

A First-Order Logic Formalization of the Unified Foundational Ontology

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

December 3, 2021

Abstract

This document presents a formalization of the Unified Foundational Ontology (UFO) in first-order logic. This formalization is documented by means of three complementary representations: (i) a representation in standard Common Logic using the CLIF syntax; (ii) a representation in natural language; and, when applicable, (iii) a UML-based diagrammatic representation. The presented formalization is supported by consistency and satisfiability checks performed through automated proofing tools.

1 Introduction

This document presents a formalization of the Unified Foundational Ontology (UFO) in first-order logic. This formalization is documented by means of three complementary representations: (i) a representation in standard Common Logic using the CLIF syntax; (ii) a representation in natural language; and, when applicable, (iii) a UML-based diagrammatic representation. The presented formalization is supported by consistency and satisfiability checks performed through automated proofing tools.

The remainder of this document is organized as a single formalization section (Section 2), which contains subsections for each submodule of the ontology.

2 Formalization

This section contains the formalization of the Unified Foundational Ontology (UFO) in first-order logics. This formalization is organized in several subsections where each presents the formalization of a portion of the whole ontology. The formalization is presented through different equivalent representations, designed to support the understanding of its contents: (i) a representation in standard Common Logic using the CLIF syntax; (ii) a representation in natural language; and, when applicable, (iii) a UML-based diagrammatic representation.

The UML-based diagrammatic representation serves as a visual representation certain predicates and axioms, being each element in Figure 1 being translated as follows:

- Rectangle shape (Figure 1a): visual representation of unary predicates associated to types in the ontology; the associated predicate is shown in lower camel case with no spaces.

classA(x)

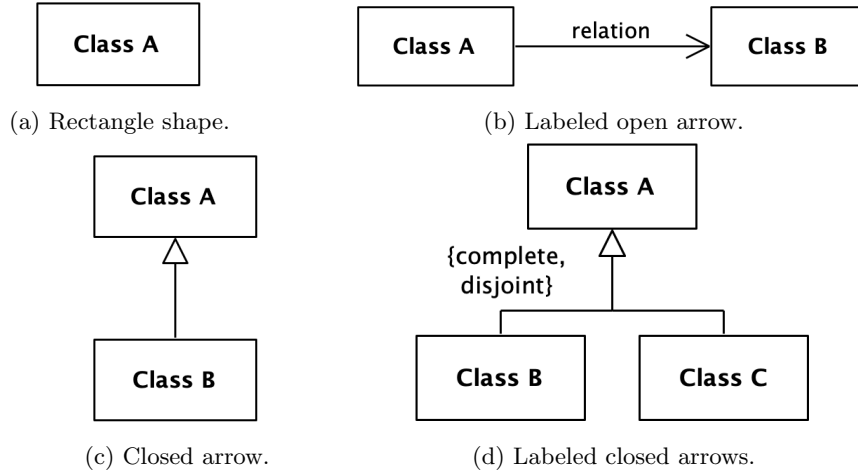


Figure 1: UML-based representation of first-order logic axioms.

- Open arrow (Figure 1b): visual representation of binary predicates; the predicate associated to the arrows' label is shown in lower camel case with no spaces; the predicate can only be true for any x and y if it is also true predicates associated to the types of each end (keeping the order of the arrow in the binary predicate's positions); this representation may also be associated to ternary predicates iff its third position represents a time-index.

$$\forall x, y (relation(x, y) \rightarrow (classA(x) \wedge classB(y)))$$

$$\forall x, y, w (relation(x, y, w) \rightarrow (classA(x) \wedge classB(y) \wedge world(w)))$$

- Closed arrow (Figure 1c): visual representation of specializations between ontology's types, where the type in the tail of the arrow is a subtype of the type in the head of the arrow.

$$\forall x (classB(x) \rightarrow classA(x))$$

- Labeled closed arrow (Figure 1d): visual representation of disjoint and/or complete constraints over sets specializations between ontology's types.

$$\forall x (classB(x) \rightarrow classA(x))$$

$$\forall x (classC(x) \rightarrow classA(x))$$

$$\forall x (classA(x) \rightarrow (classB(x) \vee classC(x)))$$

$$\neg \exists x (classB(x) \wedge classC(x))$$

{complete}
{disjoint}

2.1 Partial Taxonomy of UFO: Thing

This subsection presents most general types of UFO's taxonomy specializing the type **Thing** (Figure 2).

