

---

---

**Language resource management —  
Corpus query lingua franca (CQLF) —**

**Part 2:  
Ontology**

*Gestion des ressources linguistiques — Corpus query lingua franca  
(CQLF) —*

*Partie 2: Ontologie*

STANDARDSISO.COM : Click to view the full PDF of ISO 24623-2:2021



STANDARDSISO.COM : Click to view the full PDF of ISO 24623-2:2021



**COPYRIGHT PROTECTED DOCUMENT**

© ISO 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office  
CP 401 • Ch. de Blandonnet 8  
CH-1214 Vernier, Geneva  
Phone: +41 22 749 01 11  
Email: [copyright@iso.org](mailto:copyright@iso.org)  
Website: [www.iso.org](http://www.iso.org)

Published in Switzerland

# Contents

	Page
<b>Foreword</b> .....	<b>iv</b>
<b>Introduction</b> .....	<b>v</b>
<b>1 Scope</b> .....	<b>1</b>
<b>2 Normative references</b> .....	<b>1</b>
<b>3 Terms and definitions</b> .....	<b>1</b>
<b>4 Motivation and aims</b> .....	<b>3</b>
<b>5 Structure and content of a CQLF ontology</b> .....	<b>4</b>
5.1 OWL DL formalism.....	4
5.2 Structure of the ontology.....	5
5.3 CQLF metamodel.....	7
5.4 Functionalities.....	8
5.5 Frames.....	11
5.6 Use cases.....	11
5.7 CQLs.....	12
<b>6 Conformance statements</b> .....	<b>12</b>
6.1 Positive conformance statements.....	12
6.2 Negative conformance statements.....	13
<b>Annex A (informative) Illustrative example of a CQLF ontology</b> .....	<b>15</b>
<b>Bibliography</b> .....	<b>18</b>

STANDARDSISO.COM : Click to view the full PDF of ISO 24623-2:2021

## Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see [www.iso.org/directives](http://www.iso.org/directives)).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see [www.iso.org/patents](http://www.iso.org/patents)).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see [www.iso.org/iso/foreword.html](http://www.iso.org/iso/foreword.html).

This document was prepared by Technical Committee ISO/TC 37, *Language and terminology*, Subcommittee SC 4, *Language resource management*.

A list of all parts in the ISO 24623 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](http://www.iso.org/members.html).

## Introduction

Several families of International Standards codify various aspects of the representation of language data. These standards describe general corpus-oriented data models in the linguistic annotation framework (LAF) (see ISO 24612), various aspects of the semantic representation in the semantic annotation framework (SemAF) (see ISO 24617-1 and others), the representation of lexical data in the lexical markup framework (LMF) (see ISO 24613-1 and others), as well as the representation of metadata in the component metadata infrastructure (CMDI) (see ISO 24622-1 and others). Complementary to the standards concerning the representation of language data, the ISO 24623 series focuses on the exploitation of language data and on ways to satisfy various kinds of information needs targeting these data.

The corpus query lingua franca (CQLF) metamodel, described in ISO 24623-1, is a maximally permissive construct that establishes means of describing the scope of corpus query languages (CQLs) at a general level and with a focus on various kinds of data models assumed by query systems, with conformance conditions meant to be satisfied by a wide range of CQLs. The metamodel provides a “skeleton” for a CQL taxonomy by setting up basic categories of corpus queries (encoded as levels and modules) as well as the dependencies among them.

Consequently, the task of a more concrete characterization of CQLs is meant to be covered in other parts of the ISO 24623 series. This document establishes a framework for an ontology which focuses on the generalized information needs satisfied by corpus queries, and which is structured as a multi-layer taxonomy against which individual CQLs can make positive and negative conformance statements.

Such an ontology allows, on the one hand, a fine-grained comparison of the expressive power of CQLs, and, on the other hand, it serves a practical purpose, i.e. as a foundation for platforms where developers can enter conformance statements, and where end users can see which CQL to turn to in order to ensure that their search needs get satisfied. An example of such a platform is given by Reference [13].

STANDARDSISO.COM : Click to view the full PDF of ISO 24623-2:2021

# Language resource management — Corpus query lingua franca (CQLF) —

## Part 2: Ontology

### 1 Scope

This document specifies the structure of an ontology for a fine-grained description of the expressive power of corpus query languages (CQLs) in terms of search needs. The ontology consists of three interrelated taxonomies of concepts: the CQLF metamodel (a formalization of ISO 24623-1); the expressive power taxonomy, which describes different facets of the expressive power of CQLs; and a taxonomy of CQLs.

This document specifies:

- a) the taxonomy of the CQLF metamodel;
- b) the topmost layer of the expressive power taxonomy (whose concepts are called “functionalities”);
- c) the structure of the layers of the expressive power taxonomy and the relationships between them, in the form of subsumption assertions;
- d) the formalization of the linkage between the CQL taxonomy and the expressive power taxonomy, in the form of positive and negative conformance statements.

This document does not define the entire contents of the ontology (see [Clause 4](#)).

