# CO3L Summary

2016-04-15

## Table of Contents

# 1 Introduction

CO3L is a very simple relational language for representing O3F ontologies. This section presents an example of an ontology specification. The other sections summarise existing predicates, facets and types.

The following is a partial ontology for the on-line shopping domain. The example illustrates some important CO3L features including class and association definition, hierarchy definition, predicate and propositional symbol definition, data type definition and the use of facets. Classes (as associations) may have attributes and methods, as is the case of the Product class defined below.

Ontology on-line-shopping {

Version : “beta temporary version”

Owner : “O3F Group”

Initial_date : 2010/09/01

Last_modification_date : 2010/09/01

Class(Product)

Attribute(Product, price, Price)
EntityFacet(Product.price, Mandatory, True)

Attribute(Product, quantity, Non_negative_integer_type)
EntityFacet(Product.quantity, Mandatory, True)

Attribute(Product, title, String)
EntityFacet(Product.title, Mandatory, True)
EntityFacet(Product.title, Distinct, True)

ActionMethod(Product, sell)

Argument(Product.sell, client_email, String)
EntityFacet(Product.sell, Return_type, Selling_completion_status_type)
EntityFacet(Product.sell, Return_role, Selling_completion_status)

Class(Author)

Attribute(Author, name, String)
EntityFacet(Author.name, Mandatory, True)
EntityFacet(Author.name, Distinct, True)

Attribute(Author, nationality, String)
Association(ProductAuthor)

Argument(ProductAuthor, product, Product)
EntityFacet(ProductAuthor.product, Multiplicity, 1..*)

Argument(ProductAuthor, author, Author, 1..*)
EntityFacet(ProductAuthor.author, Multiplicity, 1..*)

Class(DVD)

Class(CD)

Class(Book)

Class(Movie)

Class(Education)

Class(Entertainment)

Hierarchy(ProductHierarchy)

Subtype(ProductHierarchy, Product, DVD)
Subtype(ProductHierarchy, Product, CD)
Subtype(ProductHierarchy, Product, Book)
Subtype(ProductHierarchy, DVD, Movie)
Subtype(ProductHierarchy, DVD, Education)
Subtype(ProductHierarchy, DVD, Entertainment)

EntityFacet(ProductHierarchy, Root, Product)
EntityFacet(ProductHierarchy!Product, Complete, True)

Predicate(Best_price)
Argument(Best_price, price, Price)

PropSymbol(Gift_wrapped)

PropSymbol(Gift_paper_depleted)

PropSymbol(Unknown_product)

Class(Price)

Attribute(Price, currency, Word)
EntityFacet(Price.currency, Default_value, Euro)

Attribute(Price, amount, Non_negative_float_type)
EntityFacet(Price.amount, Mandatory, True)

Datatype(Non_negative_integer_type, Integer)
EntityFacet(Non_negative_integer_type, Smallest_instance, 0)

Datatype(Non_negative_float_type, Integer)
EntityFacet(Non_negative_float_type, Smallest_instance, 0)

Datatype(Selling_completion_status_type, Word)
EntityFacet(Selling_completion_status_type, Instances_set, Set(successful, unsuccessful)

}

# 2 Predicates Summary

| | |
|---|---|
| **Simple type definition**<br>Datatype(Name, Type) | **Hierarchies**<br>Hierarchy(Name)<br>Subtype(Hierarchy, Type, SubType) |
| ***Classifiers*: classes and associations**<br>Class(Name)<br>Association(Name)<br>Attribute(Classifier, Attribute, Type)<br>RelationalMethod(Classifier, Mehtod)<br>FunctionalMethod(Classifier, Method, Type)<br>ActionMethod(Classifier, Method)<br>Argument(Association, Role, Type)<br>Argument(Classifier.Method, Role, Type)<br>Key(Classifier, KeyName, Attribute) | **Individuals**<br>ObjectDef(Name, Definition)<br>Instance(Object, Type) |
| **Operators and propositional symbols**<br>PropoSymbol(Symbol)<br>Predicate(Name)<br>Function(Name, Type)<br>Action(Name)<br>Argument(Operator, Role, Type) | **Dependency relations**<br>Dependency(Name)<br>DependencyArgument(DepRel, Role, Element) |
| **Relationships between attributes and methods, and operators**<br>AttributeFunction(Classifier, Attribute, Function)<br>FMethodFunction(Classif, Method, Function, Self)<br>AMethodAction(Classif, Method, Action, Self)<br>RMethodPredicate(Classif, Method, Pred, Self) | **Facets and axioms**<br>Facet(Name)<br>ValidFacetElement(Facet, Element)<br>ValidFacetType(Facet, Type)<br>EntityFacet(Entity, Facet, Value)<br>Axiom(Name, Expression) |

# 3 Facets Summary

## Facets applicable to all entities

| **Facet** | **Possible values** |
|---|---|
| Visibility | Private/Protected/Ontology/Public. **Public** is the default value |
| Access | Read_only, Final, Read_write.... |
| Dependence | Derived / Derived_union / Dependent /Independent. Independent is the default value |
| Owner | Type specification |

