_qualityStructure_taxonomy, axiom, (
  ![X]: ((qualityDimension(X) | qualitySpace(X)) <=> (qualityStructure(X)))
)).

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

% TODO: review the definition of "world" as a subtype of "qualityStructure"

fof(ax_qualityStructure_taxonomy_world, axiom, (
  ![X]: (world(X) => (qualityStructure(X)))
)).

% Abstract Individual partial taxonomy instances
% (tested rule out trivial models)

% fof(ax_abstractIndividual_instances, axiom, (
%   set(set1) & quale(quale1) & qualityStructure(qualityStructure1) & qualityDimension(qualityDimension1) & qualitySpace(qualitySpace1) & world(world1)
% )).
```

```
% Endurant

fof(ax_endurant_taxonomy, axiom, (
```

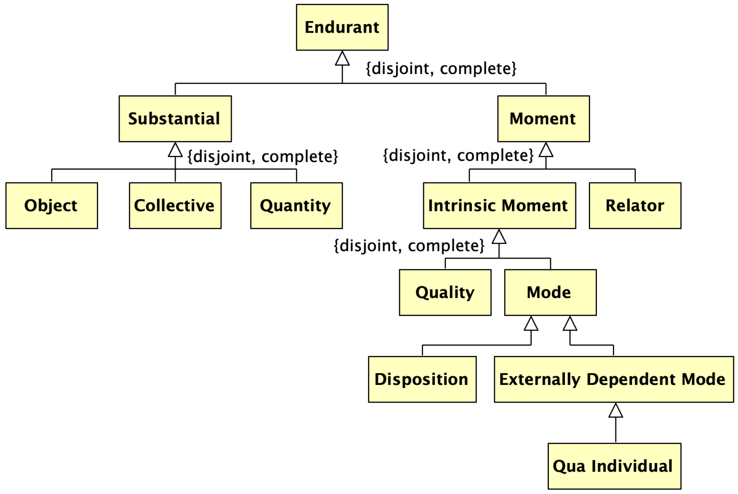

Figure 3: Partial Taxonomy of UFO – Endurant.

```
  ![X]: ((substantial(X) | moment(X)) <=> (endurant(X)))
)).

fof(ax_endurant_partition, axiom, (
  ~?[X]: (substantial(X) & moment(X))
)).

% Substantial

fof(ax_substantial_taxonomy, axiom, (
  ![X]: ((object(X) | collective(X) | quantity(X)) <=> (substantial
    (X)))
)).

fof(ax_substantial_partition, axiom, (
  ~?[X]: ((object(X) & collective(X)) | (object(X) & quantity(X)) |
     (collective(X) & quantity(X)))
)).

% Moment

fof(ax_moment_taxonomy, axiom, (
  ![X]: ((intrinsicMoment(X) | relator(X)) <=> (moment(X)))
)).

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

% Intrinsic Moment

fof(ax_intrinsicMoment_taxonomy, axiom, (
  ![X]: ((quality(X) | mode(X)) <=> (intrinsicMoment(X)))
```

```
)).

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

% Mode

fof(ax_mode_taxonomy_externallyDependentMode, axiom, (
  ![X]: (externallyDependentMode(X) => (mode(X)))
)).

% Externally Dependent Mode

fof(ax_externallyDependentMode_taxonomy_quaIndividual, axiom, (
  ![X]: (quaIndividual(X) => (externallyDependentMode(X)))
)).

% Endurant partial taxonomy instances
% (tested rule out trivial models)