### 2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 24612, *Language resource management — Linguistic annotation framework (LAF)*

ISO 24623-1, *Language resource management — Corpus query lingua franca (CQLF) — Part 1: Metamodel*

ISO/IEC 10646, *Information technology — Universal coded character set (UCS)*

W3C-OWL 2-SPEC. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax* (Second Edition). MOTIK B., PATEL-SCHNEIDER, P.F., and PARSIA, B. eds. W3C Recommendation, 11 December 2012. Available from: <http://www.w3.org/TR/owl2-syntax/>

### 3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 24612, ISO 24623-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at <https://www.iso.org/obp>
- IEC Electropedia: available at <https://www.electropedia.org/>

**3.1**  
**CQLF module**  
subcomponent of the CQLF metamodel, defined with reference to a specified data-model characteristic

Note 1 to entry: The CQLF metamodel currently distinguishes three modules within CQLF Level 1, Linear (plain-text, segmentation and simple annotation), and three modules within CQLF Level 2, Complex (hierarchical, dependency and containment).

Note 2 to entry: In 5.3, the containment module is formalized by the concept SpanContainment in order to avoid terminological ambiguity.

[SOURCE: ISO 24623-1:2018, 3.8, modified — “the CQLF metamodel” has replaced “a CQLF level” in order to improve clarity outside the context of ISO 24623-1; Note 2 to entry has been added.]

**3.2**  
**functionality**  
label for a concept in a *CQLF ontology* (3.15) that represents a family of *CQL capabilities* (3.12) contributing to the expressive power of a *CQL* (3.5), formulated at a general level and linked to one or more *CQLF modules* (3.1)

**3.3**  
**frame**  
label for a concept in a *CQLF ontology* (3.15) that represents a typical *search need* (3.6) of *end users* (3.7), understood as one facet of the expressive power of *CQLs* (3.5)

Note 1 to entry: Most frames arise from the specialization of a *functionality* (3.2) and/or the combination of multiple functionalities.

**3.4**  
**use case**  
label for a concept in a *CQLF ontology* (3.15) that represents a concrete instantiation of a *frame* (3.3), for which it can be determined unambiguously whether a given *query expression* (3.8) satisfies the *search need* (3.6) or not

Note 1 to entry: Use cases are often parameterized, i.e. they contain variable elements. Parameterized use cases are satisfied by parameterized query expressions.

**3.5**  
**CQL**  
**corpus query language**  
formal language designed to retrieve specific information from (large) language data collections, and thereby incorporate certain abstractions over commonly shared data models that make it possible for the *end user* (3.7) (or user agents) to address parts of those data models

Note 1 to entry: A CQL defines a syntactic notation for *query expressions* (3.8) and the corresponding search semantics, i.e. an intensional specification of the intended result set. For most current CQLs, semantics are implicitly defined by a particular implementation.

[SOURCE: ISO 24623-1:2018, 3.4, modified — “CQL” has been added as preferred term, “end user” has replaced “user” in the definition and Note 1 to entry has been added.]

**3.6**  
**search need**  
information pattern that an *end user* (3.7) wants to locate in a corpus, based on the primary data stream and/or simple or complex annotation

**3.7****end user**

agent who uses a *CQL* (3.5) to satisfy his or her *search needs* (3.6)

Note 1 to entry: This can be done via an interactive graphical user interface (GUI), a command-line tool, programmatically via some application programming interface (API) or by a software program developed by the end user.

**3.8****query expression**

string that is syntactically valid in a given *CQL* (3.5) and can be executed to return a result set

Note 1 to entry: Query expressions are often parameterized with variable elements. No formal specification of the parameter substitution procedure is attempted, but entries for parameterized query expressions in the ontology are required to include informal descriptions of the range of admissible values and any transformations required.

**3.9****parameter**

variable element in a *query expression* (3.8) or in the description of a *search need* (3.6)

**3.10****positive conformance statement**

assertion that a given *CQL* (3.5) supports a given *use case* (3.4) by means of a *query expression* (3.8)

**3.11****negative conformance statement**

assertion that a given *CQL* (3.5) cannot support a given *use case* (3.4), *frame* (3.3) or *functionality* (3.2)

Note 1 to entry: Negative conformance is due to technical unavailability of specific capabilities in the respective *CQL* or limitations on the complexity of *query expressions* (3.8).

**3.12****CQL capability  
capability**

corpus query language capability  
facility provided by *CQLs* (3.5) to meet a specific aspect of *search needs* (3.6)

**3.13****layer**

totality of concepts at the same level of abstraction in a *CQLF ontology* (3.15)

EXAMPLE *Functionalities* (3.2), *frames* (3.3), *use cases* (3.4).

**3.14****token**

non-empty contiguous sequence of graphemes or phonemes in a document

[SOURCE: ISO 24611:2012, 3.21, modified — Note 1 to entry has been deleted.]

