OBO-Format and Obolog Specification (1.3) DRAFT

Chris Mungall

October 8, 2009

Contents

1 Introduction **3**
1.1 Background . . . 3
1.2 Core Concepts . . . 4
1.3 Formalization . . . 4

2 Obolog Semantics **7**
2.1 type . . . 7
2.2 instance . . . 8
2.3 relation . . . 9
2.4 annotation . . . 9
2.5 type_type . . . 9
2.6 instance_instance . . . 10
2.7 subrelation . . . 11
2.8 all_some_all_times . . . 12
2.9 all_some . . . 13
2.10 all_only . . . 14
2.11 all_some_tr . . . 15
2.12 all_some_in_reference_context . . . 15
2.13 holds_atemporally_between . . . 16
2.14 holds_temporally_between . . . 16
2.15 homeomorphic_for . . . 17
2.16 domain . . . 17
2.17 range . . . 18
2.18 inverse_of . . . 18
2.19 inverse_of_on_instance_level . . . 19
2.20 holds_bidirectionally_for . . . 19

2.21 functional 20
2.22 inverse_functional 20
2.23 bijective 21
2.24 reflexive 21
2.25 irreflexive 21
2.26 transitive 22
2.27 symmetric 22
2.28 antisymmetric 23
2.29 cyclic 23
2.30 directed_path_over 24
2.31 directed_simple_path_over 24
2.32 cyclic_over 24
2.33 acyclic 24
2.34 proper_subrelation 24
2.35 transitive_over 25
2.36 holds_over_chain 25
2.37 disjoint_over 26
2.38 maximal_over 27
2.39 disjoint_from 27
2.40 relation_arity 28
2.41 relation_min_arity 28
2.42 relation_max_arity 28
2.43 posits 28
2.44 namespace 28

3 Obolog Annotations **30**
3.1 Obolog semantics 30
3.2 Obolog-A semantics 30

4 OBO Syntax **31**
4.1 Relation to OBO-Format 1.2 31
4.2 OBO Lexical Rules 31
4.3 OBO Document Structure 32
4.4 OBO Header 32
4.5 Term Stanzas 33
4.6 Typedef Stanzas 36
4.7 Instance Stanzas 38
4.8 Annotation Stanzas 39
4.9 Formula Stanzas 40
4.10 ID Macros 41

4.11 Additional considerations 41

5 OBO Semantics 43
5.1 Mapping to Obolog 43
5.2 Identifiers 43
5.3 Imports 43
5.4 Translation Table 43

6 Obolog Sublanguages and Superlanguages 48
6.1 CL 48
6.2 IKL 48
6.3 obolog^{CL} 48
6.4 obolog^{Core} 48
6.5 obolog^{A} 48
6.6 obolog^{H} and obolog^{Data} 49

7 Translating Obolog to OWL 53
7.1 Standard DL Translation 53
7.2 Simplified Translation (not normative) 60
7.3 OWL Full (RDFS) Translation 62
7.4 Obolog-lex Translation 63
7.5 Obolog-A Translation 63

8 Glossary 64

1 Introduction

OBO-Format is an exchange format for ontologies and associated information. This document provides a formalization of both the syntax and semantics of OBO-Format version 1.3.

The syntax is specified as a BNF grammar. The semantics are defined in terms of a translation to first order logic - more specifically, to the Obolog language, a collection of constructs defined using the Common Logic ISO standard.

1.1 Background

OBO Format 1.0 was devised in 2001, originally as a successor to the original Gene Ontology DAG file format. The original design was informed by local pragmatic considerations, and sacrified syntactic and semantic rigour in favour of simplicity. OBO Format conceives of ontologies in graph-oriented terms, with each node of the graph described using a set of tag-value pairs.

OBO Format 1.2. was initiated in 2003 finalized in 2006, and extended 1.0 with tags borrowed from the OWL (originally DAML+OIL) language. In 2007 Ian Horrocks suggested a formalization of both syntax and semantics in terms of OWL1 and OWL2.

1.2 Core Concepts

1.2.1 Types, instances, relations and annotations

The universe of discourse of OBO consists of entities, which are divided into types, instances, relations and annotations. This division is exhaustive and pairwise disjoint (i.e. every entity is exactly one of these things).

Instances are spatiotemporal particulars, whereas types are patterns which are shared by collections of like instances. Types are denoted by terms such as “Lung” or “Apoptosis”.

Relations obtain between entities. In OBO, relations can hold between entities of any kind (even relations). If a relation Rel holds between two or more entities we write this as a tuple of Rel:

```
Rel(Argument1, Argument2, ... ArgumentN)
```

Where Rel is drawn from a collection of relations, and Arguments 1 to N are drawn from the collection of entities.

In cases where the tuple has a single argument, we conventionally call the relation a *unary predicate*.

1.2.2 Ontology Graphs

Note that whilst tuples can have any number of arguments, OBO is geared towards relations that either take 2 or 3 arguments (i.e. 2-ary and 3-ary relations). These relations are conceived of in graph-theoretic terms, with each entity constituting a node in the graph, and each tuple consituting an egde, between the first argument (called the “child” or “subject”) and the second argument (called the “parent”, “object” or “target”). The relation acts as edge-label, with additional arguments as edge properties.

1.3 Formalization

This specification provides the normative formalization of OBO-Format in terms of a language called *Obolog*. Obolog is a collection of predicates and functions with formally defined semantics that can be used to express ontology graphs.

- Every OBO-Format document has a normative interpretation as an Obolog text. Obolog is expressed using Common Logic (CL) [TODO: currently using KIF, which can be automatically translated to CLIF, a CL syntax].
- Every Obolog text is a CL text(s?).
- Obolog can be used independently of OBO-Format, any CL syntax can be used.

We distinguish between two sub-languages: Obolog-core and Obolog-full. An Obolog-core text is a CL text that contains no quantified sentences (possibly other restrictions?). Obolog-full is Obolog-core extended with arbitrary quantified sentences. We distinguish between trivial and non-trivial membership in these classes depending on whether the text contains any sentences using Obolog predicates.

1.3.1 Translation to other FOL syntaxes

CL is a highly expressive language for First Order Logic. Other FOL syntaxes may disallow certain constructs: for example, variable as first argument in a sentence.

It is possible to translate to other syntaxes, such as Prover9 syntax, by treating the definitions of Obolog predicates as macro-expansion rules. For example, The Obolog definition of transitivity for binary relations is as follows:

$$\begin{aligned} &\mathsf{transitive}(rel) \wedge \\ &\quad rel(X,Y) \wedge \\ &\qquad rel(Y,Z) \quad \rightarrow \quad rel(X,Z) \end{aligned}$$

We can use this to expand the sentence $transitive(develops\_from)$ to:

$$\begin{aligned} &develops\_from(X,Y) \wedge \\ &\quad develops\_from(Y,Z) \quad \rightarrow develops\_from(X,Z) \end{aligned}$$

In future we may define sublanguages based on FOL profile; for example Obolog-Horn for Horn rules, which is useful for rule systems and relational databases.

1.3.2 Translation to OWL

Previous attempts to formalize OBO-Format in terms of OWL did so via a direct translation of OBO-Format. This can now be done in terms of Obolog.

1.3.3 The Relation Ontology and The Basic Formal Ontology

We attempt to be as ontologically neutral as possible. Certain ontological assumptions are assumed: a division between Continuants and Occurrents.

2 Obolog Semantics

This section specifies the semantics of the core Obolog language in terms of first order logic axioms.

Each Obolog predicate is listed, along with:

- A short natural language description of that predicate
- Examples of actual or potential use of this predicate in OBO ontologies
- Properties of this predicate, described in the Obolog language itself
- Axioms and theorems pertaining to that predicate

The specification is entirely in terms of constructs that can be expressed in Common Logic. We import some predicates from the OBO Relation Ontology (RO), specifically:

- instance_of
- is_a
- exists_at

2.1 type

unary predicate which holds over types. Types are patterns in reality, and each type is instantiated by at least one instance at some time.

Note that types are within the domain of discourse in Obolog. Relations can hold between types.

2.1.1 Examples

- FMA :

$$\mathsf{type}(\mathsf{Left\_hand})$$

- FMA :

$$\mathsf{type}(\mathsf{Lung})$$

- PATO :

$$\mathsf{type}(\mathsf{Red})$$

- PATO :

$$\mathsf{type}(\mathsf{Shape})$$

2.1.2 Axioms and Theorems

Axiom: every type has at least one instance

$$\mathsf{type}(U) \quad \rightarrow \quad \exists i[\mathsf{instance\_of}(i, U)] \vee \exists i, t[\mathsf{instance\_of}(i, U, t)]$$

Axiom: type_type relations hold between types

$$\begin{aligned}\mathsf{type\_type}(r) \quad \leftrightarrow \quad & \forall a, b[r(a, b) \rightarrow \mathsf{type}(a) \wedge \\ & \mathsf{type}(b)] \wedge \\ & \forall a, b[r(a, b, t) \rightarrow \mathsf{type}(a) \wedge \\ & \mathsf{type}(b)]\end{aligned}$$

2.2 instance

unary predicate which holds over instances. Instances are spatiotemporal particulars, every instance instantiates a type.

2.2.1 Examples

- A particular instance. :

$$\mathsf{instance(lung\_of\_patient\_02345)}$$

- A particular instance. :

$$\mathsf{instance(shape\_of\_lung\_of\_patient\_02345)}$$

2.2.2 Axioms and Theorems

Axiom: every instance is an instance of some type

$$\mathsf{instance}(i) \quad \rightarrow \quad \exists U[\mathsf{instance\_of}(i, U)] \vee \exists U, t[\mathsf{instance\_of}(i, U, t)]$$

Axiom: instance_instance relations hold between instances

$$\begin{aligned}\mathsf{instance\_instance}(r) \quad \leftrightarrow \quad & \forall a, b[r(a, b) \rightarrow \mathsf{instance}(a) \wedge \\ & \mathsf{instance}(b)] \wedge \\ & \forall a, b[r(a, b, t) \rightarrow \mathsf{instance}(a) \wedge \\ & \mathsf{instance}(b)]\end{aligned}$$

2.3 relation

unary predicate which holds over relations. Relations constitute the edge labels on ontology graphs.

Note that relations are not constrained to be binary. Relations are part of the domain of discourse in Obolog. However, they can be 'compiled out' by translating them to predicates, treating meta-relation axioms as macros.

2.3.1 Examples

- RO :

$$\mathsf{relation}(\mathsf{part\_of})$$

- RO :

$$\mathsf{relation}(\mathsf{is\_a})$$

- RO :

$$\mathsf{relation}(\mathsf{instance\_of})$$

2.3.2 Axioms and Theorems

Axiom: instance_instance holds only for relations

$$\mathsf{instance\_instance}(r) \quad \rightarrow \quad \mathsf{relation}(r)$$

Axiom: type_type holds only for relations

$$\mathsf{type\_type}(r) \quad \rightarrow \quad \mathsf{relation}(r)$$

2.4 annotation

unary predicate which holds over annotations. Annotations are reified sentences.

Axioms pertaining to annotations are dealt with in a separate theory.