% fof(ax_endurant_instances, axiom, (
%   substantial(substantial1) & moment(moment1) & object(object1) & collective(collective1) & quantity(quantity1) & intrinsicMoment(intrinsicMoment1) & relator(relator1) & quality(quality1) & mode(mode1) & disposition(disposition1) & externallyDependentMode(externallyDependentMode1) & quaIndividual(quaIndividual1)
% )).
```

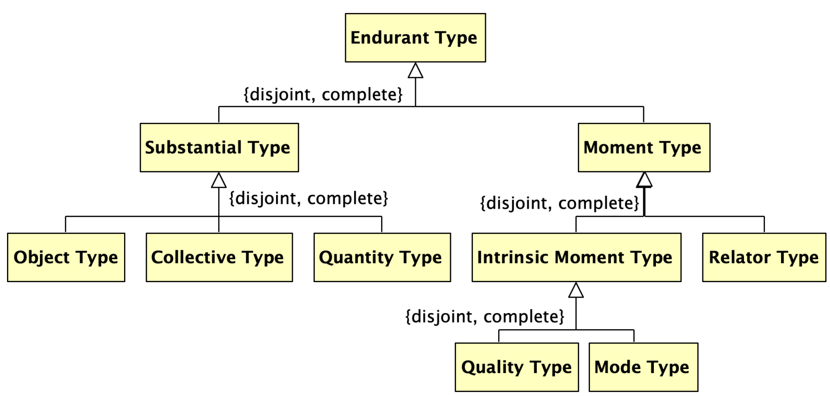

Figure 4: Partial Taxonomy of UFO – Endurant Types (by ontological nature).

```
% Endurant Type (by ontological nature)

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

```

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

% Substantial Type

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

fof(ax_substantialType_partition, axiom, (
  ~?[X]: ((objectType(X) & collectiveType(X)) | (objectType(X) &
    quantityType(X)) | (collectiveType(X) & quantityType(X)))
)).

% Moment Type

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

fof(ax_momentType_partition, axiom, (
  ~?[X]: (intrinsicMomentType(X) & relatorType(X))
)).

% Intrinsic Moment Type

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

fof(ax_intrinsicMomentType_partition, axiom, (
  ~?[X]: (qualityType(X) & modeType(X))
)).

% Endurant Type (by ontological nature) partial taxonomy instances
% (tested rule out trivial models)

% fof(ax_endurantType_instances_natures, axiom, (
%   substantialType(substantialType1) & momentType(momentType1) &
    objectType(objectType1) & collectiveType(collectiveType1) &
    quantityType(quantityType1) & intrinsicMomentType(
    intrinsicMomentType1) & relatorType(relatorType1) & qualityType
    (qualityType1) & modeType(modeType1) &
    externallyDependentModeType(externallyDependentModeType1) &
    quaIndividualType(quaIndividualType1)
% )).
```

```
% Endurant Type (by modal properties of types)

fof(ax_endurantType_taxonomy_properties, axiom, (
  ![X]: ((sortal(X) | nonSortal(X)) <=> (endurantType(X)))
)).

fof(ax_endurantType_partition_properties, axiom, (
```