- a1** For every x , x is a **Thing** iff x is either a **Type** or an **Individual**.

$$\forall x (\text{type_}(x) \vee \text{individual}(x) \leftrightarrow \text{thing}(x))$$

```
1 (cl-text ax_thing_taxonomy
2 (forall (x)
3 (iff (or (type_ x) (individual x))
```

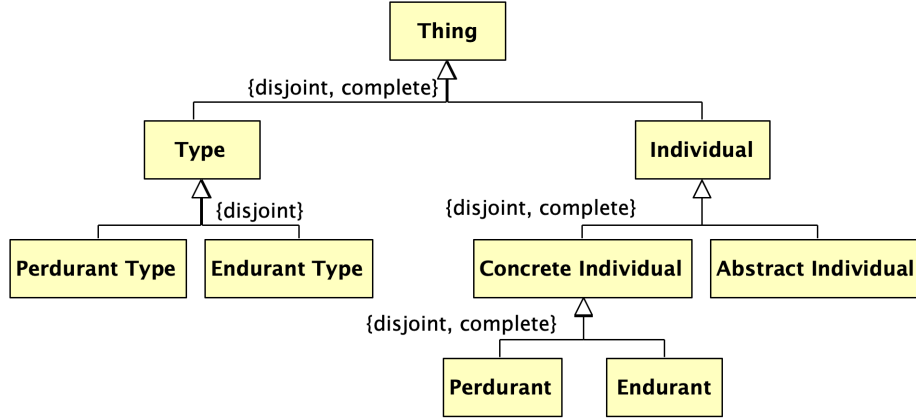


Figure 2: Visual representation of UFO's taxonomy of Thing.

```

4  (thing x))
5  )
6  )

```

- a2** There is no x such that it is a Type and an Individual.

$$\neg \exists x (\text{type_}(x) \wedge \text{individual}(x))$$

```

7  (cl-text ax_thing_partition
8  (not (exists (x)
9  (and (type_ x) (individual x)))
10 )
11 )

```

- a3** For every x , x is an Individual iff x is either a Concrete Individual or an Abstract Individual.

$$\forall x (\text{concreteIndividual}(x) \vee \text{abstractIndividual}(x) \leftrightarrow \text{individual}(x))$$

```

12 (cl-text ax_individual_taxonomy
13 (forall (x)
14 (iff (or (concreteIndividual x) (abstractIndividual x))
15 (individual x))
16 )
17 )

```

- a4** There is no x such that it is a Concrete Individual and an Abstract Individual.

$$\neg \exists x (\text{concreteIndividual}(x) \wedge \text{abstractIndividual}(x))$$

```

18 (cl-text ax_individual_partition
19 (not (exists (x)
20 (and (concreteIndividual x) (abstractIndividual x)))
21 )
22 )

```

- a5** For every x , x is a Concrete Individual iff x is either a Perdurant or an Endurant.

$$\forall x (\text{endurant}(x) \vee \text{perdurant}(x) \leftrightarrow \text{concreteIndividual}(x))$$

```

23 (cl-text ax_concreteIndividual_taxonomy
24 (forall (x)
25   (iff (or (endurant x) (perdurant x))
26         (concreteIndividual x))
27   )
28 )

```

a6 There is no x such that it is a Perdurant and an Endurant.

$\neg \exists x(\text{endurant}(x) \wedge \text{perdurant}(x))$

```

29 (cl-text ax_concreteIndividual_partition
30 (not (exists (x)
31   (and (endurant x) (perdurant x))
32   )
33 )

```

a7 For every x , x is a Concrete Individual iff x is either a Perdurant or an Endurant.