**3.15****CQLF ontology**

ontology for a fine-grained description of the expressive power of *CQLs* (3.5) in terms of *search needs* (3.6), which adheres to the structure specified in this document

**4 Motivation and aims**

CQLs differ widely in their basic sets of capabilities. Whereas some are restricted to rather specific application scenarios, others are able to cover a wider variety of applications and search needs. It is therefore both the quality and the quantity of CQL capabilities – as well as the degree to which they can be combined freely – that determine the expressive power of a CQL. A CQLF ontology as specified

in this document is not intended to articulate all the possible combinations of capabilities unless these are justified by genuine usage. Its aim is to provide representative categories for typical search needs within a taxonomy of CQL capabilities. These typical search needs evolve with general progress in the fields of corpus linguistics and digital humanities, and with the discovery of new challenges, new methods and new research questions. In order to accommodate the dynamic nature of the evolving search needs, most of the content of such an ontology is outside the scope of standardization. This document provides a structural framework for this dynamic information (by specifying the three-layer structure of the expressive power taxonomy, the content of the topmost layer of functionalities, and the relationships between different layers and taxonomies), ensuring that the ontology can adapt to new search needs that emerge as the relevant disciplines evolve.

In order to provide a normative skeleton for the ontology while at the same time making provisions for keeping its main content (search needs and corresponding query expressions) dynamic, this document does not comprise a normative listing of the middle and bottom layer of the expressive power taxonomy (i.e. frames and use cases). An exhaustive inventory of concepts at these two layers is not possible due to the fact that existing CQLs differ widely in the complexity of the supported combinations of functionalities, that new CQLs can be created offering additional combinations or subtypes of functionalities, and that new search needs emerge from progress in the relevant research fields. The frames and use cases of a CQLF ontology are expected to be supplied by a moderated community process, driven by CQL developers as well as end users (see Reference [13]). For illustration, a sample of frames and use cases together with conformance statements linking them with the CQP<sup>[6]</sup> and ANNIS<sup>[8]</sup> query languages is provided in [Annex A](#).

The permissive architecture and terminology defined by this document enables research groups to extend the relevant parts of the ontology with further CQL capabilities and search needs arising in future.

The following application scenarios are thus made possible:

- describing the scope and capabilities of a given CQL, in terms of conformance statements against a CQLF ontology (typically carried out by the CQL developers);
- comparing different CQLs with respect to their ability to meet typical search needs;
- identifying suitable CQLs and query tools that support (combinations of) CQL capabilities required by an end user, together with examples of the respective query syntax;
- guiding the development of new CQLs and query tools by building an inventory of complex search needs that are important for the community (typically carried out by end users).

## 5 Structure and content of a CQLF ontology

### 5.1 OWL DL formalism

The taxonomic framework for a CQLF ontology is modelled in OWL 2 DL<sup>[7]</sup> – a dialect of the Web Ontology Language (OWL) based on the family of description logics (DL) (see Reference [9]) as a formal framework. All definitions and requirements of the W3C OWL 2 specification shall be followed. The normative representation and exchange format for a CQLF ontology is RDF/XML<sup>[10][11]</sup>. All labels and annotations shall be represented as sequences of Unicode code points, in accordance with ISO/IEC 10646.

W3C OWL 2 DL furnishes developers with a set of tools for:

- a) stating concept hierarchies and membership of individuals,
- b) defining highly expressive property restrictions.

In particular, this document makes use of the AnnotationProperty construct of OWL DL in order to associate additional information with concepts and individuals.

For better readability, CQLF ontology axioms are provided in DL notation in [Clauses 5](#) and [6](#) rather than in the RDF/XML exchange format.

Relevant DL notions<sup>[9]</sup>:

- **concept inclusion**  $\sqsubseteq$ : This operator asserts a logical subsumption relationship between two concept expressions.

EXAMPLE 1  $A \sqsubseteq B$  asserts that  $A$  covers either a subset or the entire set of individuals contained in  $B$ .  $A$  is also said to be subsumed by  $B$ .

NOTE 1 The same notation is sometimes used to express the opposite relation ( $A$  subsumes  $B$ ) for feature structures (see Reference [12], p. 496).

- **concept equivalence**  $\equiv$ : This operator asserts an equivalence between two concept expressions.

EXAMPLE 2  $A \equiv B$  asserts that  $A$  covers exactly the same set of individuals as  $B$ .

- **intersection/conjunction**  $\sqcap$ : This operator denotes the intersection of two concept expressions, i.e. the individuals contained in both concept expressions.

NOTE 2  $A \sqsubseteq B \sqcap C$  asserts that  $A$  is subsumed by  $B$  as well as  $C$ . It is equivalent to the assertions  $A \sqsubseteq B$  and  $A \sqsubseteq C$ .

- **union/disjunction**  $\sqcup$ : This operator denotes the union of two concept expressions, i.e. the individuals contained in either or both of the concept expressions.

NOTE 3  $A \sqsubseteq B \sqcup C$  does not imply that  $A$  is subsumed by either  $B$  or  $C$  on its own. Some of the individuals covered by  $A$  can be contained in  $B$  and others in  $C$ .