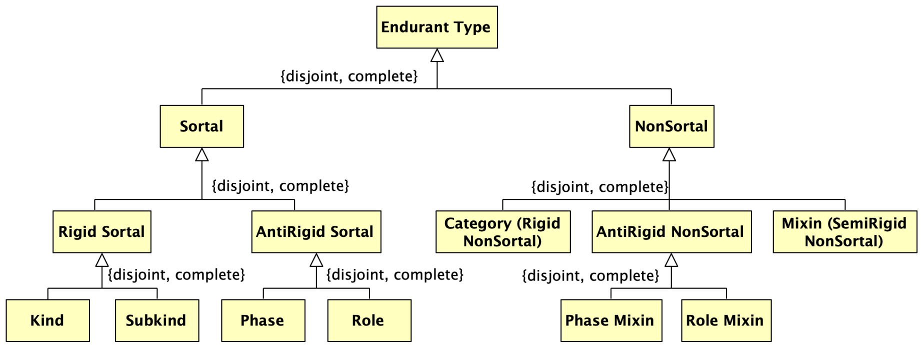

Figure 5: Partial Taxonomy of UFO – Endurant Types (by modal properties of types).

```
  ~?[X]: (sortal(X) & nonSortal(X))
)).

% Sortal

fof(ax_sortal_taxonomy, axiom, (
  ![X]: ((rigidSortal(X) | antiRigidSortal(X)) <=> (sortal(X)))
)).

fof(ax_sortal_partition, axiom, (
  ~?[X]: (rigidSortal(X) & antiRigidSortal(X))
)).

% Rigid Sortal

fof(ax_rigidSortal_taxonomy, axiom, (
  ![X]: ((kind(X) | subkind(X)) <=> (rigidSortal(X)))
)).

fof(ax_rigidSortal_partition, axiom, (
  ~?[X]: (kind(X) & subkind(X))
)).

% Anti-Rigid Sortal

fof(ax_antiRigidSortal_taxonomy, axiom, (
  ![X]: ((phase(X) | role(X)) <=> (antiRigidSortal(X)))
)).

fof(ax_antiRigidSortal_partition, axiom, (
  ~?[X]: (phase(X) & role(X))
)).

% Non-Sortal

fof(ax_nonSortal_taxonomy, axiom, (
  ![X]: ((rigidNonSortal(X) | semiRigidNonSortal(X) |
    antiRigidNonSortal(X)) <=> (nonSortal(X)))
```

```
)).

fof(ax_nonSortal_partition, axiom, (
  ~?[X]: ((rigidNonSortal(X) & semiRigidNonSortal(X)) | (
    rigidNonSortal(X) & antiRigidNonSortal(X)) | (
    semiRigidNonSortal(X) & antiRigidNonSortal(X)))
)).

% Category

fof(ax_rigidNonSortal_taxonomy, axiom, (
  ![X]: (rigidNonSortal(X) <=> (category(X)))
)).

% Mixin

fof(ax_semiRigidNonSortal_taxonomy, axiom, (
  ![X]: (semiRigidNonSortal(X) <=> (mixin(X)))
)).

% Anti-Rigid Non-Sortal

fof(ax_antiRigidNonSortal_taxonomy, axiom, (
  ![X]: ((phaseMixin(X) | roleMixin(X)) <=> (antiRigidNonSortal(X))
    )
)).

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

% Endurant Type (by modal properties of types) partial taxonomy
    instances
% (tested rule out trivial models)

% fof(ax_endurantType_instances_properties, axiom, (
%   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)
% )).
```