$\forall x(\text{endurantType}(x) \vee \text{perdurantType}(x) \rightarrow \text{type_}(x))$

```

34 (cl-text ax_type_taxonomy
35 (forall (x)
36   (if (or (endurantType x) (perdurantType x))
37       (type_ x))
38   )
39 )

```

a8 There is no x such that it is a Perdurant Type and an Endurant Type.

$\neg \exists x(\text{endurantType}(x) \wedge \text{perdurantType}(x))$

```

40 (cl-text ax_type_partition
41 (not (exists (x)
42   (and (endurantType x) (perdurantType x))
43   )
44 )

```

2.2 Partial Taxonomy of UFO: Abstract Individual

This subsection presents a portion of UFO's taxonomy specializing the type Abstract Individual (Figure 3).

a9 Every x that is a Quale is also an Abstract Individual.

$\forall x(\text{quale}(x) \rightarrow \text{abstractIndividual}(x))$

```

1 (cl-text ax_abstractIndividual_taxonomy_quale
2 (forall (x)
3   (if (quale x)
4       (abstractIndividual x))
5   )
6 )

```

a10 Every x that is a Set is also an Abstract Individual.

$\forall x(\text{set_}(x) \rightarrow \text{abstractIndividual}(x))$

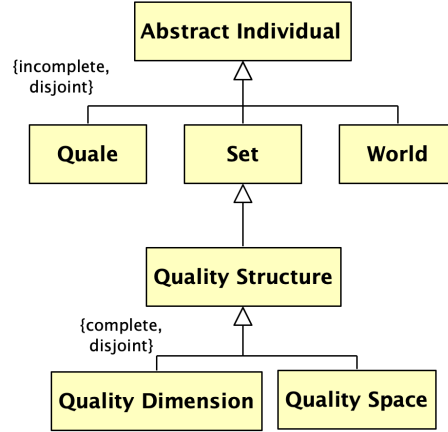


Figure 3: Visual representation of UFO's taxonomy of Abstract Individual.

```

7 (cl-text ax_abstractIndividual_taxonomy_set
8 (forall (x)
9   (if (set_ x)
10    (abstractIndividual x))
11   )
12 )

```

a11 Every x that is a World is also an Abstract Individual.

$\forall x(\text{world}(x) \rightarrow \text{abstractIndividual}(x))$

```

13 (cl-text ax_abstractIndividual_taxonomy_world
14 (forall (x)
15   (if (world x)
16    (abstractIndividual x))
17   )
18 )

```

a12 There is no x such that it is a Quale, a Set, and a World (pairwise disjoint).

$\neg \exists x((\text{quale}(x) \wedge \text{set}_-(x)) \vee (\text{quale}(x) \wedge \text{world}(x)) \vee (\text{set}_-(x) \wedge \text{world}(x)))$

```

19 (cl-text ax_abstractIndividual_pairwiseDisjoint
20 (not (exists (x)
21   (or (and (quale x) (set_ x)) (and (quale x) (world x)) (and (set_ x) (world x)
22   )))
23 )

```

a13 Every x that is a Quality Structure is also a Set.

$\forall x(\text{qualityStructure}(x) \rightarrow \text{set}_-(x))$

```

24 (cl-text ax_set_taxonomy_qualityStructure
25 (forall (x)
26   (if (qualityStructure x)
27    (set_ x))
28   )
29 )

```

a14 For every x , x is a Quality Structure iff x is either a Quality Dimension or a Quality Space.

$$\forall x(\text{qualityDimension}(x) \vee \text{qualitySpace}(x) \leftrightarrow \text{qualityStructure}(x))$$

```

30 (cl-text ax_qualityStructure_taxonomy
31 (forall (x)
32   (iff (or (qualityDimension x) (qualitySpace x))
33         (qualityStructure x))
34 )
35 )

```

a15 There is no x such that it is a Quality Dimension and a Quality Space.

$$\neg \exists x(\text{qualityDimension}(x) \wedge \text{qualitySpace}(x))$$

```

36 (cl-text ax_qualityStructure_partition
37 (not (exists (x)
38   (and (qualityDimension x) (qualitySpace x)))
39 )
40 )

```

2.3 Partial Taxonomy of UFO: Endurant

This subsection presents a portion of UFO's taxonomy specializing the type Endurant (Figure 4).

