ISO/CD 24623-2:2020(E)

ISO TC 37/SC 4/WG 6

Secretariat: KATS

Language resource management — Corpus query lingua franca (CQLF) — Part 2: Ontology

CD stage

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

Published in Switzerland

Contents

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).

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).

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.

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.

Introduction

Technical Committee ISO/TC 37, *Language and Terminology*, Subcommittee SC 4, *Language Resource Management* has developed several families of standards codifying various aspects of representation of language data. These standards describe general corpus-oriented data models in the Linguistic Annotation Framework (LAF, ISO 24612) family, various aspects of the semantic representation in the family of the Semantic Annotation Framework (SemAF, ISO 24617-1 [2] and others), the representation of lexical data in the Lexical Markup Framework family (LMF, ISO 24613-1 [1] and others), as well as the representation of metadata in the Component Metadata Infrastructure (CMDI, ISO 24622-1 [3] and others). Complementary to the standards concerning the representation of language data, the Corpus Query Lingua Franca (henceforth CQLF) family of standards focuses on the exploitation of language data and ways to satisfy various kinds of information needs targeting these data.

The CQLF Metamodel, described by Part 1 of the standard series (CQLF-1, 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 CQLF-1 levels and modules) as well as the dependencies among them.

Consequently, the task of a more concrete characterization of CQLs falls to other members of the CQLF standard family. This document (ISO 24623-2, "CQLF-2" for short) establishes an ontology that focuses on the generalized information needs satisfied by corpus queries, in the form of a multi-layer taxonomy against which individual CQLs can make positive and negative conformance statements.

Establishing this ontology allows, on the one hand, a fine-grained comparison of the expressive power of CQLs, and, on the other hand, it is going to serve a practical purpose: as a foundation for a platform 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.

Language resource management — Corpus query lingua franca (CQLF) — Part 2: Ontology

1 Scope

This document defines an ontology for fine-grained description of the expressive power of CQLs in terms of search needs. The ontology consists of three interrelated taxonomies of concepts: a) the CQLF Metamodel (a formalization of CQLF-1), b) the Expressive Power taxonomy, which describes different facets of the expressive power of CQLs, and c) a taxonomy of CQLs.

The normative parts of this document comprise a) the taxonomy of the CQLF Metamodel, b) the Functionality layer of the Expressive Power taxonomy, c) the structure of the layers of the Expressive Power taxonomy and the relationships between them, in the form of subsumption assertions, as well as 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 provide a normative listing of the middle and bottom layer of the Expressive Power taxonomy (called Frames and Use Cases, respectively). An exhaustive inventory of the concepts at these two layers is not possible due to the fact that CQLs differ widely in the complexity of the supported combinations of Functionalities and that new CQLs can be created offering additional combinations. Frames and Use Cases are expected to be filled in through a moderated community process, driven by CQL developers as well as end users. An informative annex to this document contains a sample of Frames and Use Cases together with conformance statements linking them with the CQP [5] and ANNIS [7] query languages.

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/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*

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

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

Motik, B., Patel-Schneider, Peter F., and Parsia, B. (2012). *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation, 11 December 2012. (Latest version available at 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 http://www.electropedia.org/

3.1
CQLF module
subcomponent of a CQLF level, defined with reference to a specified data model characteristic

Note 1 to entry: 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).

[SOURCE: ISO 24623-1:2018 [5], 3.8]

3.2
Functionality
label for a concept in the CQLF Ontology that represents a family of capabilities contributing to the expressive power of a CQL, formulated at a general level and linked to one or more CQLF modules

3.3
Frame
label for a concept in the CQLF Ontology that represents a typical search need of end users, understood as one facet of the expressive power of CQLs

Note 1 to entry: Most Frames arise from the specialization of a Functionality and/or the combination of multiple Functionalities.

3.4
Use Case
label for a concept in the CQLF Ontology that represents a concrete instantiation of a Frame, for which it can be determined unambiguously whether a given query expression satisfies the search need 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 (or user agents) to address parts of those data models

Note 1 to entry: A CQL defines a syntactic notation for query expressions 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 [5], 3.4, modified – "user" was replaced with "end user" in the definition and Note 1 to entry was added.]