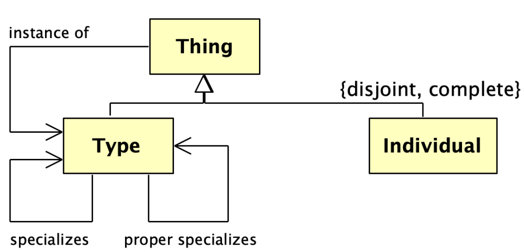

Figure 6: Types, individuals, instantiation, and specialization.

```
%%%%%%%%%%% Instance of, Types, and Individuals %%%%%%%%%%

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

fof(ax_dType_a1, axiom, (
  ![X]: (type(X) <=> (?[Y,W]: iof(Y,X,W)))
)).

fof(ax_dIndividual_a2, axiom, (
  ![X]: (individual(X) <=> (~?[Y,W]: iof(Y,X,W)))
)).

% TODO: confirm whether we are including second-order types in this formalization

fof(ax_multiLevel_a3, axiom, (
  ![X,Y,W]: (iof(X,Y,W) => (type(X) | individual(X)))
)).

fof(ax_twoLevelConstrained_a4, axiom, (
  ~?[X,Y,Z,W]: (type(X) & iof(X,Y,W) & iof(Y,Z,W))
)).

% fof(ax_iofInUse, axiom, (
%   type(t2) & individual(i2) & world(w2) & iof(i2,t2,w2)
% )).

% Ax |= "th_everythingIsAThing_t1"; conjecture commented for convenience

% fof(th_everythingIsAThing_t1, conjecture, (
%   ![X]: (type(X) | individual(X))
% )).

% Ax |= "th_thingPartition_t2"; conjecture commented for convenience

% fof(th_thingPartition_t2, conjecture, (
%   ~?[X]: (type(X) & individual(X))
% )).

%%%%%%%% Specialization and Proper Specialization %%%%%%%%

fof(ax_dSpecializes, axiom, (
  ![X,Y]: (specializes(X,Y) => (type(X) & type(Y)))
)).

fof(ax_specialization_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_properSpecializes_d1, axiom, (
```

```
  ![X,Y]: (properSpecializes(X,Y) <=> (specializes(X,Y) & ~specializes(Y,X)))
)).

% fof(ax_specializesInUse, axiom, (
%   type(t3_1) & type(t3_2) & specializes(t3_1,t3_2) & properSpecializes(t3_1,t3_2) & specializes(t3_1,t3_1)
% )).

% Ax |= "th_cyclicSpecializations_t3"; conjecture commented for convenience

% fof(th_cyclicSpecializations_t3, conjecture, (
%   ![X,Y]: (specializes(X,Y) => (specializes(X,X) & specializes(Y,Y)))
% )).

% Ax |= "th_transitiveSpecializations_t4"; conjecture commented for convenience

% fof(th_transitiveSpecializations_t4, conjecture, (
%   ![X,Y,Z]: ((specializes(X,Y) & specializes(Y,Z)) => (specializes(X,Z)))
% )).

fof(ax_sharedSpecializations_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)))
  )))
)).
```

```
%%%%%%%%%%%%%%%% Sortality and Rigidity %%%%%%%%%%%%%%%%

% 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.
% TODO: verify whether it is a problem not to introduce predicates "world(W1) &" and "world(W2) &" before each instantiation

fof(ax_dRigid_a18, axiom, (
  ![T]: (rigid(T) <=> (endurantType(T) & (
    ![X]: ((![W1]: (world(W1) & iof(X,T,W1))) => (![W2]: (world(W2) => iof(X,T,W2)))
  ))))
)).

fof(ax_dAntiRigid_a19, axiom, (
  ![T]: (antiRigid(T) <=> (endurantType(T) & (
    ![X]: ((![W1]: (world(W1) & iof(X,T,W1))) => (?[W2]: (world(W2) & ~iof(X,T,W2)))
  ))))
)).

fof(ax_dSemiRigid_a20, axiom, (
```

```
  ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid(T)))
)).

% Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented for convenience

% fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
%   ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid(T)))
% )).

% Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented for convenience

% fof(th_pairwiseDisjointRigidities_t6, conjecture, (
%   ~![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)) | (rigid(T) & antiRigid(T)))
% )).

% Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture commented for convenience

% fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
%   ~![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
% )).

% Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8"; conjecture commented for convenience

% fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture, (
%   ~![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
% )).

fof(ax_endurantsKind_a21, axiom, (
  ![E]: (endurant(E) => (
    ?[U]: (kind(U) & (![W]: (world(W) & iof(E,U,W))))
  ))
)).

fof(ax_uniqueKind_a22, axiom, (
  ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
    ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
  ))
)).

% 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

% fof(ax_dSortal_a23, axiom, (
%   ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,W]: (iof(E,S,W) => iof(E,U,W)))))))
% )).

```

```
fof(ax_dSortal_a23, axiom, (
  ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) & specializes(S,U)))))
)).

% If we have the taxonomy's axiomatization, then a24 becomes a theorem
% Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24"; conjecture commented for convenience

% fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
%   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
% )).

% Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience

% fof(th_kindsAreRigid_t9, conjecture, (
%   ![U]: ((kind(U)) => (rigid(U)))
% )).

% Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented for convenience

% fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
%   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
%     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2,W2)))
%   )
% )).

% Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented for convenience

% fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
%   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
%     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
%   )
% )).

% Ax |= "th_kindsAreSortal_t12"; conjecture commented for convenience

% fof(th_kindsAreSortal_t12, conjecture, (
%   ![K]: ((kind(K)) => (sortal(K)))
% )).

% Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for convenience

% fof(th_sortalSpecializeKinds_t13, conjecture, (
%   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
% )).

% Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented for convenience

% fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
```

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

% If we have the taxonomy's axiomatization, then a25 becomes a theorem
% Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented for convenience

% fof(th_kindAndSubkindAreDisjoint_a25, conjecture, (
%   ~?[T]: (kind(T) & subkind(T))
% )).

fof(ax_kindAndSubkindAreRigidSortals_a26, axiom, (
  ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))
)).