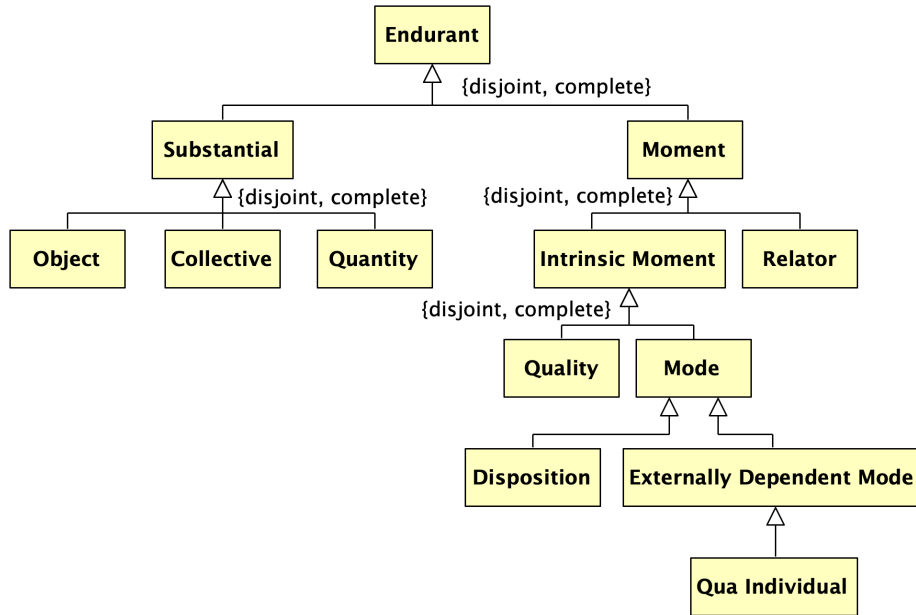


Figure 4: Visual representation of UFO's taxonomy of Endurant.

a16 For every x , x is an Endurant iff x is either a Substantial or a Moment.

$$\forall x(\text{substantial}(x) \vee \text{moment}(x) \leftrightarrow \text{endurant}(x))$$

```

1 (cl-text ax_endurant_taxonomy
2 (forall (x)
3   (iff (or (substantial x) (moment x))
4     (endurant x))
5   )
6 )

```

a17 There is no x such that it is a Substantial and a Moment.

$\neg \exists x(\text{substantial}(x) \wedge \text{moment}(x))$

```

7 (cl-text ax_endurant_partition
8 (not (exists (x)
9   (and (substantial x) (moment x)))
10 )
11 )

```

a18 For every x , x is a Substantial iff x is either an Object, a Collective, or a Quantity.

$\forall x(\text{object}(x) \vee \text{collective}(x) \vee \text{quantity}(x) \leftrightarrow \text{substantial}(x))$

```

12 (cl-text ax_substantial_taxonomy
13 (forall (x)
14   (iff (or (object x) (collective x) (quantity x))
15     (substantial x))
16   )
17 )

```

a19 There is no x such that it is an Object, a Collective, and a Quantity (pairwise disjoint).

$\neg \exists x((\text{object}(x) \wedge \text{collective}(x)) \vee (\text{object}(x) \wedge \text{quantity}(x)) \vee (\text{collective}(x) \wedge \text{quantity}(x)))$

```

18 (cl-text ax_substantial_partition
19 (not (exists (x)
20   (or (and (object x) (collective x)) (and (object x) (quantity x)) (and (
21     collective x) (quantity x))))
22 )

```

a20 For every x , x is a Moment iff x is either an Intrinsic Moment or a Relator.