- **top concept**  $\top$ : denotes the set of all individuals in the domain, i.e. the entire universe. Also referred to as “Thing” or “the root class”.

- **bottom concept**  $\perp$ : denotes the empty set of individuals in the domain. Also referred to as “Nothing” or “the empty class”.

- **concept assertion**  $\in$ : This operator asserts that an individual belongs to a concept. Also known as “class assertion” because the concepts represent classes (see T-Box below).

EXAMPLE 3  $x \in A$  asserts that the individual  $x$  is a member of the concept  $A$ .

- **A-Box**: The domain of interest is spanned by a universe of individuals which serve as the fundamental atoms for the ontology of what shall be modelled. They become members of concepts through concept assertions (also referred to as “A-Box axioms”) and implicitly through the subsumption relations expressed by concept inclusion assertions (in the T-Box).

- **T-Box**: Concepts are represented within the terminological box (T-Box). They are classes into which individuals are organized by the A-Box axioms. The T-Box thus provides a vocabulary of concepts and a rule set of hierarchical relations between them (“is-a” relations expressed by concept inclusion axioms). Ideally, sibling categories cover a mutually exclusive space of sub-categories and/or individuals.

## 5.2 Structure of the ontology

The T-Box of a CQLF ontology consists of three separate taxonomies of concepts. The main taxonomy describes different facets of the expressive power of CQLs. It is called “expressive power taxonomy” and is divided into three layers.

Concepts in the top layer are called “functionalities”. They represent (families of) individual search capabilities that can be provided by CQLs at a general level. Functionalities serve as entry points for navigating the main taxonomy. Functionalities belong to the normative part of the ontology and are defined in [5.4](#).

Concepts in the middle layer are called “frames”. They represent typical search needs of end users, which often involve combinations of multiple functionalities, at a relatively abstract level. For every frame, subsumption assertions shall indicate which functionalities are required for the search need. A frame  $A$  can also be subsumed by another frame  $A'$  if  $A$  extends the search need represented by  $A'$ . The normative part of the ontology does not include any instances of frames; the structure of the frame layer is defined in [5.5](#).

Concepts in the bottom layer are called “use cases”. They represent parameterized instantiations of frames, which should be sufficiently concrete so that it is possible to determine unambiguously whether a given CQL can satisfy a given use case. For every use case, a subsumption assertion shall indicate which frame is instantiated by the use case. There can also be subsumption assertions to multiple frames as well as to other use cases. The normative part of the ontology does not include any instances of use cases; the structure of the use case layer is defined in [5.6](#).

The second taxonomy of concepts formalizes the CQLF metamodel defined in ISO 24623-1. Subsumption assertions link all functionalities to the CQLF metamodel. Both the CQLF metamodel taxonomy (defined in [5.3](#)) and the subsumption assertions (defined in [5.4](#)) belong to the normative part of the ontology.

The third taxonomy of concepts represents individual CQLs whose expressive power is described with respect to the ontology. It shall have a flat structure without subsumption assertions between different CQLs. The normative part of the ontology does not include any instances of CQLs; the structure of the taxonomy is defined in [5.7](#).

Individuals in the A-Box are positive conformance statements in the form of parameterized query expressions. Concept assertions shall assign each individual to a CQL concept (representing the CQL it is formulated in) and to a use case concept (representing the search need that the query expression satisfies). The normative part of the ontology does not include any individuals, i.e. its A-Box is empty. A CQL can also make negative conformance statements to declare that it cannot satisfy specific use cases, frames or functionalities because of its design limitations. As general disjunction assertions for concepts, negative conformance statements are part of the T-Box. If neither a positive nor a negative conformance statement exists between a CQL and a given use case, it shall be considered undetermined whether or not the CQL can satisfy the corresponding search need. Positive and negative conformance statements are further defined in [Clause 6](#).

No concept or subsumption assertion shall be made that would lead to logical inconsistencies in the ontology.

The overall structure of a CQLF ontology is illustrated in [Figure 1](#).