% If we have the taxonomy's axiomatization, then a27 becomes a theorem
% Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for convenience

% fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
%   ~?[T]: (phase(T) & role(T))
% )).

fof(ax_phaseAndRoleAreAntiRigidSortals_a28, axiom, (
  ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
)).

% Skipping (a29) because we leave the concept of semi-rigid sortals out of this ontology.

fof(ax_categoriesAreRigidNonSortals_a30, axiom, (
  ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))
)).

fof(ax_mixinsAreSemiRigidNonSortals_a30, axiom, (
  ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
)).

% If we have the taxonomy's axiomatization, then a31 becomes a theorem
% Ax |= "th_phaseMixinsAndRoleMixinsAreDisjoint_a31"; conjecture commented for convenience

% fof(th_phaseMixinAndRoleMixinAreDisjoint_a31, conjecture, (
%   ~?[T]: (phaseMixin(T) & roleMixin(T))
% )).

fof(ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a28, axiom, (
  ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) & nonSortal(T)))
)).

% Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture commented for convenience
```

```
% fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
%   ~?[T]: (endurantType(T) & (
%     (
%       (kind(T) & subkind(T))
%       | (kind(T) & phase(T))
%       | (kind(T) & role(T))
%       | (kind(T) & category(T))
%       | (kind(T) & mixin(T))
%       | (kind(T) & phaseMixin(T))
%       | (kind(T) & roleMixin(T))
%     ) | (
%       (subkind(T) & phase(T))
%       | (subkind(T) & role(T))
%       | (subkind(T) & category(T))
%       | (subkind(T) & mixin(T))
%       | (subkind(T) & phaseMixin(T))
%       | (subkind(T) & roleMixin(T))
%     ) | (
%       (phase(T) & role(T))
%       | (phase(T) & category(T))
%       | (phase(T) & mixin(T))
%       | (phase(T) & phaseMixin(T))
%       | (phase(T) & roleMixin(T))
%     ) | (
%       (role(T) & category(T))
%       | (role(T) & mixin(T))
%       | (role(T) & phaseMixin(T))
%       | (role(T) & roleMixin(T))
%     ) | (
%       (category(T) & mixin(T))
%       | (category(T) & phaseMixin(T))
%       | (category(T) & roleMixin(T))
%     ) | (
%       (mixin(T) & phaseMixin(T))
%       | (mixin(T) & roleMixin(T))
%     ) | (
%       (phaseMixin(T) & roleMixin(T))
%     )
%   ))
% )).

% Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
    "; conjecture commented for convenience

% fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19,
    conjecture, (
%   ![T]: (endurantType(T) => (
%     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
    mixin(T) | phaseMixin(T) | roleMixin(T)
%   ))
% )).
```

```
%%%%%%%%%%%%%%%% Sortality and Rigidity %%%%%%%%%%%%%%%%

% 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.
```

```
% TODO: verify whether it is a problem not to introduce predicates "world(W1) &" and "world(W2) &" before each instantiation

fof(ax_dRigid_a18, axiom, (
  ![T]: (rigid(T) <=> (endurantType(T) & (
    ![X]: ((![W1]: (world(W1) & iof(X,T,W1))) => (![W2]: (world(W2) => iof(X,T,W2)))
  ))))
)).

fof(ax_dAntiRigid_a19, axiom, (
  ![T]: (antiRigid(T) <=> (endurantType(T) & (
    ![X]: ((![W1]: (world(W1) & iof(X,T,W1))) => (?[W2]: (world(W2) & ~iof(X,T,W2)))
  ))))
)).

fof(ax_dSemiRigid_a20, axiom, (
  ![T]: (semiRigid(T) <=> (endurantType(T) & ~rigid(T) & ~antiRigid(T)))
)).

% Ax |= "th_thEndurantTypeHaveRigidity_t5"; conjecture commented for convenience

% fof(th_thEndurantTypeHaveRigidity_t5, conjecture, (
%   ![T]: (endurantType(T) <=> (rigid(T) | semiRigid(T) | antiRigid(T)))
% )).