2.4.1 Axioms and Theorems

2.5 type_type

unary predicate which holds over relations whose domain and range are types

unary predicate which holds over relations whose domain and range are instances

2.5.1 Axioms and Theorems

Axiom: type_type relations hold between types

$$\begin{aligned}\mathsf{type\_type}(r) \quad &\leftrightarrow \quad \forall a, b[r(a,b) \rightarrow \mathsf{type}(a) \wedge \\ &\qquad \mathsf{type}(b)] \wedge \\ &\qquad \forall a, b[r(a,b,t) \rightarrow \mathsf{type}(a) \wedge \\ &\qquad \mathsf{type}(b)]\end{aligned}$$

Axiom: type_type holds only for relations

$$\mathsf{type\_type}(r) \quad \rightarrow \quad \mathsf{relation}(r)$$

Axiom: all_only relates from type level relations

$$\mathsf{all\_only}(tr, ir) \quad \rightarrow \quad \mathsf{type\_type}(tr)$$

$$\mathsf{all\_some}(tr, ir) \quad \rightarrow \quad \mathsf{type\_type}(tr)$$

Axiom: all_some_all_times holds between an instance-instance relation and a type-type relation

$$\begin{aligned}\mathsf{all\_some\_all\_times}(tr, ir) \quad &\rightarrow \quad \mathsf{instance\_instance}(ir) \wedge \\ &\qquad \mathsf{type\_type}(tr)\end{aligned}$$

2.6 instance_instance

2.6.1 Axioms and Theorems

Axiom: instance_instance relations hold between instances

$$\begin{aligned}\mathsf{instance\_instance}(r) \quad &\leftrightarrow \quad \forall a, b[r(a,b) \rightarrow \mathsf{instance}(a) \wedge \\ &\qquad \mathsf{instance}(b)] \wedge \\ &\qquad \forall a, b[r(a,b,t) \rightarrow \mathsf{instance}(a) \wedge \\ &\qquad \mathsf{instance}(b)]\end{aligned}$$

Axiom: instance_instance holds only for relations

$$\mathsf{instance\_instance}(r) \rightarrow \mathsf{relation}(r)$$

Axiom: all_only relates to instance level relations

$$\mathsf{all\_only}(tr, ir) \rightarrow \mathsf{instance\_instance}(ir)$$

$$\mathsf{all\_some}(tr, ir) \rightarrow \mathsf{instance\_instance}(ir)$$

Axiom: all_some_all_times holds between an instance-instance relation and a type-type relation

$$\mathsf{all\_some\_all\_times}(tr, ir) \rightarrow \mathsf{instance\_instance}(ir) \wedge \mathsf{type\_type}(tr)$$

2.7 subrelation

a transitive meta-level relation between two relations r1 and r2, such that if r1 holds between a and b (optionally: t) then r2 must hold between a and b (t)

2.7.1 Examples

- RO :

$$\mathsf{subrelation}(\mathsf{agent\_in}, \mathsf{participates\_in})$$

- RO :

$$\mathsf{subrelation}(\mathsf{proper\_part\_of}, \mathsf{part\_of})$$

- GO: Every negative regulation relationship is necessarily a regulation relationship :

$$\mathsf{subrelation}(\mathsf{negatively\_regulates}, \mathsf{regulates})$$

2.7.2 Axioms and Theorems

- transitive

Axiom: If a binary relation holds, binary subrelations necessarily hold [derived from transitivity axiom]

$$\begin{array}{l} \mathsf{subrelation}(r1, r2) \wedge \\ \qquad r1(x,y) \quad \rightarrow \quad r2(x,y) \end{array}$$

Axiom: If a ternary relation holds, ternary subrelations necessarily hold [derived from transitivity axiom]

$$\begin{array}{l} \mathsf{subrelation}(r1, r2) \wedge \\ \qquad r1(x,y,t) \quad \rightarrow \quad r2(x,y,t) \end{array}$$

$$\begin{array}{rcl} \mathsf{proper\_subrelation}(pr, r) & \rightarrow & \mathsf{irreflexive}(pr) \wedge \\ & & \mathsf{subrelation}(pr, r) \end{array}$$

2.8 all_some_all_times

relates a type-level relation ¡i¿tr¡/i¿ to its instance-level counterpart ¡b¿ir¡/b¿, in a temporally invariant way such that for two types, A and B, related by ¡i¿tr¡/i¿, it is the case that all instances of A are related by ¡b¿ir¡/b¿ to some instance of B at all times for which the instance of A exists

The examples here assume the type-level relation is indicated using the suffix '_some', but best practice has not yet been decided

2.8.1 Examples

- RO :

$$\mathsf{all\_some\_all\_times}(\mathsf{part\_of\_some}, \mathsf{part\_of})$$

- GO :

$$\begin{array}{rcl} \mathsf{part\_of\_some}(\mathsf{nucleus}, \mathsf{cell}) & \rightarrow & \forall n, t[\mathsf{instance\_of}(n, \mathsf{nucleus}, t) \rightarrow \exists c[\mathsf{instance\_of}(c, \mathsf{cell}, t) \\ & & \mathsf{part\_of}(n, c, t)]] \end{array}$$

2.8.2 Axioms and Theorems

- functional

Axiom: if an all-some-all-times relations holds at the type level between A and B, it holds for all instances of A to some instance of B at all times that the instance of A exists

$$\begin{aligned}\mathsf{all\_some\_all\_times}(tr, ir) \quad \rightarrow \quad & tr(A, B) \wedge \\ & \mathsf{instance\_of}(ai, A, t) \rightarrow \exists bi[\mathsf{instance\_of}(bi, B, t) \wedge \\ & ir(ai, bi, t)]\end{aligned}$$

Axiom: all_some_all_times holds between an instance-instance relation and a type-type relation

$$\begin{aligned}\mathsf{all\_some\_all\_times}(tr, ir) \quad \rightarrow \quad & \mathsf{instance\_instance}(ir) \wedge \\ & \mathsf{type\_type}(tr)\end{aligned}$$

2.9 all_some

relates a type-level relation ¡i¿tr¡/i¿ to its instance-level counterpart ¡b¿ir¡/b¿, in an atemporal way, such that for two types, A and B, related by ¡i¿tr¡/i¿, it is the case that all instances of A are related by ¡b¿ir¡/b¿ to some instance of B

Corresponds to an existential restriction in OWL

2.9.1 Examples

- RO :

$$\begin{gathered}\mathsf{all\_some}(\mathsf{part\_of\_some}, \mathsf{part\_of}) \wedge \\ \mathsf{part\_of\_some}(\mathsf{mitosis}, \mathsf{M\_phase\_of\_mitotic\_cell\_cycle})\end{gathered}$$

2.9.2 Axioms and Theorems

$$\mathsf{all\_some}(tr, ir) \quad \rightarrow \quad \mathsf{instance\_instance}(ir)$$

$$\mathsf{all\_some}(tr, ir) \quad \rightarrow \quad \mathsf{type\_type}(tr)$$

$$\mathsf{all\_some}(tr, ir) \rightarrow tr(A, B) \wedge \mathsf{instance\_of}(ai, A) \rightarrow \exists bi[\mathsf{instance\_of}(bi, B) \wedge ir(ai, bi)]$$

$$\mathsf{inverse\_of\_on\_instance\_level}(r, r2) \leftrightarrow \mathsf{all\_some}(r, rp) \wedge \mathsf{all\_some}(r2, rp2) \wedge \mathsf{inverse\_of}(r2, rp2)$$

2.10 all_only

relates a type-level relation ¡i¿tr¡/i¿ to its instance-level counterpart ¡b¿ir¡/b¿, in an atemporal way, such that for two types, A and B, related by ¡i¿tr¡/i¿, it is the case that no instances of A are related by ¡b¿ir¡/b¿ to something that is not an instance of B

2.10.1 Axioms and Theorems

Axiom: all_only definition

$$\mathsf{all\_only}(tr, ir) \leftrightarrow tr(A, B) \wedge \mathsf{instance\_of}(ai, A) \rightarrow \neg(\exists bi[\neg(\mathsf{instance\_of}(bi, B)) \wedge ir(ai, bi)])$$

Axiom: all_only relates to instance level relations

$$\mathsf{all\_only}(tr, ir) \rightarrow \mathsf{instance\_instance}(ir)$$

Axiom: all_only relates from type level relations

$$\mathsf{all\_only}(tr, ir) \rightarrow \mathsf{type\_type}(tr)$$

Thereom: propagation of all_only relations under is_a

$$\mathsf{all\_only}(tr, ir) \rightarrow tr(A, B) \wedge \mathsf{is\_a}(B, C) \rightarrow tr(A, C)$$

Thereom: propagation of all_only relations over is_a

$$\mathsf{all\_only}(tr, ir) \quad \rightarrow \quad \mathsf{is\_a}(A, B) \wedge$$
$$tr(B, C) \rightarrow tr(A, C)$$

2.11 all_some_tr

relates a type-level relation ¡i¿tr¡/i¿ to its instance-level counterpart ¡b¿ir¡/b¿, such that for two types, A and B, related by ¡i¿tr¡/i¿, it is the case that all instances of A stand ing a ¡b¿ir¡/b¿ relation to some B for some time, and neither becomes detached or starts in a detached state

2.11.1 Axioms and Theorems

Axiom: all_some_tr definition

$$\begin{aligned}
\mathsf{all_some_tr}(tr, ir) \quad \leftrightarrow \quad & tr(A, B) \wedge \\
& \mathsf{instance_of}(ai, A, t) \rightarrow \exists bi[\exists t1[\mathsf{instance_of}(bi, B, t1) \wedge \\
& ir(ai, bi, t1)] \wedge \\
& =¿(\mathsf{exists_at}(ai, t2) \wedge \\
& \mathsf{exists_at}(b1, t2) \wedge \\
& ir(ai, bi, t2))]
\end{aligned}$$

2.12 all_some_in_reference_context

relates a type-level relation ¡i¿tr¡/i¿ to its instance-level counterpart ¡b¿ir¡/b¿, such that for two types, A and B, related by ¡i¿tr¡/i¿, it is the case that all instances of A stand in a ¡b¿ir¡/b¿ relation to some B where both instances stand in relation r2 to the same entity

See Neuhaus, Osumi-Sutherland for details

2.12.1 Examples

- FBdv :

$$\begin{aligned}
&\mathsf{all_some_in_reference_context}(\mathsf{begins_at_end_of_r}, \mathsf{begins_at_end_of}, \mathsf{reference_process}) \wedge \\
&\mathsf{range}(\mathsf{reference_process}, \mathsf{life}) \wedge \\
&\mathsf{begins_at_end_of_r}(\mathsf{germ_band_retraction}, \mathsf{embryonic_stage_11})
\end{aligned}$$

2.12.2 Axioms and Theorems

$$\begin{aligned}\mathsf{all\_some\_in\_reference\_context}(tr, ir, rr) \quad \rightarrow \quad & tr(A, B) \wedge \\ & \mathsf{instance\_of}(ai, A) \rightarrow \exists bi[\mathsf{instance\_of}(bi, B) \wedge \\ & rr(ai, ri) \rightarrow rr(ai, ri) \wedge \\ & ir(ai, bi)]\end{aligned}$$

2.13 holds_atemporally_between

A relation R holds atemporally between two types A and B if for all R(a,b), it is the case that a and b are instances of A and B

2.13.1 Examples

- RO :

$$\mathsf{holds\_atemporally\_between}(\mathsf{part\_of}, \mathsf{Occurrent}, \mathsf{Occurrent})$$