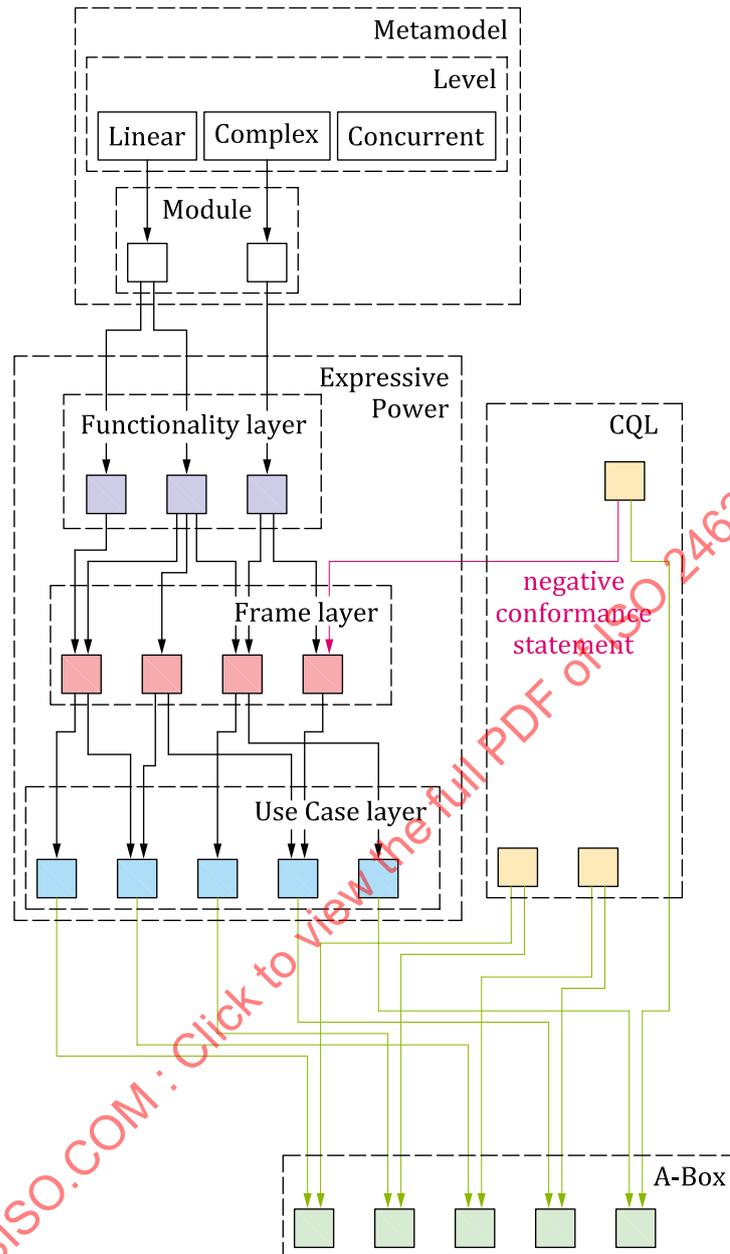


Figure 1 — General structure of a CQLF ontology

### 5.3 CQLF metamodel

The theoretical concept of modules<sup>[5]</sup> as standardized in the context of ISO 24623-1 is formalized by the CQLF metamodel taxonomy. It consists of the concepts and subsumption assertions defined below. Each concept is identified by its label (as `rdfs:label` annotation), followed by all its subsumption assertions within the taxonomy.

Abstract root concepts:

- **Metamodel:** This is the abstract root concept of the CQLF metamodel taxonomy.
- **Level**  $\sqsubseteq$  Metamodel: This is the abstract root concept of all CQLF levels.
- **Module**  $\sqsubseteq$  Metamodel: This is the abstract root concept of all CQLF modules.

CQLF levels:

- **Linear**  $\sqsubseteq$  Level: Plain-text search as well as search in segmented data.
- **Complex**  $\sqsubseteq$  Level: Search in data annotated with hierarchical structures and/or dependency information, or querying simple annotations by means of containment-based queries.
- **Concurrent**  $\sqsubseteq$  Level: Search in multiple concurrent (overlapping, intersecting and often conflicting) annotations built upon a single data stream.

CQLF modules:

- **PlainText**  $\sqsubseteq$  Module  $\sqcap$  Linear: Segmentation-independent string search.
- **SimpleAnnotation**  $\sqsubseteq$  Module  $\sqcap$  Linear: Segmentation-based search for annotations describing the primary data stream; understood more generally as search for annotations of individual objects in the context of this document.
- **Segmentation**  $\sqsubseteq$  Module  $\sqcap$  Linear: Search for segmental annotation, in particular tokens and token sequences.
- **Hierarchical**  $\sqsubseteq$  Module  $\sqcap$  Complex: Tree-based representations, e.g. for phrase-structure description.
- **Dependency**  $\sqsubseteq$  Module  $\sqcap$  Complex: Identification of relationships in which objects function as nodes linked by directed arcs.
- **SpanContainment**  $\sqsubseteq$  Module  $\sqcap$  Complex: Non-recursive simplified hierarchical relationships encoded as character span containment.
- **Paradigmatic**  $\sqsubseteq$  Module  $\sqcap$  Concurrent: Different annotation layers providing data packages describing the same location.
- **Overlapping**  $\sqsubseteq$  Module  $\sqcap$  Concurrent: Concurrent annotations built upon character spans which overlap in their start and/or end offsets.

As the coarsest categories of corpus query classification, CQLF modules provide a prior global and maximally comprehensive framework for the organization of search needs and CQLs. The relationship between the metamodel taxonomy and the expressive power taxonomy is not strictly hierarchical. For this reason, functionalities are more suitable as entry points for navigation of the ontology than CQLF modules.

## 5.4 Functionalities

Functionalities represent individual capabilities of CQLs at a very general level. They form the top layer of the expressive power taxonomy and serve as navigational entry points. Functionalities are connected to CQLF modules via subsumption assertions.