$\forall x(\text{intrinsicMoment}(x) \vee \text{relator}(x) \leftrightarrow \text{moment}(x))$

```

23 (cl-text ax_moment_taxonomy
24 (forall (x)
25   (iff (or (intrinsicMoment x) (relator x))
26     (moment x))
27   )
28 )

```

a21 There is no x such that it is an Intrinsic Moment and a Relator.

$\neg \exists x(\text{intrinsicMoment}(x) \wedge \text{relator}(x))$

```

29 (cl-text ax_moment_partition
30 (not (exists (x)
31   (and (intrinsicMoment x) (relator x)))
32 )
33 )

```

a22 For every x , x is an Intrinsic Moment iff x is either a Quality or a Mode.

$\forall x(\text{quality}(x) \vee \text{mode}(x) \leftrightarrow \text{intrinsicMoment}(x))$

```
34 (cl-text ax_intrinsicMoment_taxonomy
35 (forall (x)
36   (iff (or (quality x) (mode x))
37     (intrinsicMoment x))
38   )
39 )
```

a23 There is no x such that it is an Intrinsic Moment and a Relator.

$\neg \exists x(\text{quality}(x) \wedge \text{mode}(x))$

```
40 (cl-text ax_intrinsicMoment_partition
41 (not (exists (x)
42   (and (quality x) (mode x)))
43   )
44 )
```

a24 Every x that is a Disposition is also a Mode.

$\forall x(\text{disposition}(x) \rightarrow \text{mode}(x))$

```
45 (cl-text ax_mode_taxonomy_disposition
46 (forall (x)
47   (if (disposition x)
48     (mode x))
49   )
50 )
```

a25 Every x that is an Externally Dependent Mode is also a Mode.

$\forall x(\text{externallyDependentMode}(x) \rightarrow \text{mode}(x))$

```
51 (cl-text ax_mode_taxonomy_externallyDependentMode
52 (forall (x)
53   (if (externallyDependentMode x)
54     (mode x))
55   )
56 )
```

a26 Every x that is an Qua Individual is also an Externally Dependent Mode.

$\forall x(\text{quaIndividual}(x) \rightarrow \text{externallyDependentMode}(x))$

```
57 (cl-text ax_externallyDependentMode_taxonomy_quaIndividual
58 (forall (x)
59   (if (quaIndividual x)
60     (externallyDependentMode x))
61   )
62 )
```

2.4 Partial Taxonomy of UFO: Endurant Type by Ontological Natures

This subsection presents a portion of UFO's taxonomy specializing the type Endurant (Figure 5).

a27 For every x , x is an Endurant Type iff x is either a Substantial Type or a Moment Type.

$\forall x(\text{substantialType}(x) \vee \text{momentType}(x) \leftrightarrow \text{endurantType}(x))$