2.13.2 Axioms and Theorems

Axiom: holds_atemporally_between definition

$$\begin{aligned}\mathsf{holds\_atemporally\_between}(rel, U1, U2) \quad \leftrightarrow \quad & rel(i1, i2) \rightarrow \mathsf{instance\_of}(i1, U1) \wedge \\ & \mathsf{instance\_of}(i2, U2)\end{aligned}$$

2.14 holds_temporally_between

A relation R holds temporally between two types A and B if for all R(a,b,t), it is the case that a and b are instances of A and B

2.14.1 Examples

- RO :

$$\mathsf{holds\_temporally\_between}(\mathsf{part\_of}, \mathsf{Continuant}, \mathsf{Continuant})$$

2.14.2 Axioms and Theorems

Axiom: holds_temporally_between definition

$$\mathsf{holds\_temporally\_between}(rel, U1, U2) \leftrightarrow rel(i1, i2, t) \rightarrow \mathsf{instance\_of}(t, \mathsf{span{:}TemporalRegion}) \wedge \mathsf{instance\_of}(i1, U1, t) \wedge \mathsf{instance\_of}(i2, U2, t)$$

2.15 homeomorphic_for

A relation is homeomorphic for a particular type if the relation always holds between like types

2.15.1 Examples

- BFO :

$$\mathsf{homeomorphic\_for}(\mathsf{part\_of}, \mathsf{IndependentContinuant})$$

- BFO :

$$\mathsf{homeomorphic\_for}(\mathsf{part\_of}, \mathsf{Process})$$

2.15.2 Axioms and Theorems

Axiom: homeomorphic atemporal relations

$$\mathsf{homeomorphic\_for}(r, U) \rightarrow r(a, bt) \wedge \mathsf{instance\_of}(a, U) \rightarrow \mathsf{instance\_of}(b, U)$$

Axiom: homeomorphic time-indexed relations

$$\mathsf{homeomorphic\_for}(r, U) \rightarrow r(a, b, t) \wedge \mathsf{instance\_of}(a, U, t) \rightarrow \mathsf{instance\_of}(b, U, t)$$

2.16 domain

Constrains relations such that the subject (first argument) of the relation only holds between instances of the specified type

2.16.1 Axioms and Theorems

Axiom: domain constraints on atemporal relations

$$\begin{aligned} \mathsf{domain}(rel, D) \wedge & \\ rel(i1, i2) &\rightarrow \mathsf{instance\_of}(i2, D) \end{aligned}$$

Axiom: domain constraints on time-indexed relations

$$\begin{aligned} \mathsf{domain}(rel, D) \wedge & \\ rel(i1, i2, t) &\rightarrow \mathsf{instance\_of}(i2, D, t) \end{aligned}$$

2.17 range

Constrains relations such that the object (second argument) of the relation only holds between instances of the specified type

2.17.1 Axioms and Theorems

Axiom: range constraints on atemporal relations

$$\begin{aligned} \mathsf{range}(rel, R) \wedge & \\ rel(i1, i2t) &\rightarrow \mathsf{instance\_of}(i1, D) \end{aligned}$$

Axiom: range constraints on time-indexed relations

$$\begin{aligned} \mathsf{range}(rel, R) \wedge & \\ rel(i1, i2, t) &\rightarrow \mathsf{instance\_of}(i1, D, t) \end{aligned}$$

2.18 inverse_of

holds between two relations such that a sentence of one implies a sentence of the other, with 1st and 2nd arguments swapped.

note that this should not be applied to type_type all_some relations

2.18.1 Axioms and Theorems

- symmetric

$$\mathsf{inverse\_of}(r, s) \leftrightarrow r(a, b) \rightarrow s(b, a) \wedge r(a, b, t) \rightarrow s(b, a, t)$$

$$\mathsf{inverse\_of\_on\_instance\_level}(r, r2) \leftrightarrow \mathsf{all\_some}(r, rp) \wedge \mathsf{all\_some}(r2, rp2) \wedge \mathsf{inverse\_of}(r2, rp2)$$

2.19 inverse_of_on_instance_level

holds between two relations such that their instance level counterparts are inverses.

2.19.1 Axioms and Theorems

- symmetric

$$\mathsf{inverse\_of\_on\_instance\_level}(r, r2) \leftrightarrow \mathsf{all\_some}(r, rp) \wedge \mathsf{all\_some}(r2, rp2) \wedge \mathsf{inverse\_of}(r2, rp2)$$

2.20 holds_bidirectionally_for

holds_bidirectionally_for(SR,R,Inv), X SR Y =¿ X R Y and Y Inv X

2.20.1 Examples

- - :

  holds_bidirectionally_for(integral_part_of, part_of_some, has_part_some)

2.20.2 Axioms and Theorems

$$\mathsf{holds\_bidirectionally\_for}(sr, r, ir) \leftrightarrow sr(a, b) \rightarrow r(a, b) \wedge ir(b, a)$$

2.21 functional

A functional relation acts like a function in that the subject relates to a single object.

2.21.1 Axioms and Theorems

$$\mathsf{bijective}(r) \leftrightarrow \mathsf{functional}(r) \wedge \mathsf{inverse\_functional}(r)$$

Axiom: functional atemporal relations

$$\mathsf{functional}(rel) \wedge rel(x, y1) \wedge rel(x, y2) \rightarrow \mathsf{equivalent\_to}(y1, y2)$$

Axiom: functional time-indexed relations

$$\mathsf{functional}(rel) \wedge rel(x, y1, t) \wedge rel(x, y2, t) \rightarrow \mathsf{equivalent\_to}(y1, y2)$$

2.22 inverse_functional

A inverse_functional relation acts like a function in that the object relates to a single subject.

2.22.1 Axioms and Theorems

$$\mathsf{bijective}(r) \leftrightarrow \mathsf{functional}(r) \wedge \mathsf{inverse\_functional}(r)$$

Axiom: inverse_functional atemporal relations

$$\mathsf{inverse\_functional}(rel) \wedge rel(x1, y) \wedge rel(x2, y) \rightarrow \mathsf{equivalent\_to}(x1, x2)$$

Axiom: inverse_functional time-indexed relations

$$\begin{aligned} &\mathsf{inverse\_functional}(rel) \wedge \\ &\quad rel(x1, y, t) \wedge \\ &\quad\quad rel(x2, y, t) \quad \rightarrow \quad \mathsf{equivalent\_to}(x1, x2) \end{aligned}$$

2.23 bijective

2.23.1 Axioms and Theorems

$$\begin{aligned} \mathsf{bijective}(r) \quad \leftrightarrow \quad & \mathsf{functional}(r) \wedge \\ & \mathsf{inverse\_functional}(r) \end{aligned}$$

2.24 reflexive

A reflexive relation always holds between an entity and itself

2.24.1 Axioms and Theorems

Axiom: reflexivity of atemporal relations: if the relation ever holds for an entity, it holds between that entity and itself

$$\begin{aligned} &\mathsf{reflexive}(rel) \wedge \\ &\quad \exists b[rel(a, b)] \quad \rightarrow \quad rel(a, a) \end{aligned}$$

Axiom: reflexivity of time-indexed relations: if the relation ever holds for an entity at some time, it holds between that entity and itself at all times

$$\begin{aligned} &\mathsf{reflexive}(rel) \wedge \\ &\exists b, t[rel(a, b, t)] \quad \rightarrow \quad rel(a, a, t2) \end{aligned}$$

2.25 irreflexive

An irreflexive relation never holds between an entity and itself

2.25.1 Axioms and Theorems

$$\mathsf{irreflexive}(rel) \quad \rightarrow \quad \neg(\exists x[rel(x,x)])$$

$$\mathsf{irreflexive}(rel) \quad \rightarrow \quad \neg(\exists x,t[rel(x,x,t)])$$

$$\begin{aligned}\mathsf{proper\_subrelation}(pr,r) \quad \rightarrow \quad & \mathsf{irreflexive}(pr) \wedge \\ & \mathsf{subrelation}(pr,r)\end{aligned}$$

2.26 transitive

If R is transitive, then we can infer a R c from a R b and b R c. If R is time-indexed, then we can infer a R c @t from a R b @t and b R c @t

2.26.1 Axioms and Theorems

Axiom: transitivity of atemporal relations

$$\begin{aligned}&\mathsf{transitive}(rel) \wedge \\ &\quad rel(X,Y) \wedge \\ &\qquad rel(Y,Z) \quad \rightarrow \quad rel(X,Z)\end{aligned}$$

Axiom: transitivity of time-indexed relations. The 3rd argument must match to make the inference

$$\begin{aligned}&\mathsf{transitive}(rel) \wedge \\ &\quad rel(x,y,t) \wedge \\ &\qquad rel(y,z,t) \quad \rightarrow \quad rel(x,z,t)\end{aligned}$$

2.27 symmetric

2.27.1 Axioms and Theorems

Axiom: symmetricality implies cyclicity

$$\mathsf{symmetric}(rel) \quad \rightarrow \quad \mathsf{cyclic}(rel)$$

$$\text{symmetric}(rel) \wedge rel(i1, i2) \rightarrow rel(i2, i1)$$

$$\text{symmetric}(rel) \wedge rel(i1, i2, t) \rightarrow rel(i2, i1, t)$$

2.28 antisymmetric

a binary relation R is antisymmetric if, for all a and b, if a is R to b and b is R to a, then a = b.

2.28.1 Axioms and Theorems

$$\text{antisymmetric}(rel) \leftrightarrow rel(U1, U2) \wedge rel(U2, U1) \rightarrow \text{equivalent\_to}(U1, U2)$$

$$\text{antisymmetric}(rel) \leftrightarrow rel(i1, i2, t) \wedge rel(i2, i1, t) \rightarrow \text{equivalent\_to}(i1, i2)$$

2.29 cyclic

A cyclic relation is one which holds bidirectionally between two non-identical entities

The definition of cyclicity involves two entities. Note that for transitive relations longer chains are accounted for by transitivity axioms.

2.29.1 Axioms and Theorems

Axiom: cylic definition

$$\text{cyclic}(rel) \leftrightarrow \exists X, Y[rel(X, Y) \wedge rel(Y, X) \wedge \neg(\text{equivalent\_to}(X, Y))]$$

$$\text{acyclic}(rel) \rightarrow \neg(\text{cyclic}(rel))$$

Axiom: symmetricality implies cyclicity

$$\text{symmetric}(rel) \rightarrow \text{cyclic}(rel)$$

2.30 directed_path_over

S directed_path_over R iff for all a_1,a_n it is the case that S(a_1,a_n) implies a chain R(a_1,a_2),R(a_2,a_3),...,R(a_n-1,a_n). Vertices may be visisted more than once. If the chain includes R(x,y) then it may not contain R(y,x), even if R is symmetric - the path is directed.

note that this is not the same as the transitive version of the relation. S(x,x) only holds for cyclic structures.

2.30.1 Axioms and Theorems

2.31 directed_simple_path_over

S directed_simple_path_over R iff for all a_1,a_n it is the case that S(a_1,a_n) implies a chain R(a_1,a_2),R(a_2,a_3),...,R(a_n-1,a_n). Vertices may be not be visisted more than once, with the exception of the start vertex.

2.31.1 Axioms and Theorems

2.32 cyclic_over

S cyclic_over R iff there is a simple directed path over R starting and ending at v1 over vertices in V, then S holds between all pairs in V

2.32.1 Examples

- A ring of 6 carbon instances connected c1-c2,...,c6-c1. Each is connected_in_cycle_with all the others :

  $$\mathsf{cyclic\_over}(\mathsf{connected\_in\_cycle\_with}, \mathsf{directly\_connected\_to})$$

2.32.2 Axioms and Theorems

2.33 acyclic

An acyclic relation is a relation for which the cylicity property does not hold.