## Facets applicable to sets

| **Facet** | **Possible values** |
|---|---|
| Largest_instance | It depends on the type to which it is applied |
| Smallest_instance | |
| Instance_maximum_size | Non negative integer |
| Instance_minimum_size | |
| Default_instance | It depends on the type to which it is applied |
| Instances_set | Set of possible values (e.g., Set( on, off)). |
| Instances_sequence | Sequence of possible values and its order relation, Sequence(low, medium, high) |

## Facets applicable to *Classifiers* (Classes and Associations)

| **Faceta** | **Possible values** |
|---|---|
| Materialization | Abstract/Concrete/Interface. **Concrete** is the default vaue |
| Realization | Word. Name of the realized interface |
| Process_control | Passive / Active. **Passive** is the default value |
| Association_type | Association, Aggregation, Composition. **Association** is the default value |
| Whole | Name / role of one of an association arguments |
| Part | |

## Hierarchies and generalizations facets

| **Faceta** | **Possible values** |
|---|---|
| Root | *Type Specification*. Name of the hierarchy root node. By default, hierarchy nodes are not root nodes |
| Leaf | *Type Specification*. Name of one of the hierarchy leaf nodes. By default, hierarchy nodes are not leafs |
| Complete | **Boolean**. False is the default value. |

## Facets applicable to attributes, arguments, operators and methods

| **Faceta** | **Possible values** |
|---|---|
| Maximum_value | It depends on the type to which it is applied |
| Minimum_value | |
| Maximum_size | Non negative integer |
| Minimum_size | |
| Default_value | It depends on the type to which it is applied |
| Values_set | Set of possible values (e.g., Set( on, off)) |
| Values_sequence | Sequence of possible values and its respective implicit order relation (e.g., Sequence(low, medium, high)) |
| Mandatory | Boolean. **False** is the default value for attributes. **True** is the default value for arguments |
| Distinct | Boolean. **False** is the default value |
| Scope | Classifier / Instance. **Instance** is the default value. |
| Navigability | Navigable / Non_navigable / Non_specified |
| Return_type | *Type Specification*. There is no default value |
| Return_role | Word. There is no default value |
| Arg_direction | In /Out / Inout. **Inout** is the default value. Arg_direction is not applicable to functions or functional methods |
| Arg_number | Natural (First argument is argument number 1) |
| Multiplicity | The same as in UML Class Diagram. Examples: 1, *, 3..* |

# 4 Pre-defined data types

| Type | Description |
|---|---|
| Thing | Any term. This is the root of the O3F type hierarchy. |
| Scalar | Any simple existing or to be declared datatype, such as Number, Word, Date and Set_of(Char) |
| Classifer | Any class or association |
| Number | Any number, including integers and floats |
| Integer | Integers, including naturals |
| Natural | Positive integers |
| Float | Floating point numbers |
| String | Character strings delimited by double quotes |
| Word | Character strings with only one word. It may contain any letter, number, the hyphen, and the underline. A Word cannot star with a number |
| Char | A single character delimited by quotes (e.g., 'a'). |
| Boolean | *True* / *False* |
| CalendarDate | Represents a date in a given calendar. Format:<br>*Calendar* : *Signal YYYY/MM/DD*<br>*Calendar*: There are several calendars, including the Christian or Gregorian calendar (gc), the islamic calendar (ic), and the Chinese calendar, which does not have a starting point (cc). Calendars are identified by their names, which are Words.<br>*Signal*: Positive (+) is used for more recent dates than the calendar initial date (e.g., AC dates in the Christina or Gregorian calendar); Negative (-) is used for older dates than the calendar initial date (e.g., BC dates in the Christian calendar) |
| Date | Represents a positive date in the Christinan (Gregorian) calendar: YYYY/MM/DD. YYYY/MM/DD is the abbreviated form of gc:+YYYY/MM/DD |
| Time | Time instant: HH:MM:SS:mmm |
| CalendarDate_and_time | Calendar:SYYYY/MM/DD-HH:MM:SS:mmm |
| Date_and_time | YYYY/MM/DD-HH:MM:SS:mmm |
| URL | URL, e.g., http://iscte.pt/ontologies |
| eMailAddress | eMail address, e.g., luis.botelho@iscte.pt |

| **Type** | **Description** |
|---|---|
| Collection | Any kind of collection. The elements of a collection may have all the same type or different types. Possible collections are bags (*Bag*), sets (*Set*) and sequences (*Sequence*). |
| Collection_of | The same as Collection but for collections of single type elements |
| Bag | Collection of possibly repeated elements in which the order is not important |
| Bag_of | The same as Bag, but for bags of single type elements. Example: *Bag_of Char* |
| Ordered_set | Collection of elements without repetition. The order is important |
| Ordered_set_of | Collection of elements of the same type, without repetition. The order is important |
| Set | Collection of different elements in which the order is not important. |
| Set_of | The same as *Set* but for sets of single type elements. Example *Set_of Person* |
| Sequence | Ordered collection of possibly repeated elements |
| Sequence_of | The same as Sequence but for a single type sequence of elements. Example *Sequence_of identity_number* |