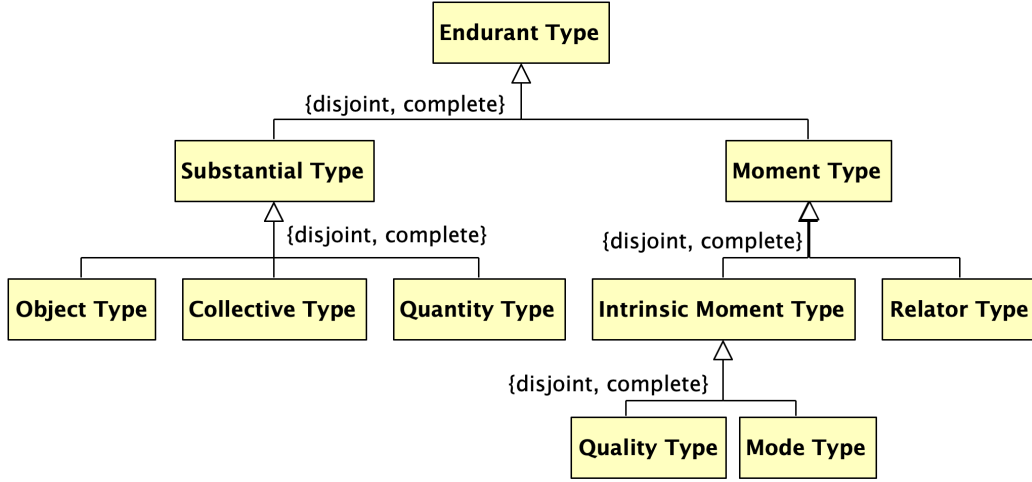


Figure 5: Visual representation of UFO's taxonomy of Endurant Type classified by ontological natures.

```

1 (cl-text ax_endurantType_taxonomy_nature
2 (forall (x)
3   (iff (or (substantialType x) (momentType x))
4     (endurantType x))
5 )
6 )

```

a28 There is no x such that it is a Substantial Type and a Moment Type.

$$\neg \exists x (\text{substantialType}(x) \wedge \text{momentType}(x))$$

```

7 (cl-text ax_endurantType_partition_nature
8 (not (exists (x)
9   (and (substantialType x) (momentType x))))
10 )
11 )

```

a29 For every x , x is a Substantial Type iff x is either an Object Type, a Collective Type, or a Quantity Type.

$$\forall x (\text{objectType}(x) \vee \text{collectiveType}(x) \vee \text{quantityType}(x) \leftrightarrow \text{substantialType}(x))$$

```

12 (cl-text ax_substantialType_taxonomy
13 (forall (x)
14   (iff (or (objectType x) (collectiveType x) (quantityType x))
15     (substantialType x))
16 )
17 )

```

a30 There is no x such that it is an Object Type, a Collective Type, and a Quantity Type (pairwise disjoint).

$$\neg \exists x ((\text{objectType}(x) \wedge \text{collectiveType}(x)) \vee (\text{objectType}(x) \wedge \text{quantityType}(x)) \vee (\text{collectiveType}(x) \wedge \text{quantityType}(x)))$$

```

18 (cl-text ax_substantialType_partition
19 (not (exists (x)
20   (or (and (objectType x) (collectiveType x)) (and (objectType x) (quantityType
21     x)) (and (collectiveType x) (quantityType x))))
22 )

```

- a31** For every x , x is a Moment Type iff x is either an Intrinsic Moment Type or a Relator Type.
 $\forall x(\text{intrinsicMomentType}(x) \vee \text{relatorType}(x) \leftrightarrow \text{momentType}(x))$

```

23 (cl-text ax_momentType_taxonomy
24 (forall (x)
25   (iff (or (intrinsicMomentType x) (relatorType x))
26     (momentType x))
27 )
28 )

```

- a32** There is no x such that it is an Intrinsic Moment Type and a Relator Type.
 $\neg \exists x(\text{intrinsicMomentType}(x) \wedge \text{relatorType}(x))$

```

29 (cl-text ax_momentType_partition
30 (not (exists (x)
31   (and (intrinsicMomentType x) (relatorType x)))
32 )
33 )

```

- a33** For every x , x is an Intrinsic Moment Type iff x is either a Quality Type or a Mode Type.
 $\forall x(\text{qualityType}(x) \vee \text{modeType}(x) \leftrightarrow \text{intrinsicMomentType}(x))$

```

34 (cl-text ax_intrinsicMomentType_taxonomy
35 (forall (x)
36   (iff (or (qualityType x) (modeType x))
37     (intrinsicMomentType x))
38 )
39 )

```

- a34** There is no x such that it is an Intrinsic Moment Type and a Relator Type.
 $\neg \exists x(\text{qualityType}(x) \wedge \text{modeType}(x))$

```

40 (cl-text ax_intrinsicMomentType_partition
41 (not (exists (x)
42   (and (qualityType x) (modeType x)))
43 )
44 )

```