2.33.1 Axioms and Theorems

$$\mathsf{acyclic}(rel) \quad \rightarrow \quad \neg(\mathsf{cyclic}(rel))$$

2.34 proper_subrelation

An irreflexive subrelation predicate

2.34.1 Axioms and Theorems

$$\mathsf{proper\_subrelation}(pr, r) \rightarrow \mathsf{irreflexive}(pr) \wedge \mathsf{subrelation}(pr, r)$$

$$\mathsf{proper\_subrelation}(r1, r2) \wedge r1(x, y) \wedge \neg(x = y) \leftrightarrow r2(x, y)$$

2.35 transitive_over

R is transitive_over S if R and S compose to R. i.e. a R b S c implies a R c

2.35.1 Examples

- GO :

$$\mathsf{transitive\_over}(\mathsf{regulates}, \mathsf{part\_of})$$

2.35.2 Axioms and Theorems

- transitive

Axiom: transitive_over for atemporal relations

$$\mathsf{transitive\_over}(rel, over) \rightarrow rel(i1, i2) \wedge over(i2, i3) \rightarrow rel(i1, i3)$$

Axiom: transitive_over for time-indexed relations

$$\mathsf{transitive\_over}(rel, over) \rightarrow rel(i1, i2, t) \wedge over(i2, i3, t) \rightarrow rel(i1, i3, t)$$

2.36 holds_over_chain

R holds_over_chain R1 R2 iff R1 and R2 compose together to make R. i.e. a R1 b R2 c implies a R c

2.36.1 Examples

- PATO :

$$\mathsf{holds\_over\_chain}(\mathsf{inheres\_in\_part\_of}, \mathsf{inheres\_in}, \mathsf{part\_of})$$

- ZFA :

$$\mathsf{holds\_over\_chain}(\mathsf{starts\_during\_or\_after}, \mathsf{part\_of}, \mathsf{starts\_during})$$

2.36.2 Axioms and Theorems

Axiom: holds_over_chain for atemporal relations

$$\mathsf{holds\_over\_chain}(rel, r1, \mathsf{r2}) \quad \rightarrow \quad r1(i1, i2) \wedge r2(i2, i3) \rightarrow rel(i1, i3)$$

Axiom: holds_over_chain for time-indexed relations

$$\mathsf{holds\_over\_chain}(rel, r1, \mathsf{r2}) \quad \rightarrow \quad r1(i1, i2, t) \wedge r2(i2, i3, t) \rightarrow rel(i1, i3, t)$$

2.37 disjoint_over

R is disjoint_over S if R holds between entities that are not S-siblings.

2.37.1 Examples

- If X is spatially disconnected from Y, then there are no Z such that Z part_of_some X and Z part_of_some Y. :

$$\mathsf{disjoint\_over}(\mathsf{spatially\_disconnected\_from}, \mathsf{part\_of\_some})$$

2.37.2 Axioms and Theorems

$$\mathsf{disjoint\_over}(r, over) \quad \rightarrow \quad r(a, b) \rightarrow \neg(\exists x[over(x, a) \wedge over(x, b)])$$

$$\mathsf{disjoint\_over}(r, over) \quad \rightarrow \quad r(a, b) \rightarrow \neg(\exists x[over(x, a, t) \wedge over(x, b, t)])$$

2.38 maximal_over

R is maximal_over S iff when R(a,x,y) holds, it is the case that for all b [b S a implies b S x and b S y].

2.38.1 Examples

- If A maximally_overlaps B and C, all the parts overlapping A also overlap B and C. :

$$\text{maximally\_overlaps}(\text{maximally\_overlaps}, \text{overlaps})$$

2.38.2 Axioms and Theorems

$$\text{maximal\_over}(r, over) \rightarrow r(a, x, y) \rightarrow \text{forall}(b, over(b, a), =¿(over(b, x) \wedge over(b, y)))$$

2.39 disjoint_from

A is disjoint_from B if there is nothing that is an instance_of both A and B (at any one time)

2.39.1 Examples

- FBbt :

$$\text{disjoint\_from}(\text{embryo}, \text{larva})$$

2.39.2 Axioms and Theorems

- symmetric

Axiom:

$$\text{disjoint\_from}(a, b) \leftrightarrow \neg(\exists x[\text{instance\_of}(x, a) \wedge \text{instance\_of}(x, b)]) \wedge \neg(\exists x[\text{instance\_of}(x, a, t) \wedge \text{instance\_of}(x, b, t)])$$

Theorem: disjoint types do not share is_a children

$$\mathsf{disjoint\_from}(a,b) \rightarrow \neg(\exists x[\mathsf{is\_a}(x,a,t) \wedge \mathsf{is\_a}(x,b,t)])$$

2.40 relation_arity

2.40.1 Axioms and Theorems

2.41 relation_min_arity

2.41.1 Axioms and Theorems

2.42 relation_max_arity

2.42.1 Axioms and Theorems

2.43 posits

2.43.1 Axioms and Theorems

2.44 namespace

Relation between an identifier and a namespace. Each identifier belongs to only one namespace. Labels are unique within namespace. The unique identifier assumption holds within a namespace unless otherwise stated.

Namespaces can also be thought of as ontologies.

2.44.1 Axioms and Theorems

- functional

Axiom: identity implies identity of labels if Unique Label Assumptions holds within an ontology.

$$\begin{aligned} &\mathsf{namespace}(a,x) \wedge \\ &\mathsf{namespace}(b,x) \wedge \\ &\mathsf{identifier}(a,an) \wedge \\ &\mathsf{identifier}(b,bn) \wedge \\ &\mathsf{unique\_identifier\_assumption}(x) \wedge \\ &\neg(a=b) \rightarrow \neg(an=bn) \end{aligned}$$

Axiom: identity implies identity of identifiers if Unique Label Assumptions holds within an ontology.

$$
\begin{aligned}
\mathsf{namespace}(a, x) \;\wedge & \\
\mathsf{namespace}(b, x) \;\wedge & \\
\mathsf{label}(a, an) \;\wedge & \\
\mathsf{label}(b, bn) \;\wedge & \\
\mathsf{unique\_label\_assumption}(x) \;\wedge & \\
\neg(a = b) & \quad \rightarrow \quad \neg(an = bn)
\end{aligned}
$$

3 Obolog Annotations

Note: here *annotation* means something different than in OWL (although there are similarities).

In Obolog, annotations are contextually true sentences, typically supported by various kinds of provenance and metadata.

Annotation sentences are reified using the function `that`

3.1 Obolog semantics

The function `that` is undefined, so annotations have no entailments.

Optionally, this function can be translated to unreified form so that the sentence is globally true. This is non-normative.

3.2 Obolog-A semantics

Obolog-A is defined using the KR language IKL, an extension of Common Logic. The semantics of `that` are as for IKL.

4 OBO Syntax

Ian Horrocks transcribed a grammar for OBO Format 1.2. Syntax. This section borrows heavily from this work.

4.1 Relation to OBO-Format 1.2

OBO-Format 1.3 is forwards and backwards compatible with OBO-Format 1.2.

TODO: List of new features.

4.2 OBO Lexical Rules

```
OBO-Doc := header {nl}  { stanza }

header := { tagval-line }
stanza := '[' word ']' nl tagval-line { tagval-line } nl {nl}
tagval-line :=
 str-tag ':' {white} unquoted-text {white} [line-comment] nl |
 normal-tag ':' {white} obo-tokenstream [{white} xrefs] [ {white} mods] [ {white}

unquoted-text := { ( char - ('!' | nl-char)) }

str-tag := 'name' | 'comment'

normal-tag := word

word := word-token { word-token }
word-token := alphanumeric | '_' | '-'

nl := {white} [ bang-comment ] {white} nl-char
nl-char := ('\n' | '\r')
bang-comment := '!' {(char - nl-char)}

obo-tokenstream := obo-token { obo-token }
obo-token := quoted-string | bare-token

bare-token := token-char { token-char }
token-char := esc-char | char - (white | nl | '{' | '}' | '[' | ']')
```

```
esc-char := '\' char

xrefs := '[' {white} xref { {white} ',' xref } {white} ']'
mods := '{' {white} mod { {white} ',' mod } {white} '}'

line-comment := '!' { char - nl-char }
```

4.3 OBO Document Structure

An OBO file consists of a header followed by zero or more stanzas.

```
OBO-Doc := header { stanza }
stanza := term-stanza | typedef-stanza | instance-stanza | annotation-stanza | for
```

Each stanza is associated with a single namespace. Note that a single ID-space such as GO can have IDs dividied across multiple namespaces.

4.4 OBO Header

The header consists of a number of tag-value pairs, most of which we will ignore for the time being. Many of these (e.g., ¡remark¿ could clearly be treated as annotations; others (e.g., ¡default-namespace¿ correspond to parts of an XML document preamble.

```
header :=
   <format-version>
   [ <data-version> ]
   [ <date> ]
   [ <saved-by> ]
   [ <auto-generated-by> ]
   [ <subsetdef> ]
   { import }
   { <synonymtypedef> }
   { <idspace> }
   [ <default-namespace> ]
   [ <default-relationship-id> ]
   { <idmapping> }
   { expansion-macro }
   [ <remark> ]
```

```
import := 'import:' <URL>

expansion-macro :=
  'relax-unique-name-assumption-for-idspace:' idspace |
  'treat-xrefs-as-equivalent:' idspace |
  'treat-xrefs-as-is_a:' idspace |
  'treat-xrefs-as-inverted-is_a:' idspace |
  'treat-xrefs-as-genus-differentia:' idspace relation-id type-id |
  'treat-xrefs-as-relationship:' idspace relation-id
  'treat-xrefs-as-unique:' idspace
```

The default-namespace specifies the default namespace for all stanzas in that file. A namespace that is stated using the namespace tag always takes precedence over the default-namespace tag.

Behavior is undefined if neither stanza namespace nor header default-namespace is specified, it is recommended the parser assign a default namespace equivalent to the path or URL from which the ontology was downloaded.

4.5 Term Stanzas

Term stanzas introduce and define the meaning of types (AKA terms, concepts, classes and unary predicates).