Some functionalities (such as `PartialMatch`) apply to multiple CQLF modules. They are not subsumed by any of these modules but rather by their disjunction, indicated by an assertion of the form  $A \sqsubseteq B \sqcup C$  below.

The layer of functionalities consists of the following concepts, descriptions and subsumption assertions. Each functionality concept is identified by its label (as `rdfs:label` annotation) and a subsumption assertion to the abstract root concept `Functionality`, followed by a description of the search need in italics (as `cqlf:searchNeed` annotation) and further subsumption assertions that connect the functionality to the Metamodel taxonomy.

Abstract root concepts:

- **ExpressivePower**

— **Functionality**  $\sqsubseteq$  ExpressivePower

Functionalities:

— **Annotation**  $\sqsubseteq$  Functionality

*find individual objects based on their linguistic annotation*

Annotation  $\sqsubseteq$  SimpleAnnotation  $\sqcup$  Paradigmatic

— **ConstraintCombination**  $\sqsubseteq$  Functionality

*Boolean operators for combining constraints on objects*

ConstraintCombination  $\sqsubseteq$  Metamodel

EXAMPLE 1 Conjunction (&), disjunction (|), negation (!, !=), difference (-).

— **Containment**  $\sqsubseteq$  Functionality

*containment of an object in a specific context*

Containment  $\sqsubseteq$  SpanContainment  $\sqcup$  Overlapping

— **ExternalResource**  $\sqsubseteq$  Functionality

*reference to externally encoded structured data*

ExternalResource  $\sqsubseteq$  Metamodel

EXAMPLE 2 Reference from annotation object to dictionary, database of speakers.

— **FuzzySearch**  $\sqsubseteq$  Functionality

*approximate string matching*

FuzzySearch  $\sqsubseteq$  Segmentation  $\sqcup$  PlainText  $\sqcup$  SimpleAnnotation

— **GraphRelation**  $\sqsubseteq$  Functionality

*relationships of objects as nodes linked by directed arcs*

GraphRelation  $\sqsubseteq$  Dependency

— **LinearRelation**  $\sqsubseteq$  Functionality

*horizontal relationships of objects*

LinearRelation  $\sqsubseteq$  Segmentation  $\sqcup$  PlainText

EXAMPLE 3 Precedence/co-occurrence of objects.

— **MatchingStrategy**  $\sqsubseteq$  Functionality

*define specific matching modality*

MatchingStrategy  $\sqsubseteq$  Metamodel

EXAMPLE 4 Match the first object in a sequence, match the whole span between two objects, greedy versus non-greedy search.

— **Metainformation**  $\sqsubseteq$  Functionality

*query metainformation associated with the primary data*

Metainformation  $\sqsubseteq$  Metamodel

EXAMPLE 5 Text genre, publication date, author sex.

— **PartialMatch**  $\sqsubseteq$  Functionality

*match plain text or linguistic annotation value against a generalized pattern*

PartialMatch  $\sqsubseteq$  PlainText  $\sqcup$  SimpleAnnotation

EXAMPLE 6 Prefix/suffix search, match against regular expression.

— **PlainTextSearch**  $\sqsubseteq$  Functionality

*find segmentation-independent strings*

PlainTextSearch  $\sqsubseteq$  PlainText

EXAMPLE 7 “[...]ad a little la[...]”, “[...]mber 47 said to num[...]”

— **Position**  $\sqsubseteq$  Functionality

*relative position of an object with respect to another*

Position  $\sqsubseteq$  PlainText  $\sqcup$  Segmentation  $\sqcup$  SpanContainment  $\sqcup$  Hierarchical  $\sqcup$  Dependency

EXAMPLE 8 Containment at left, distance between two objects.

— **Quantification**  $\sqsubseteq$  Functionality

*numeric range specification on a constraint*

Quantification  $\sqsubseteq$  Metamodel

EXAMPLE 9 At least  $N$ , at most  $M$ .

— **Repetition**  $\sqsubseteq$  Functionality

*multiple occurrence of an object*

Repetition  $\sqsubseteq$  PlainText  $\sqcup$  Segmentation

— **Sensitivity**  $\sqsubseteq$  Functionality

*treatment of special character features*

Sensitivity  $\sqsubseteq$  PlainText  $\sqcup$  Annotation

EXAMPLE 10 Case-insensitive, ignore diacritics.

— **Size**  $\sqsubseteq$  Functionality

*specify the size of an object*

Size  $\sqsubseteq$  PlainText  $\sqcup$  Segmentation  $\sqcup$  Hierarchical  $\sqcup$  Dependency

EXAMPLE 11 String length, number of objects in sequence, arity of tree branch or graph vertex.

— **TreeRelation**  $\sqsubseteq$  Functionality

*vertical relationships of objects as nodes linked by hierarchically directed arcs*

TreeRelation  $\sqsubseteq$  Hierarchical

EXAMPLE 12 Domination of an object by another, common parent node.