% Ax |= "th_pairwiseDisjointRigidities_t6"; conjecture commented for convenience

% fof(th_pairwiseDisjointRigidities_t6, conjecture, (
%   ~![T]: ((rigid(T) & semiRigid(T)) | (semiRigid(T) & antiRigid(T)) | (rigid(T) & antiRigid(T)))
% )).

% Ax |= "th_rigidAntiRigidSpecializationConstraint_t7"; conjecture commented for convenience

% fof(th_rigidAntiRigidSpecializationConstraint_t7, conjecture, (
%   ~![T1,T2]: (rigid(T1) & antiRigid(T2) & specializes(T1,T2))
% )).

% Ax |= "th_semiRigidAntiRigidSpecializationConstraint_t8"; conjecture commented for convenience

% fof(th_semiRigidAntiRigidSpecializationConstraint_t8, conjecture, (
%   ~![T1,T2]: (semiRigid(T1) & antiRigid(T2) & specializes(T1,T2))
% )).

fof(ax_endurantsKind_a21, axiom, (
  ![E]: (endurant(E) => (
    ?[U]: (kind(U) & (![W]: (world(W) & iof(E,U,W))))
  ))
```

```
)).

fof(ax_uniqueKind_a22, axiom, (
  ![E,U,W]: ((world(W) & kind(U) & iof(E,U,W)) => (
    ~?[U2,W2]: (kind(U2) & iof(E,U2,W2) & ~(U = U2))
  ))
)).

% 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

% fof(ax_dSortal_a23, axiom, (
%   ![S]: (sortal(S) <=> (endurantType(S) & (?[U]: (kind(U) & (![E,W]: (iof(E,S,W) => iof(E,U,W)))))))
% )).

fof(ax_dSortal_a23, axiom, (
  ![S]: ((sortal(S)) <=> (endurantType(S) & (?[U]: (kind(U) & specializes(S,U)))))
)).

% If we have the taxonomy's axiomatization, then a24 becomes a theorem
% Ax |= "th_nonSortalsAreEndurantsThatAreNotSortals_a24"; conjecture commented for convenience

% fof(th_nonSortalsAreEndurantsThatAreNotSortals_a24, conjecture, (
%   ![NS]: ((nonSortal(NS)) <=> (endurantType(NS) & ~sortal(NS)))
% )).

% Ax |= "th_kindsAreRigid_t9"; conjecture commented for convenience

% fof(th_kindsAreRigid_t9, conjecture, (
%   ![U]: ((kind(U)) => (rigid(U)))
% )).

% Ax |= "th_kindsHaveDisjointExtensions_t10"; conjecture commented for convenience

% fof(th_kindsHaveDisjointExtensions_t10, conjecture, (
%   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
%     ~?[X,W1,W2]: (world(W1) & world(W2) & iof(X,K1,W1) & iof(X,K2,W2)))
%   )
% )).

% Ax |= "th_kindsHaveDisjointTaxonomies_t11"; conjecture commented for convenience

% fof(th_kindsHaveDisjointTaxonomies_t11, conjecture, (
%   ![K1,K2]: ((kind(K1) & kind(K2) & ~(K1=K2)) => (
%     ~?[T]: (specializes(T,K1) & specializes(T,K2)))
%   )
% )).

```

```
% Ax |= "th_kindsAreSortal_t12"; conjecture commented for convenience

% fof(th_kindsAreSortal_t12, conjecture, (
%   ![K]: ((kind(K)) => (sortal(K)))
% )).

% Ax |= "th_sortalSpecializeKinds_t13"; conjecture commented for convenience

% fof(th_sortalSpecializeKinds_t13, conjecture, (
%   ![S]: ((sortal(S)) => (?[K]: (kind(K) & specializes(S,K))))
% )).

% Ax |= "th_sortalsSpecializeAUniqueKind_t14"; conjecture commented for convenience

% fof(th_sortalsSpecializeAUniqueKind_t14, conjecture, (
%   ![S]: ((sortal(S)) => (~?[U,U2]: (kind(U) & kind(U2) & specializes(S,U) & specializes(S,U2) & ~(U=U2))))
% )).