Currently the grammar enforces an ordering of tags. This should be changed. Tag ordering is recommended for generation, but optional for parsing. It is strongly recommended that tags are grouped (i.e. tags of the same type should not be separated by other tags).

```
term-stanza :=
   '[Term]'
   typeid-TVP
   name-TVP
   [ namespace-TVP ]
   { altid-TVP }
   [ def-TVP ]
   [ comment-TVP ]
   { <subset> }
   { <synonym> }
```

```
   { xref-TVP }
   { isa-TVP }
   { intersection-TVP }
   { union-TVP }
   { disjoint-TVP }
   { relationship-TVP }
   { equiv-TVP }
   { p-obj-value-TVP | p-data-value-TVP }
   [ is-obsolete-TVP ]
   [ is-anonymous-TVP ]
   [ <replaced_by> ]
   { <consider> }
   { formula-TVP }

typeid-TVP :=
   'id:' type-id
   [ 'is_anonymous: true' ]
type-id := <string>

name-TVP :=
  'name:'<string>

comment-TVP :=
  'name:'<string>

namespace-TVP :=
  'namespace:' id

xref-TVP :=
  ( 'xref:' | 'xref_analog:' ) id

def-TVP :=
  'def:' quoted-string xrefs

altid-TVP :=
  'alt_id:' id

isa-TVP :=
   'is_a:' type-id
   [ 'namespace=' <namespace-id> ]
```

```
   [ 'derived=true' | 'derived=false' ]

intersection-TVP :=
   'intersection_of:' termOrRestr
   [ 'namespace=' <namespace-id> ]
termOrRestr := type-id | restriction
restriction := relationship-id type-id
relationship-id := <string>

union-TVP :=
   'union_of:' termOrRestr
   [ 'namespace=' <namespace-id> ]

disjoint-TVP :=
   'disjoint_from:' type-id
   [ 'namespace=' <namespace-id> ]
   [ 'derived=true' | 'derived=false' ]

relationship-TVP :=
   'relationship:' restriction
   [ 'not_necessary=true' | 'not_necessary=false' ]
   [ 'inverse_necessary=true' | 'inverse_necessary=false' ]
   [ 'cardinality=' <non-neg-int> ]
   [ 'maxCardinality=' <non-neg-int> ]
   [ 'minCardinality=' <non-neg-int> ]

equiv-TVP :=
   'equivalent_to:' type-id

formula-TVP :=
   'formula:' [formula-syntax] formula-str {xref}

formula-syntax :=
   'KIF' | 'CLIF' | 'XCL' | 'Prover9'
```

4.6 Typedef Stanzas

Typedef stanzas introduce and define the meaning of relations (AKA roles, properties and predicates).

```
typedef-stanza :=
   '[Typedef]'
   typedef-TVP
   'name:'<string>
   [ <namespace> ]
   { <alt_id> }
   [ <def> ]
   [ <comment> ]
   { <subset> }
   { <synonym> }
   { <xref> }
   [ domain-TVP ]
   [ range-TVP ]
   { unary-property-TVP }
   { r-isa-TVP }
   { r-intersection-TVP }
   [ inverse-TVP ]
   [ inst-inverse-TVP ]
   [ transover-TVP ]
   { holdsover-TVP }
   { equivchain-TVP }
   { disjover-TVP }
   { r-relationship-TVP }
   { r-equiv-TVP }
   { p-obj-value-TVP | p-data-value-TVP }
   [ is-obsolete-TVP ]
   [ <replaced_by> ]
   { <consider> }
   { formula-TVP }

typedefid-TVP :=
   'id:' relationship-id
   [ 'is_anonymous: true' ]

domain-TVP := 'domain:' termOrReserved
termOrReserved := type-id | <reserved-id>
```

```
range-TVP := 'range:' termOrReserved

unary-property-TVP :=
    unary-property ':' ( 'true' | 'false' )
unary-property :=
   'is_anti_symmetric' |
   'is_cyclic' |
   'is_reflexive' |
   'is_irreflexive' |
   'is_symmetric' |
   'is_transitive' |
   'is_functional' |
   'is_inverse_functional' |
   'is_metadata_tag'

is-obsolete-TVP := 'is_obsolete:' ( 'true' | 'false' )

r-isa-TVP := 'is_a:' relationship-id [ isa-mlist ]
isa-mlist := '{' isa-modifier { ',' isa-modifier } '}'
isa-modifier := namespace-mod | derived-mod
namespace-mod := 'namespace=' namespace-id
derived-mod := 'derived=true' | 'derived=false'

r-intersection-TVP := 'intersection_of:' relationship-id

inverse-TVP := 'inverse_of:' relationship-id
inst-inverse-TVP := 'inverse_of_on_instance_level:' relationship-id

transover-TVP := 'transitive_over:' relationship-id

holdsover-TVP := 'holds_over_chain:' relationship-id relationship-id

equivchain-TVP := 'equivalent_to_chain:' relationship-id relationship-id

disjover-TVP := 'disjoint_over:' relationship-id relationship-id

r-relationship-TVP :=
   'relationship:' r-relationship-type relationship-id
```

```
r-relationship-type :=
 'all_some_all_times' |
 'all_some' |
 'all_some_all_tr' |
 'all_only_all_times' |
 'all_only' |

r-equiv-TVP :=
   'equivalent_to:' type-id
```

4.7 Instance Stanzas

Instance stanzas introduce and define the meaning of instances (AKA individuals, individual names, constants).

```
instance-stanza :=
   '[Instance]'
   instanceid-TVP
   'name:'<string>
   [ <namespace>]
   { <alt_id> }
   [ <comment> ]
   { <synonym> }
   { <xref> }
   'instance_of:' type-id { 'instance_of:' type-id }
   { i-relationship-TVP }
   { p-obj-value-TVP | p-data-value-TVP }
   [ is-obsolete-TVP ]
   [ <replaced_by> ]
   { <consider> }

instanceid-TVP :=
   'id:' instance-id
   [ 'is_anonymous: true' ]

p-obj-value-TVP := 'property_value:' relationship-id instance-id

p-data-value-TVP := 'property_value:' relationship-id '"' <string> '"' <XML-Schema
```

4.8 Annotation Stanzas

Annotation stanzas introduce and define the meaning of annotations. In OBO an annotation is a statement assumed to be true in some non-global context.

```
annotation-stanza :=
   '[Annotation]'
   [annotationid-TVP]
   [ <namespace>]
   { <alt_id> }
   [ <comment> ]
   { <synonym> }
   { <xref> }
   subject-TVP
   [ is_negated-TVP ]
   relation-TVP
   object-TVP
   [ description-TVP ]
   [ assigned_by-TVP ]
   [ source-TVP ]
   [ context-TVP ]
   { evidence-TVP }
   {'instance_of:' type-id }
   { p-obj-value-TVP | p-data-value-TVP }
   [ <is_obsolete> ]
   [ <replaced_by> ]
   { <consider> }

annotationid-TVP :=
   'id:' annotation-id

subject-TVP :=
   'subject:' entity-id

is_negated-TVP :=
   'is_negated:true' | 'is_negated:false'

relation-TVP :=
   relation:' relation-id
```

```
object-TVP :=
   'object:' object-id

description-TVP :=
   'description:' desc-str

assigned_by-TVP :=
   'assigned_by:' inst-id

source-TVP :=
   'source:' inst-id

context-TVP :=
   'context:' context-id

evidence-TVP :=
   'evidence:' entity-id
```

4.9 Formula Stanzas

Formula stanzas introduce logical formulae.

```
formula-stanza :=
   '[Formula]'
   [formulaid-TVP]
   [ <namespace>]
   { <alt_id> }
   [ <comment> ]
   { <synonym> }
   { <xref> }
   [ syntax-TVP ]
   [ description-TVP ]
   [ <is_obsolete> ]
   [ <replaced_by> ]
   { <consider> }
   formula-body-TVP

formulaid-TVP :=
```

```
   'id:' formula-id

syntax-TVP :=
   'syntax:' formula-syntax

formula-body-TVP :=
   'body:' formula-str
```

4.10 ID Macros

4.10.1 ID Expressions

```
idref := id-expr
idref := identifier
id-expr := genus-id '^' differentium { '^' differentium }
differentium := rel-id '(' idref ')
genus-id := idref
```

Any time an identifier matches the above pattern, it is auto-expanded to a stanza:

```
[Term]
id: <id-expr>
intersection_of: <genus-id>
intersection_of: <rel-id1> <idref-1>
 .
 .
intersection_of: <rel-id-n> <idref-n>
```

4.10.2 Unique ID and Label Assumption

The local unique ID and label assumption is assumed true for a namespace unless explicitly declared false.

```
'unique-id-assumption:' namespace-id ( 'true' | 'false' )
'unique-label-assumption:' namespace-id ( 'true' | 'false' )
```

4.10.3 Xref Expansion

4.11 Additional considerations

Header	Xref	Expansion
treat-xrefs-as-equivalent: IDSpace	xref: IDSpace:LocalID	equivalent_to: IDSpace:Local
treat-xrefs-as-genus-differentia: IDSpace Rel Filler	xref: IDSpace:LocalID	intersection_of: IDSpace:Lo intersection_of: Rel Filler
treat-xref-as-relationship: IDSpace Rel	xref: IDSpace:LocalID	relationship: Rel IDSpace:L
treat-xref-as-is_a: IDSpace	xref: IDSpace:LocalID	is_a: IDSpace:LocalID
treat-xref-as-inverted-is_a: IDSpace	id: ID xref: IDSpace:LocalID	id: IDSpace:LocalID is_a: ID

5 OBO Semantics

5.1 Mapping to Obolog

The translation is defined using a translation function T which translates (a fragment of) OBO into Obolog. The definition of T is often recursive, but it will eventually "ground out" in Obolog. We use functional syntax to specify the Obolog sentence. This could equally be specified in CLIF.

5.2 Identifiers

If an id: tag is not specified in a stanza, an anonymous gensym ID is created. Currently the id tag is required for all stanzas, with the exception of Annotation stanzas.

5.3 Imports

TODO: CL construct?

TODO: selective imports?

5.4 Translation Table

OBO Syntax -S	Translation - T(S)
`header stanza-1 ... stanza-n`	`T(header) T(stanza-1) ... T(stanza-n)`

`[Typedef]`
` id: type-id`
` tagvals-1`
` ..`
` tagvals-n`
	`relation(rel-id)`
	`T(Relation id:rel-id tagvals-1)`
	`.`
	`.`
	`T(Relation id:rel-id tagvals-n)`

`[Term]`
` id: type-id`
` tagvals-1`
` ..`
` tagvals-n`
	`type(type-id)`
	`T(Type id:type-id tagvals-1)`
	`.`
	`.`
	`T(Type id:type-id tagvals-n)`

[Instance] id: type-id tagvals-1 .. tagvals-n	type(type-id) T(Instance id:type-id tagvals-1) . . T(Instance id:type-id tagvals-n)
[Annotation] id: annot-id relation: rel subject: subj-id object: obj-id argument: 3 arg-3 . . argument: n arg-n tagvals-1 .. tagvals-n	annotation(annot-id) posits(annot-id (that subj-id obj-id arg-3 .. arg-n)) T(Annotation id:annot-id tagvals-1) . . T(Instance id:type-id tagvals-n)
Type id:type-id is_a: is_a-1 .. is_a: is_a-n	is_a(type-id is_a-1) . . is_a(type-id is_a-n)
Relation id:rel-id is_a: is_a-1 .. is_a: is_a-n	subrelation(rel-id is_a-1) . . subrelation(rel-id is_a-n)
_ id:entity-id relationship: rel-1 arg-1-1 .. arg-1-m .. relationship: rel-n arg-n-1 .. arg-1-m	rel-1(entity-id, arg-1-1, ... arg-1-m) . . rel-n(entity-id, arg-n-1, ... arg-n-m)
Type id:type-id intersection_of: args-1 .. intersection_of: args-n	equivalent_to(type-id intersection_of(T(intersection_element:args-1) . . T(intersection_element:args-n)))

intersection_element: type-id	type-id
intersection_element: rel arg-1 .. arg-n	rel(arg-1 .. arg-n)
Relation id:rel-id intersection_of: rel-1 .. intersection_of: rel-n	equivalent_to(rel-id relation_intrsection(rel-1 .. rel-n))
Type id:type-id disjoint_from: disj-1 . . disjoint_from: disj-n	disjoint_from(type-id disj-1) . . disjoint_from(type-id disj-n)
Relation id:rel-id disjoint_over: over-1 . . disjoint_over: over-n	disjoint_over(rel-id over-1) . . disjoint_over(rel-id over-n)
Relation id:rel-id transitive_over: over-1 . . transitive_over: over-n	transitive_over(rel-id over-1) . . transitive_over(rel-id over-n)
Relation id:rel-id holds_over_chain: x-1 y-1 . . holds_over_chain: x-n y-n	holds_over_chain(rel-id x-1 y-1) . . holds_over_chain(rel-id x-n y-n)
Relation id:rel-id equivalent_to_chain: x-1 y-1 . . equivalent_to_chain: x-n y-n	equivalent_to_chain(rel-id x-1 y-1) . . equivalent_to_chain(rel-id x-n y-n)