The extension of a functionality  $F$  is the set of all parameterized query expressions that involve  $F$ . The functionality layer shall cover the entire universe of corpus query expressions:

Functionality  $\equiv \top$

In other terms, every individual  $x$  shall be a member of some functionality  $F$ .

## 5.5 Frames

Frames represent typical search needs at an intermediate degree of abstraction. They involve more or less complex combinations of functionalities. All frames shall be subsumed by the abstract root concept of the layer:

— **Frame**  $\sqsubseteq$  ExpressivePower

Every frame  $A$  shall make additional subsumption assertions towards one or more functionalities  $F_i$  and/or other frames  $A_j$ :

$$A \sqsubseteq \text{Frame}$$

$$A \sqsubseteq F_1 \sqcap F_2 \sqcap \dots \sqcap F_n \sqcap A_1 \sqcap \dots \sqcap A_k$$

These assertions shall ensure that a direct or indirect subsumption relation holds between  $A$  and a given functionality  $F$  if and only if  $F$  is a substantial element of the search need represented by  $A$ . Likewise, a direct or indirect subsumption relation can hold between  $A$  and another Frame  $A'$  if  $A$  extends the search need represented by  $A'$ .

Each Frame  $A$  shall be given a short descriptive label (as **rdfs:label** annotation), which can be written in an abstract, formal notation, as well as a clear human-readable description of the search need (as **cqlf:searchNeed** annotation).

The extension of a frame  $A$  is the set of all parameterized query expressions  $x$  that satisfy the search need of some use case that instantiates  $A$  (see 5.6). The frame layer shall cover the entire universe of corpus query expressions:

$$\text{Frame} \equiv \top$$

In other terms, every individual  $x$  shall be a member of some frame  $A$ .

NOTE Some frames can involve only a single functionality. Such simple frames describe typical specializations of the functionality, e.g. frames such as PrefixSuffixMatch and RegularExpressionMatch for the PartialMatch functionality. Most frames will be complex, though, i.e. they combine the capabilities of multiple functionalities or the search needs of multiple other frames.

See [Annex A](#) for some illustrative examples of frame specifications.

## 5.6 Use cases

Use cases are instantiations of frames. Each use case shall represent a concrete parameterized search need that is sufficiently specific so that it can be fully satisfied by a parameterized query expression (in any CQL that conforms with the use case).

All use cases shall be subsumed by the abstract root concept of the layer:

— **UseCase**  $\sqsubseteq$  ExpressivePower

Every use case  $U$  shall also make a subsumption assertion towards the frame  $A$  that it instantiates:

$$U \sqsubseteq \text{UseCase}$$

$$U \sqsubseteq A$$

$U$  can make further subsumption assertions to other frames  $A_i$  but should not do so in most cases.

In order for  $U$  to instantiate the frame  $A$ , it shall involve all capabilities required by the search need of  $A$  in a non-trivial manner and combine them in the same way as  $A$ .  $U$  shall not involve any substantial further capabilities or combinations.

**EXAMPLE** Frame  $A$  represents a search need that can be paraphrased as “find two objects  $A$  and  $B$  with specific annotation values at a certain linear distance”. The use case  $U =$  “find word form ① followed by part-of-speech tag ② at a distance of exactly ③ tokens” is considered an instantiation of  $A$ . However, the use case  $U_1 =$  “find word form ① followed by some other token at a distance of exactly ③ tokens” is not an instantiation of  $A$  because it omits the annotation constraint on the second token;  $U_2 =$  “find word form matching regular expression ① followed by part-of-speech tag ② at a distance of exactly ③ tokens” is not an instantiation of  $A$  because it adds the capability of regular expression matching (i.e. the `PartialMatch` functionality), which is not involved in the search need  $A$ .

Each use case  $U$  shall be given a descriptive label (as `rdfs:label` annotation) and a clear extended description of the search need (as `cqlf:searchNeed` annotation). Any variable element of the search need shall be indicated by a parameter placeholder in the label, using a Unicode character in the range U+2460 (CIRCLED DIGIT ONE) to U+2473 (CIRCLED NUMBER TWENTY). All parameters and their permissible values shall be described as part of the `cqlf:searchNeed` annotation.

The extension of a use case  $U$  is the set of all parameterized query expressions that satisfy the search need of  $U$ . The use case layer shall cover the entire universe of corpus query expressions:

$$\text{UseCase} \equiv \top$$

In other terms, every individual  $x$  shall be a member of some use case  $U$ .

See [Annex A](#) for some illustrative examples of use case specifications.

## 5.7 CQLs

Concepts in the CQL taxonomy represent individual CQLs whose expressive power is described with respect to the ontology.

Every CQL  $L$  shall be subsumed by the abstract root concept of the taxonomy:

— **CQL**

$$L \sqsubseteq \text{CQL}$$

No further subsumption assertions shall be made, resulting in a flat taxonomy of independent concepts.