% If we have the taxonomy's axiomatization, then a25 becomes a theorem
% Ax |= "th_kindAndSubkindAreDisjoint_a25"; conjecture commented for convenience

% fof(th_kindAndSubkindAreDisjoint_a25, conjecture, (
%   ~?[T]: (kind(T) & subkind(T))
% )).

fof(ax_kindAndSubkindAreRigidSortals_a26, axiom, (
  ![T]: ((kind(T) | subkind(T)) <=> (rigid(T) & sortal(T)))
)).

% If we have the taxonomy's axiomatization, then a27 becomes a theorem
% Ax |= "th_phaseAndRoleAreDisjoint_a27"; conjecture commented for convenience

% fof(th_phaseAndRoleAreDisjoint_a27, conjecture, (
%   ~?[T]: (phase(T) & role(T))
% )).

fof(ax_phaseAndRoleAreAntiRigidSortals_a28, axiom, (
  ![T]: ((phase(T) | role(T)) <=> (antiRigid(T) & sortal(T)))
)).

% Skipping (a29) because we leave the concept of semi-rigid sortals out of this ontology.

fof(ax_categoriesAreRigidNonSortals_a30, axiom, (
  ![T]: ((category(T)) <=> (rigid(T) & nonSortal(T)))
)).

fof(ax_mixinsAreSemiRigidNonSortals_a30, axiom, (
  ![T]: ((mixin(T)) <=> (semiRigid(T) & nonSortal(T)))
```

```
)).

% If we have the taxonomy's axiomatization, then a31 becomes a theorem
% Ax |= "th_phaseMixinsAndRoleMixinsAreDisjoint_a31"; conjecture commented for convenience

% fof(th_phaseMixinAndRoleMixinAreDisjoint_a31, conjecture, (
%   ~?[T]: (phaseMixin(T) & roleMixin(T))
% )).

fof(ax_phaseMixinAndRoleMixinAreAntiRigidSortals_a28, axiom, (
  ![T]: ((phaseMixin(T) | roleMixin(T)) <=> (antiRigid(T) & nonSortal(T)))
)).

% Ax |= "th_leafCategoriesArePairwiseDisjoint_t18"; conjecture commented for convenience

% fof(th_leafCategoriesArePairwiseDisjoint_t18, conjecture, (
%   ~?[T]: (endurantType(T) & (
%     (
%       (kind(T) & subkind(T))
%       | (kind(T) & phase(T))
%       | (kind(T) & role(T))
%       | (kind(T) & category(T))
%       | (kind(T) & mixin(T))
%       | (kind(T) & phaseMixin(T))
%       | (kind(T) & roleMixin(T))
%     ) | (
%       (subkind(T) & phase(T))
%       | (subkind(T) & role(T))
%       | (subkind(T) & category(T))
%       | (subkind(T) & mixin(T))
%       | (subkind(T) & phaseMixin(T))
%       | (subkind(T) & roleMixin(T))
%     ) | (
%       (phase(T) & role(T))
%       | (phase(T) & category(T))
%       | (phase(T) & mixin(T))
%       | (phase(T) & phaseMixin(T))
%       | (phase(T) & roleMixin(T))
%     ) | (
%       (role(T) & category(T))
%       | (role(T) & mixin(T))
%       | (role(T) & phaseMixin(T))
%       | (role(T) & roleMixin(T))
%     ) | (
%       (category(T) & mixin(T))
%       | (category(T) & phaseMixin(T))
%       | (category(T) & roleMixin(T))
%     ) | (
%       (mixin(T) & phaseMixin(T))
%       | (mixin(T) & roleMixin(T))
%     ) | (
%       (phaseMixin(T) & roleMixin(T))
%     )
```

```
%   ))
% )).

% Ax |= "th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19
    "; conjecture commented for convenience

% fof(th_leafCategoriesCompletelyCategorizeAllEndurantTypes_t19,
    conjecture, (
%   ![T]: (endurantType(T) => (
%     kind(T) | subkind(T) | phase(T) | role(T) | category(T) |
    mixin(T) | phaseMixin(T) | roleMixin(T)
%   ))
% )).
```