Relation id:relation-id inverse_of: inv-1 . . inverse_of: inv-n	inverse_of(relation-id inv-1) . . inverse_of(relation-id inv-n)
Relation id:relation-id inverse_of_on_instance_level: inv-1 . . inverse_of_on_instance_level: inv-n	inverse_of_on_instance_level(relation-id inv-1) . . inverse_of_on_instance_level(relation-id inv-n)
_ id:entity-id name: name-str	label(entity-id name-str)
_ id:entity-id is_anonymous: true	anonymous(entity-id)
_ id:entity-id namespace: namespace-str	namespace(entity-id namespace-str)
_ id:entity-id alt_id: alt-ref-1 . . alt_id: alt-ref-n	alternate_id(entity-id alt-ref-1) . . alternate_id(entity-id alt-ref-n)
Type id:type-id def: def-str def-xref-1 .. def-xref-n	text_definition(type-id def-str) has_source((that text_definition(type-id def-str)) def-xref-1) . . has_source((that text_definition(type-id def-str)) def-xref-n)

`_ id:entity-id`
`synonym: syn-details-1`
`.`
`.`
`synonym: syn-details-n`
	`T(synonym entity-id syn-details-1)`
	`.`
	`.`
	`T(synonym entity-id syn-details-n)`

`synonym entity-id`
`syn-str-n scope-enum type`
`xref-n-1 .. xref-n-m`
	`T(scope: scope-enum)(entity-id syn-str-1 type)`
	`has_source(`
	`(that`
	`T(scope: scope-enum)(entity-id`
	`syn-str-1 type))`
	`def-xref-1)`
	`.`
	`.`
	`has_source(`
	`(that`
	`T(scope: scope-enum)(entity-id`
	`syn-str-1 type))`
	`def-xref-n)`

`synonym entity-id`
`syn-str-n scope-enum`
`xref-n-1 .. xref-n-m`
	`T(scope: scope-enum)(entity-id syn-str-1)`
	`has_source(`
	`(that`
	`T(scope: scope-enum)(entity-id`
	`syn-str-1))`
	`def-xref-1)`
	`.`
	`.`
	`has_source(`
	`(that`
	`T(scope: scope-enum)(entity-id`
	`syn-str-1))`
	`def-xref-n)`

`scope: EXACT`
`scope: RELATED`
`scope: BROAD`
`scope: NARROW`
	`exact_synonym`
	`related_synonym`
	`broad_synonym`
	`narrow_synonym`

`id: genus ^ differentium-1 .. differentium-n`	`todo`

Table 1: Translation to Obolog

6 Obolog Sublanguages and Superlanguages

This section is incomplete

6.1 CL

CL is the set of all common logic texts

6.2 IKL

IKL is the set of all IKL texts. Every CL text is an IKL text.
$\mathsf{CL} \subset IKL$

6.3 obolog^{CL}

Any CL text that imports the Obolog predicate definitions is an obolog^{CL} text
$\mathsf{obolog}^{CL} \subset \mathsf{CL}$

6.4 obolog^{Core}

Any obolog^{CL} text that does not use quantified sentences (excluding the CL axioms for the Obolog predicates themselves).

$$\mathsf{obolog}^{Core} \subset \mathsf{obolog}^{CL}$$

The suffix "Core" denotes a subset that excludes arbitrary quantified sentences.

6.5 obolog^{A}

A subset of IKL in which the that function is used for atomic sentences only.

Every obolog^{A} text is an IKL text:

$$\mathsf{obolog}^{A} \subset \mathsf{IKL}$$

Every obolog^{CL} text is an obolog^{A} text

$$\mathsf{obolog}^{CL} \subset \mathsf{obolog}^{A}$$

Again we can define $\mathsf{obolog}^{A/Core}$ as the subset of obolog^{A}that excluded quantified sentences.

$$\mathsf{obolog}^{A/Core} = \mathsf{obolog}^{A} \cap \mathsf{obolog}^{Core}$$

6.6 obolog^{H} and obolog^{Data}

6.6.1 obolog^{H}

Horn clause subset of $\mathsf{obolog}^{CL}\dot{\mathrm{I}}$t extends obolog^{Core} by allowing certain quantified sentences, specifically those correponding to horn rules.

$$\mathsf{obolog}^{H} \subset \mathsf{obolog}^{CL}$$

6.6.2 obolog^{Data}

$$\mathsf{obolog}^{Data} \subset \mathsf{obolog}^{H}$$

Datalog subset of obolog^{CL} i.e. only atomic sentences, or universally quantified sentences with one term on the LHS, and no function symbols. Query evaluation with Datalog is sound and complete and can be done efficiently even for large texts (databases).

6.6.3 Horn Rules

either restatements of obolog axioms as horn rules, or rules derived from axioms + RO definitionsRule: reflexivity of is_a

$$\mathsf{type}(x) \quad \rightarrow \quad \mathsf{is\_a}(x,x)$$

Rule: transitivity of is_a

$$\begin{aligned} &\mathsf{is\_a}(a,b) \wedge \\ &\quad \mathsf{is\_a}(b,c) \quad \rightarrow \quad \mathsf{is\_a}(a,c) \end{aligned}$$

Rule: equivalence if mutual is_a

$$\begin{aligned} &\mathsf{is\_a}(a,b) \wedge \\ &\quad \mathsf{is\_a}(b,a) \quad \rightarrow \quad \mathsf{equiavelent\_to}(a,b) \end{aligned}$$

Rule: equivalence if mutual is_a, inv

$$\mathsf{equiavelent\_to}(a,b) \quad \rightarrow \quad \mathsf{is\_a}(a,b)$$

Rule: transitivity of subrelation

$$\begin{aligned} \mathsf{subrelation}(a,b) \wedge \\ \mathsf{subrelation}(b,c) \quad &\rightarrow \quad \mathsf{subrelation}(a,c) \end{aligned}$$

XRule: propagation over/under is_a for all-some relations

$$\begin{aligned} \mathsf{is\_a}(a,b) \wedge \\ r(b,c) \wedge \\ \mathsf{is\_a}(c,d) \wedge \\ \exists ri[\mathsf{all\_some}(r,ri)] \quad &\rightarrow \quad r(a,d) \end{aligned}$$

XRule: propagation over/under is_a for all-only relations

$$\begin{aligned} \mathsf{is\_a}(a,b) \wedge \\ r(b,c) \wedge \\ \mathsf{is\_a}(c,d) \wedge \\ \exists ri[\mathsf{all\_only}(r,ri)] \quad &\rightarrow \quad r(a,d) \end{aligned}$$

XRule: all-only constraints on subtypes

$$\begin{aligned} \mathsf{all\_only}(ru,ri) \wedge \\ \mathsf{all\_some}(re,ri) \wedge \\ ru(b,y) \wedge \\ re(a,x) \wedge \\ \mathsf{is\_a}(a,b) \quad &\rightarrow \quad \mathsf{is\_a}(x,y) \end{aligned}$$

XRule: transitivity

$$\begin{aligned} \mathsf{transitive}(r) \wedge \\ r(a,b) \wedge \\ r(c,c) \quad &\rightarrow \quad r(a,c) \end{aligned}$$

XRule: inverse_of

$$
\begin{aligned}
\mathsf{inverse\_of}(r, s) \wedge & \\
r(a, b) &\quad \rightarrow \quad r(b, a)
\end{aligned}
$$

Rule: symmetricality of inverse_of

$$
\mathsf{inverse\_of}(r, s) \quad \rightarrow \quad \mathsf{inverse\_of}(s, r)
$$

Rule: symmetricality of disjoint_from

$$
\mathsf{disjoint\_from}(r, s) \quad \rightarrow \quad \mathsf{disjoint\_from}(s, r)
$$

XRule: subrelations

$$
\begin{aligned}
r(a, b) \wedge & \\
\mathsf{subrelation}(r, s) &\quad \rightarrow \quad s(a, b)
\end{aligned}
$$

XRule: transitive_over

$$
\begin{aligned}
\mathsf{transitive\_over}(r, over) \wedge & \\
r(a, b) \wedge & \\
over(b, c) &\quad \rightarrow \quad r(a, c)
\end{aligned}
$$

XRule: holds_over_chain

$$
\begin{aligned}
\mathsf{holds\_over\_chain}(r, r1, r2) \wedge & \\
r1(a, b) \wedge & \\
r2(b, c) &\quad \rightarrow \quad r(a, c)
\end{aligned}
$$

XRule: cyclic

$$
\begin{aligned}
r(a, b) \wedge & \\
r(b, a) \wedge & \\
\neg(a = b) &\quad \rightarrow \quad \mathsf{cyclic}(r)
\end{aligned}
$$

XRule: functional relations

$$\begin{aligned}\mathsf{functional}(r) \wedge \\ r(a,b) \wedge \\ r(a,c) \quad &\rightarrow \quad \mathsf{equivalent\_to}(b,c)\end{aligned}$$

XRule: symmetricality

$$\begin{aligned}\mathsf{symmetric}(r) \wedge \\ r(a,b) \quad &\rightarrow \quad r(b,a)\end{aligned}$$

XRule: domain

$$\begin{aligned}\mathsf{domain}(r,x) \wedge \\ r(a,b) \quad &\rightarrow \quad \mathsf{instance\_of}(a,x)\end{aligned}$$

XRule: range

$$\begin{aligned}\mathsf{range}(r,x) \wedge \\ r(a,b) \quad &\rightarrow \quad \mathsf{instance\_of}(b,x)\end{aligned}$$

$$\begin{aligned}\mathsf{all\_only}(ru,ri) \wedge \\ ru(X,Y) \wedge \\ ri(a,b) \wedge \\ \mathsf{instance\_of}(a,\mathsf{Occurrent}) \wedge \\ \mathsf{instance\_of}(b,\mathsf{Occurrent}) \wedge \\ \mathsf{instance\_of}(a,X) \quad &\rightarrow \quad \mathsf{instance\_of}(b,Y)\end{aligned}$$

Constraint: disjoint pairs share no is_a children

$$\begin{aligned}\mathsf{is\_a}(a,x) \wedge \\ \mathsf{is\_a}(b,x) \wedge \\ \mathsf{disjoint\_from}(a,b) \quad &\rightarrow \quad \mathsf{unsatisfiable}(a)\end{aligned}$$

7 Translating Obolog to OWL

We provide 3 interpretations for the logical semantics of Obolog.

- A standard/normative OWL-DL translation, which makes use of "hidden" time-slices to get around the lack of n-ary relations in OWL
- A simplied OWL-DL translation
- An OWL-Full (RDFS) translation, in which types (classes) and type level relations are in the domain of discourse

The normative translation is non-trivial. It attempts to do justice to the use of 3-ary relations in defining binary type-level relations. It is predicated on the widely accepted assumption that Continuants (entities exist in whole in any point in time, but can gain or lose parts through time) are difficult to deal with in OWL due to all relations (properties) being binary.