Each CQL  $L$  shall be identified by the name or abbreviation it is commonly associated with (as `rdfs:label` annotation). It shall also be given a description including precise version information and a reference to documentation of the CQL syntax (as `cqlf:description` annotation).

The extension of a CQL  $L$  is the set of all parameterized query expressions formulated in  $L$ . The CQL taxonomy shall cover the entire universe of corpus query expressions:

$$\text{CQL} \equiv \top$$

In other terms, every individual  $x$  shall be a member of some CQL  $L$ .

## 6 Conformance statements

### 6.1 Positive conformance statements

The individuals  $x$  in the A-Box of a CQLF ontology are parameterized query expressions. Every individual  $x$  shall make concept assertions towards the CQL  $L$  in which it is formulated and the use case  $U$  it satisfies:

$$x \in L \sqcap U$$

In exceptional cases, assertions can be made for multiple CQLs and use cases. No concept assertions shall be made to any other concepts in the ontology.

The individual  $x$  makes a positive conformance statement that the expressive power of  $L$  encompasses use case  $U$ , supported by the parameterized query expression as concrete evidence.

NOTE 1 Positive conformance statements are only allowed with respect to use cases, which are required to be specific enough so that a concrete query expression can be formulated that satisfies the use case in its entirety. Frames are too general for positive conformance statements: it is possible that a CQL can satisfy some but not all instantiations of a frame.

Each individual  $x$  shall have the following annotations:

- **rdfs:label**: The parameterized query expression, which shall use the same parameter placeholders as corresponding parameters in the satisfied use case  $U$ . The query expression shall be formulated in such a way that parameter values can be inserted by string substitution; any transformations required shall be described in the **cqlf:parameters** annotation.

NOTE 2 Parameter placeholders are Unicode characters in the range U+2460 (CIRCLED DIGIT ONE) to U+2473 (CIRCLED NUMBER TWENTY), see 5.6.

- **cqlf:parameters**: Detailed information on permissible values for each parameter as well as possible transformations required before substituting the parameter in the query expression.

EXAMPLE For the parameterized CQP<sup>[6]</sup> query expression [lemma = "①"], typical examples of information that can be provided in the **cqlf:parameters** annotation are that parameter ① is a regular expression in PCRE (Perl compatible regular expressions) syntax, that it is automatically anchored at the start and end of the annotation string, and that double quotes (QUOTATION MARK) shall be escaped by reduplication (i.e. by substituting " with "").

- **cqlf:example**: A fully realized example of the query expression with all parameters substituted by arbitrarily chosen values, so that it can directly be executed in an implementation of CQL  $L$ .

Further information and comments about the parameterized query expression can be provided as **rdfs:comment** annotation.

See [Annex A](#) for some illustrative examples of positive conformance statements.

## 6.2 Negative conformance statements

Negative conformance statements document unsupported functionalities  $F$  of a CQL  $L$ , as well as known design limitations which make it impossible to satisfy a particular frame  $A$  or use case  $U$ . Formally, they assert that  $L$  and  $F$  (or  $A$  or  $U$ , respectively) are disjoint concepts.

If  $L$  does not have the capability represented by a functionality  $F$ , the following assertion shall be made:

$$L \sqcap F \equiv \perp$$

If  $L$  cannot support any instantiation of a frame  $A$  because of its design limitations, the following assertion shall be made (unless it is already implied):

$$L \sqcap A \equiv \perp$$

If  $L$  is known not to satisfy a use case  $U$  in its entirety, the following assertion shall be made (unless it is already implied):

$$L \sqcap U \equiv \perp$$

As a logical consequence of the negative conformance statement  $L \sqcap U \equiv \perp$ , it is impossible for  $L$  to satisfy any frame or use case subsumed by  $F$ . Explicit negative conformance statements against such frames and use cases are redundant and should not be made. As a logical consequence of  $L \sqcap A \equiv \perp$ , it is impossible for  $L$  to satisfy any use case instantiating  $A$ , and explicit negative conformance statements against such use cases should not be made. If  $L \sqcap U \equiv \perp$  (whether stated explicitly or implied), there can

never be a positive conformance statement  $x \in L \sqcap U$  because it would create a logical inconsistency in the ontology.

If there is neither positive nor negative conformance between a CQL  $L$  and a use case  $U$ , it shall be considered undetermined whether or not  $F$  satisfies  $U$ . The description of any CQL  $L$  with respect to a CQLF ontology should aim to cover all use cases, either in the form of an explicit positive or negative conformance statement or via negative conformance implied by subsumption.

NOTE The lack of a positive conformance statement between  $L$  and  $U$  indicates either that  $L$  does not satisfy  $U$  or that  $L$  satisfies  $U$  but this fact has not been documented yet in the ontology. The latter situation is likely to arise in a community process when further frames and use cases are added after the initial documentation of CQL  $L$ . Therefore, the lack of a positive conformance statement cannot be interpreted as negative conformance.

STANDARDSISO.COM : Click to view the full PDF of ISO 24623-2:2021