The translation works within this constraint, and provides an interpretation of BFO/OWL in which references to continuant types are translated as references to corresponding time-slices.

In addition, we provide an extension translation for obolog-A, and a translation for obolog-lex in terms of OWL Annotation Properties.

7.1 Standard DL Translation

Abstract

Mapping to OWL, tackles n-ary arguments. This mapping preserves the meaning of continuants yets avoids unneccessary reference to SpatioTemporalRegions in source OWL ontologies, by re-interpreting the owl membership/type relation in the context of continuants

7.1.1 Functions

Functions used in the OWL-DL Translation

7.1.2 Function: time_slice

Take a continuant type as argument and returns the unique type that represents the corresponding SpatioTemporalRegion.

Example:

$$\mathsf{time\_slice(Heart) = HeartSpatioTemporalRegion}$$

$$\mathsf{type\_domain(time\_slice, Continuant)}$$

$$\mathsf{type\_range}(\mathsf{time\_slice}, \mathsf{SpatioTemporalRegion})$$

Axiom: time_slice maps a continuant to a ST Region

$$\forall C, S[\mathsf{time\_slice}(C) = S \quad \rightarrow \quad \mathsf{is\_a}(S, \mathsf{SpatioTemporalRegion})]$$

7.1.3 Function: slice_of

Take an instance of a time slice and returns the instance of the corresponding continuant.

Example:

$$\mathsf{slice\_of}(\mathsf{john\_at\_2pm\_today}) = \mathsf{john}$$

$$\mathsf{domain}(\mathsf{slice\_of}, \mathsf{SpatioTemporalRegion})$$

$$\mathsf{range}(\mathsf{slice\_of}, \mathsf{Continuant})$$

7.1.4 Function: at

Take an instance of a continuant and a time, and return the instance of the corresponding time-slice.

Example:

$$\mathsf{at}(\mathsf{john}, \mathsf{2pm\_today}) = \mathsf{john\_at\_2pm\_today}$$

Axiom: inverse

$$\forall c, t[\mathsf{slice\_of}(\mathsf{at}(c, t)) = c]$$

7.1.5 Meta-Relation: relslice

relates a slice-relation to the corresponding continuant relation

OWL implicitly refers to relslices. However, there is no need to directly reference them in an OWL ontology.

Example:

$$\mathsf{relslice}(\mathsf{st\_located\_in}, \mathsf{located\_in})$$

$$\begin{aligned}\mathsf{relslice}(sr, r) \quad \rightarrow \quad & \mathsf{domain}(sr, \mathsf{Occurrent}) \wedge \\ & \mathsf{range}(sr, \mathsf{Occurrent}) \wedge \\ & \neg(\mathsf{domain}(r, \mathsf{Occurrent}) \wedge \\ & \mathsf{range}(r, \mathsf{Occurrent}))\end{aligned}$$

7.1.6 OWL Conversion Axioms

Instantiation and instance-level relations

7.1.7 Meta-Relation: owl:type

We use a predicate of owl:type rather that a unary predicate to avoid clashes between obolog and owl predicates

$$\mathsf{comment}(\mathsf{owl{:}type}(i, C) \quad \leftrightarrow \quad C(i))$$

7.1.8 Meta-Relation: owl:fact

We use a predicate of owl:fact rather than binary predicates to avoid clashes between obolog and owl predicates. Also note we make the property the 2nd argument, consistent with RDF

$$\mathsf{comment}(\mathsf{owl{:}fact}(i, r, j) \quad \leftrightarrow \quad r(i, j))$$

Axiom: OWL instantiation of Continuants.; i owl-type continuant C iff i instance_of time_slice(C).; owl talk of continuants is always interpreted as owl talk of time slices of continuants

$$\begin{aligned} \mathsf{owl{:}type}(i, C) \wedge \\ \mathsf{owl{:}type}(i, \mathsf{Continuant}) \quad \leftrightarrow \quad \mathsf{instance\_of}(i, \mathsf{time\_slice}(C)) \end{aligned}$$

Example:

$$\mathsf{owl{:}type}(\mathsf{heart1234\_at\_t1}, \mathsf{Heart}) \quad \leftrightarrow \quad \mathsf{instance\_of}(\mathsf{heart1234\_at\_t1}, \mathsf{time\_slice}(\mathsf{Heart}))$$

Axiom: re-interpret owl triples between slices as holding between; the slice version of the relation

$$\begin{aligned} \mathsf{owl{:}fact}(i, R, j) \wedge \\ \mathsf{instance\_of}(i, \mathsf{time\_slice}(A)) \wedge \\ \mathsf{instance\_of}(j, \mathsf{time\_slice}(B)) \wedge \\ \mathsf{relslice}(SR, R) \quad \rightarrow \quad SR(i, j) \end{aligned}$$

Example:

$$\begin{aligned}\mathsf{owl{:}type}(\mathsf{person1}, \mathsf{Human}) \wedge \\ \mathsf{owl{:}type}(\mathsf{heart1}, \mathsf{Heart}) \wedge \\ \mathsf{occurs\_at}(\mathsf{person1}, \mathsf{now}) \wedge \\ \mathsf{occurs\_at}(\mathsf{heart1}, \mathsf{now}) \wedge \\ \mathsf{relslice}(\mathsf{st\_located\_in}, \mathsf{located\_in}) \wedge \\ \mathsf{owl{:}fact}(\mathsf{heart1}, \mathsf{located\_in}, \mathsf{person1}) \quad &\rightarrow \quad \mathsf{st\_located\_in}(\mathsf{heart1}, \mathsf{person1})\end{aligned}$$

7.1.9 OWL Conversion Axioms

Type-level relations *Axiom: type-level all-some relation treated as existential restrictions; Can we prove this and make it a theorem?*

$$\begin{aligned}R(A, B) \wedge \\ \mathsf{all\_some\_all\_times}(R, RI) \quad &\rightarrow \quad \mathsf{owl{:}SubClassOf}(A, \mathsf{owl{:}someValuesFrom}(RI, B))\end{aligned}$$

Proof:; just restating obolog axiom...

$$\begin{aligned}R(A, B) \wedge \\ \mathsf{all\_some\_all\_times}(R, RI) \quad &\rightarrow \quad \mathsf{instance\_of}(i, A, t) \rightarrow \exists j[\mathsf{instance\_of}(j, B, t) \wedge \\ &\quad\ RI(i, j, t)]\end{aligned}$$

Proof:; ..with time slices. use 'at' function

$$\begin{aligned}R(A, B) \wedge \\ \mathsf{all\_some\_all\_times}(R, RI) \quad &\rightarrow \quad \mathsf{instance\_of}(\mathsf{at}(i, t), \mathsf{time\_slice}(A)) \rightarrow \exists j[\mathsf{instance\_of}(\mathsf{at}(j, t), \mathsf{time\_slic} \\ &\quad\ RI(i, j, t)]\end{aligned}$$

Proof:; ..transform relation

$$\begin{aligned}R(A, B) \wedge \\ \mathsf{all\_some\_all\_times}(R, RI) \quad &\rightarrow \quad \mathsf{instance\_of}(\mathsf{at}(i, t), \mathsf{time\_slice}(A)) \wedge \\ &\quad\ \mathsf{relslice}(SRI, RI) \rightarrow \exists j[\mathsf{instance\_of}(\mathsf{at}(j, t), \mathsf{time\_slice}(B)) \wedge \\ &\quad\ SRI(\mathsf{at}(i, t), \mathsf{at}(j, t))]\end{aligned}$$

Proof:; ..back to owl -convert to owl:type

$$\begin{aligned} R(A,B) \wedge \\ \mathsf{all\_some\_all\_times}(R,RI) &\rightarrow \mathsf{owl{:}type}(\mathsf{at}(i,t),A) \wedge \\ &\mathsf{relslice}(SRI,RI) \rightarrow \exists j[\mathsf{owl{:}type}(\mathsf{at}(j,t),B) \wedge \\ &SRI(\mathsf{at}(i,t),\mathsf{at}(j,t))] \end{aligned}$$

Proof:; ..back to owl - relation

$$\begin{aligned} R(A,B) \wedge \\ \mathsf{all\_some\_all\_times}(R,RI) &\rightarrow \mathsf{owl{:}type}(\mathsf{at}(i,t),A) \rightarrow \exists j[\mathsf{owl{:}type}(\mathsf{at}(j,t),B) \wedge \\ &\mathsf{owl{:}fact}(RI,\mathsf{at}(i,t),\mathsf{at}(j,t))] \end{aligned}$$

Proof:; we can replace functions with variables.; End result looks like an owl restriction axiom...

$$\begin{aligned} R(A,B) \wedge \\ \mathsf{all\_some\_all\_times}(R,RI) &\rightarrow \mathsf{owl{:}type}(x,A) \rightarrow \exists y[\mathsf{owl{:}type}(y,B) \wedge \\ &\mathsf{owl{:}fact}(RI,x,y)] \end{aligned}$$

Example:

$$\begin{aligned} \mathsf{located\_in\_some}(\mathsf{Brain},\mathsf{Skull}) \wedge \\ \mathsf{all\_some\_all\_times}(\mathsf{located\_in\_some},\mathsf{located\_in}) &\rightarrow \mathsf{owl{:}SubClassOf}(\mathsf{Brain},\mathsf{owl{:}someValuesFrom}(\mathsf{loca} \end{aligned}$$

Axiom: type-level all-only relation treated as universal restrictions; Can we prove this and make it a theorem?

$$\begin{aligned} R(A,B) \wedge \\ \mathsf{all\_only}(R,RI) &\rightarrow \mathsf{owl{:}SubClassOf}(A,\mathsf{owl{:}allValuesFrom}(RI,B)) \end{aligned}$$

Axiom: time-restricted type level relations

$$\begin{aligned} R(A,B) \wedge \\ \mathsf{all\_some\_tr}(R,RI) &\rightarrow \mathsf{owl{:}SubClassOf}(A,\mathsf{owl{:}allValuesFrom}(RI,B)) \end{aligned}$$

7.1.10 Axioms relating time-slices to continuants

This subsubsection.... *Axiom: relating instantiation of time slices to instantiation of continuants; i instance_of time_slice(C) iff slice_of(i) instance_of C at some time*

$$\mathsf{instance\_of}(i, \mathsf{time\_slice}(C)) \quad \leftrightarrow \quad \exists t[\mathsf{instance\_of}(\mathsf{slice\_of}(i), C, t)]$$

$$\begin{aligned} &\mathsf{instance\_of}(i, \mathsf{time\_slice}(C)) \wedge \\ &\quad \mathsf{t{:}hasDuration}(i, d) \quad \leftrightarrow \quad \forall t[\mathsf{during}(t, d) \rightarrow \mathsf{instance\_of}(\mathsf{slice\_of}(i), C, d)] \end{aligned}$$

Example:

$$\begin{aligned} &\mathsf{owl{:}type}(\mathsf{heart1234}, \mathsf{Heart}) \wedge \\ &\mathsf{owl{:}fact}(\mathsf{heart1234}, \mathsf{t{:}hasDuration}, \mathsf{range\_987}) \quad \rightarrow \quad \forall t[\mathsf{during}(t, \mathsf{range\_987}) \rightarrow \mathsf{instance\_of}(\mathsf{slice\_of}(\mathsf{hea} \end{aligned}$$

Axiom: converting relations between 2 slices to relations between continuants

$$\begin{aligned} \mathsf{instance\_of}(i, \mathsf{time\_slice}(A)) &\wedge \\ \mathsf{instance\_of}(j, \mathsf{time\_slice}(B)) &\wedge \\ \mathsf{occurs\_at}(i, t) &\wedge \\ \mathsf{occurs\_at}(j, t) &\wedge \\ SR(i, j) \quad \leftrightarrow \quad & \mathsf{relslice}(SR, R) \wedge \\ & R(\mathsf{slice\_of}(i), \mathsf{slice\_of}(j), t) \end{aligned}$$

Example:

$$\begin{aligned} \mathsf{owl{:}type}(\mathsf{person1}, \mathsf{Human}) &\wedge \\ \mathsf{owl{:}type}(\mathsf{heart1}, \mathsf{Heart}) &\wedge \\ \mathsf{occurs\_at}(\mathsf{person1}, \mathsf{now}) &\wedge \\ \mathsf{occurs\_at}(\mathsf{heart1}, \mathsf{now}) &\wedge \\ \mathsf{owl{:}fact}(\mathsf{heart1}, \mathsf{located\_in}, \mathsf{person1}) \quad \rightarrow \quad & \mathsf{instance\_of}(\mathsf{person1}, \mathsf{Human}, \mathsf{now}) \wedge \\ & \mathsf{instance\_of}(\mathsf{heart1}, \mathsf{Heart}, \mathsf{now}) \wedge \\ & \mathsf{located\_in}(\mathsf{slice\_of}(\mathsf{heart1}), \mathsf{slice\_of}(\mathsf{person1}), \mathsf{now}) \end{aligned}$$

Axiom: converting relations between slice and non-slice to relations between continuant and occurrent

$$\begin{aligned} \mathsf{instance\_of}(i, \mathsf{time\_slice}(A)) \wedge \\ \neg(\mathsf{instance\_of}(j, \mathsf{time\_slice}(B))) \wedge \\ \mathsf{occurs\_at}(i, t) \wedge \\ SR(i, j) \quad \leftrightarrow \quad & \mathsf{relslice}(SR, R) \wedge \\ & R(\mathsf{slice\_of}(i), j, t) \end{aligned}$$

Example:

$$\begin{aligned} \mathsf{owl{:}type}(\mathsf{heart1}, \mathsf{Heart}) \wedge \\ \mathsf{owl{:}type}(\mathsf{heartdev1}, \mathsf{HeartDevelopment}) \wedge \\ \mathsf{occurs\_at}(\mathsf{heart1}, \mathsf{now}) \wedge \\ \mathsf{owl{:}fact}(\mathsf{heart1}, \mathsf{participates\_in}, \mathsf{heartdev1}) \quad \rightarrow \quad & \mathsf{participates\_in}(\mathsf{heart1}, \mathsf{heartdev1}, \mathsf{now}) \end{aligned}$$

Axiom: succession of time-slices indicates equality of continuants

$$\begin{aligned} \mathsf{instance\_of}(i, \mathsf{slice\_of}(A)) \wedge \\ \mathsf{instance\_of}(j, \mathsf{slice\_of}(B)) \wedge \\ \mathsf{preceded\_by}(j, i) \quad \leftrightarrow \quad & \mathsf{slice\_of}(i) = \mathsf{slice\_of}(j) \end{aligned}$$

$$\begin{aligned} \mathsf{instance\_of}(i, \mathsf{slice\_of}(A)) \wedge \\ \mathsf{instance\_of}(j, \mathsf{slice\_of}(B)) \wedge \\ \mathsf{occurs\_at}(i, t1) \wedge \\ \mathsf{occurs\_at}(j, t2) \wedge \\ \mathsf{preceded\_by}(j, i) \quad \leftrightarrow \quad & \mathsf{instance\_of}(\mathsf{slice\_of}(i), A, t1) \wedge \\ & \mathsf{instance\_of}(\mathsf{slice\_of}(j), B, \mathsf{t2}) \wedge \\ & \mathsf{slice\_of}(i) = \mathsf{slice\_of}(j) \end{aligned}$$

Example:

$$\begin{aligned} \mathsf{owl{:}type}(\mathsf{heart1}, \mathsf{Heart}) \wedge \\ \mathsf{owl{:}type}(\mathsf{heart2}, \mathsf{ExtractedHeart}) \wedge \\ \mathsf{occurs\_at}(\mathsf{heart1}, \mathsf{t1}) \wedge \\ \mathsf{occurs\_at}(\mathsf{heart2}, \mathsf{t2}) \wedge \\ \mathsf{precededed\_by}(\mathsf{t2}, \mathsf{t1}) \wedge \end{aligned}$$

$$\mathsf{owl{:}fact}(\mathsf{heart2}, \mathsf{preceded\_by}, \mathsf{heart1}) \rightarrow \mathsf{instance\_of}(\mathsf{slice\_of}(\mathsf{heart1}), \mathsf{Heart}, \mathsf{t1}) \wedge \mathsf{instance\_of}(\mathsf{slice\_of}(\mathsf{heart2}), \mathsf{ExtractedHeart}, \mathsf{t2}) \wedge \mathsf{slice\_of}(\mathsf{heart1}) = \mathsf{slice\_of}(\mathsf{heart2})$$

Axiom: transformation_of; In RO, transformation_of is a type-level relation only. ; Should this go in RO?

$$\mathsf{transformation\_of}(A, B) \leftrightarrow \mathsf{preceded\_by}(\mathsf{time\_slice}(A), \mathsf{time\_slice}(B))$$

7.2 Simplified Translation (not normative)

Abstract

Simple mapping to OWL, ignoring temporal arguments

$$\mathsf{type}(A) \leftrightarrow \mathsf{owl{:}Class}(A)$$

$$\mathsf{instance}(A) \leftrightarrow \mathsf{owl{:}Individual}(A)$$

$$\mathsf{relation}(R) \leftrightarrow \mathsf{owl{:}ObjectProperty}(A) \vee \mathsf{owl{:}DatatypeProperty}(A)$$

$$\mathsf{transitive}(R) \leftrightarrow \mathsf{owl{:}TransitiveProperty}(R)$$

Axiom: is_a treated the same as SubClass, ignoring the temporal part of the definition for continuants

$$\mathsf{is\_a}(A, b) \leftrightarrow \mathsf{owl{:}SubClassOf}(A, B)$$

Axiom: subrelation = subPropertyOf

$$\mathsf{subrelation}(A, B) \leftrightarrow \mathsf{owl{:}SubPropertyOf}(A, B)$$

$$\mathsf{domain}(R, X) \leftrightarrow \mathsf{owl{:}domain}(R, X)$$

$$\mathsf{range}(R, X) \leftrightarrow \mathsf{owl{:}range}(R, X)$$

$$\mathsf{disjoint\_from}(A, B) \quad \leftrightarrow \quad \mathsf{owl{:}DisjointFrom}(A, B)$$

Axiom: type-level all-some relation treated as existential restrictions

$$\begin{aligned} R(A, B) \, \wedge \\ \mathsf{all\_some}(R, RI) \quad &\leftrightarrow \quad \mathsf{owl{:}SubClassOf}(A, \mathsf{owl{:}someValuesFrom}(RI, B)) \end{aligned}$$

Axiom: type-level all-only relation treated as universal restrictions

$$\begin{aligned} R(A, B) \, \wedge \\ \mathsf{all\_only}(R, RI) \quad &\leftrightarrow \quad \mathsf{owl{:}SubClassOf}(A, \mathsf{owl{:}allValuesFrom}(RI, B)) \end{aligned}$$

Axiom: intersectionOf

$$\mathsf{equivalent\_to}(A, \mathsf{intersection\_of}(P, Q)) \quad \leftrightarrow \quad \mathsf{owl{:}EquivalentClass}(A, \mathsf{owl{:}intersectionOf}(P, Q))$$

Axiom: disjoint_over

$$\begin{aligned} \mathsf{disjoint\_over}(r, s) \, \wedge \\ r(A, B) \quad &\leftrightarrow \quad \mathsf{owl{:}SubClassOf}(\mathsf{owl{:}intersectionOf}(\mathsf{owl{:}someValuesFrom}(s, A), \mathsf{owl{:}someVa} \end{aligned}$$

Axiom: maximal_over

$$\begin{aligned} \mathsf{maximal\_over}(r, over) \, \wedge \\ \mathsf{inverse\_of}(over, iover) \, \wedge \\ r(a, x, y) \quad &\leftrightarrow \quad \mathsf{owl{:}SubClassOf}(a, \mathsf{owl{:}allValuesFrom}(iover, \mathsf{owl{:}intersectionOf}(\mathsf{owl{:}so} \end{aligned}$$

Theorem: inverse_of, instance level

$$\begin{aligned} \mathsf{inverse\_of}(r, s) \, \wedge \\ \mathsf{instance\_instance}(r) \, \wedge \\ r(A, B) \quad &\leftrightarrow \quad \mathsf{owl{:}fact}(B, s, A) \end{aligned}$$

Theorem: inverse_of, type level

$$\begin{aligned}
\mathsf{inverse_of}(r,s) \wedge & \\
\mathsf{type_type}(r) \wedge & \\
\mathsf{all_some}(r,ri) \wedge & \\
r(A,B) \quad \leftrightarrow & \quad \mathsf{owl{:}SubClassOf}(A, \mathsf{owl{:}allValuesFrom}(RI,B)) \wedge \\
& \quad \mathsf{owl{:}SubClassOf}(B, \mathsf{owl{:}allValuesFrom}(RI,A))
\end{aligned}$$

Axiom: homeomorphic_for

$$\mathsf{homeomorphic\_for}(r,A) \quad \leftrightarrow \quad \mathsf{owl{:}SubClassOf}(A, \mathsf{owl{:}allValuesFrom}(r,A))$$

Axiom: transitive_over

$$\mathsf{transitive\_over}(r,s) \quad \leftrightarrow \quad \mathsf{owl{:}SubPropertyOf}(r, \mathsf{SubObjectPropertyChain}(r,s))$$

Axiom: holds_over_chain

$$\mathsf{holds\_over\_chain}(r,s,t) \quad \leftrightarrow \quad \mathsf{owl{:}SubPropertyOf}(r, \mathsf{SubObjectPropertyChain}(s,t))$$

7.3 OWL Full (RDFS) Translation

Abstract

Mapping to OWL-Full, with types and type-level relations in the domain of discourse.

7.3.1 Meta-Relation: triple

We use a ternary relation of type triple to store RDF facts. This is to easily segregate the RDF universe from the rest of the Obolog universe.

$$\mathsf{is\_a}(A,b) \quad \leftrightarrow \quad \mathsf{rdfs{:}SubClassOf}(A,B)$$

Axiom: subrelation = subPropertyOf

$$\mathsf{subrelation}(A,B) \quad \leftrightarrow \quad \mathsf{rdfs{:}SubPropertyOf}(A,B)$$

Axiom: all binary relations in Obol are added as triples

$$\begin{aligned}
R(A,B) \wedge & \\
\mathsf{relation}(R) \quad \leftrightarrow & \quad \mathsf{triple}(R,A,B)
\end{aligned}$$

7.4 Obolog-lex Translation

Annotation Properties

7.5 Obolog-A Translation

TODO

OWL2 features?

8 Glossary

- Instance
- Type
- Relation
- Relationship
- Sentence
- Tuple
- Text