3.6
search need
information pattern that an end user 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 to satisfy his or her search needs

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

3.8
query expression
string that is syntactically valid in a given CQL 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 or in the description of a search need

3.10
positive conformance statement
assertion that a given CQL supports a given Use Case by means of a query expression

3.11
negative conformance statement
assertion that a given CQL cannot support a given Use Case, Frame or Functionality

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.

3.12
CQL capability
corpus query language capability
facility provided by CQLs to meet a specific aspect of search needs

3.13
layer
totality of concepts at the same level of abstraction in the CQLF Ontology

EXAMPLES: Functionalities, Frames, Use Cases

3.14
token
non-empty contiguous sequence of graphemes or phonemes in a document

[SOURCE: ISO 24611:2012, 3.21, modified — The note was deleted.]

4 Motivation and aims (informative)

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 of their combination – that determine the expressive power of a CQL. The CQLF Ontology 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.

Yet another important aspect is the degree of explication that a CQL delivers. Some CQLs can be able to express a particular search need in a condensed and highly specialized manner, while others rely on complex combinations of a few elementary capabilities. The CQLF Ontology leverages information from CQLs with a more explicit formalization of capabilities in order to create a systematic taxonomy of search needs and thus be able to classify CQLs of rather implicit formalization. Their respective degree of explication is visible in the parameterized query expressions included in all positive conformance statements.

This document defines the structure of an ontology representing CQL capabilities and search needs in a taxonomy consisting of three layers of varying degrees of abstraction, as well as the conformance of individual CQLs to this central taxonomy.

End users navigate the taxonomy starting from a compact layer of CQLF capabilities. Selecting a subset of relevant capabilities allows them to locate relevant search needs in the middle layer of the taxonomy efficiently, and then choose a concrete instantiation of the search need that is closest to their requirements in the bottom layer. This instantiation links to all CQLs that satisfy the selected search need and provides a parameterized query expression for each CQL.

The definition of the structure of the CQLF Ontology as described in this document is expected to be instantiated and expanded in a dynamic community-based project (see Annex B). The permissive architecture and terminology defined by CQLF-2 enables research groups to extend the relevant parts of the ontology with further CQL capabilities and search needs.

CQLF-2 is primarily intended for the following application scenarios:

- describing the scope and capabilities of a given CQL, in terms of conformance statements against the CQLF Ontology (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; and
- guiding the development of new CQLs and query tools by building an inventory of complex search needs that are important for the community (by end users).

5 CQLF Ontology

5.1 OWL DL formalism

The taxonomic framework of the CQLF Ontology is modelled in OWL 2 DL [6] – a dialect of the Web Ontology Language (OWL) based on the family of description logics (hence DL, see [8]) as a formal framework. All definitions and requirements of the OWL 2 Specification shall be followed. The normative representation and exchange format for the CQLF Ontology is RDF/XML [9, 10]. All labels and annotations shall be represented as sequences of Unicode code points, following ISO/IEC 10646.

OWL 2 DL furnishes developers with a set of tools for a) stating concept hierarchies and membership of individuals and b) defining highly expressive property restrictions. In particular, the CQLF Ontology 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; a link to the complete RDF/XML serialization of the normative part of the ontology can be found in Clause 7.

Before turning to the DL specification of the CQLF Ontology, a few relevant DL notions will be introduced [8]:

- **concept inclusion** ⊑: 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.
- **concept equivalence** ≡: 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** ⊓: This operator denotes the intersection of two concept expressions, i.e. the individuals contained in both concept expressions.
  Note: $A \sqsubseteq B \sqcap C$ 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** ⊔: This operator denotes the union of two class expressions, i.e. the individuals contained in either or both of the concept expressions.
- **top concept** ⊤: denotes the set of all individuals in the domain, i.e. the entire universe. Thing, the root class.
- **bottom concept** ⊥: denotes the empty set of individuals in the domain. Nothing, the empty class.
- **concept assertion** ∈: This operator asserts that an individual belongs to a concept. Also known as class assertion because 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 the 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 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 several 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 by CQLF-1 (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 CQLF 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 it 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 the CQLF Ontology is illustrated in Figure 1.

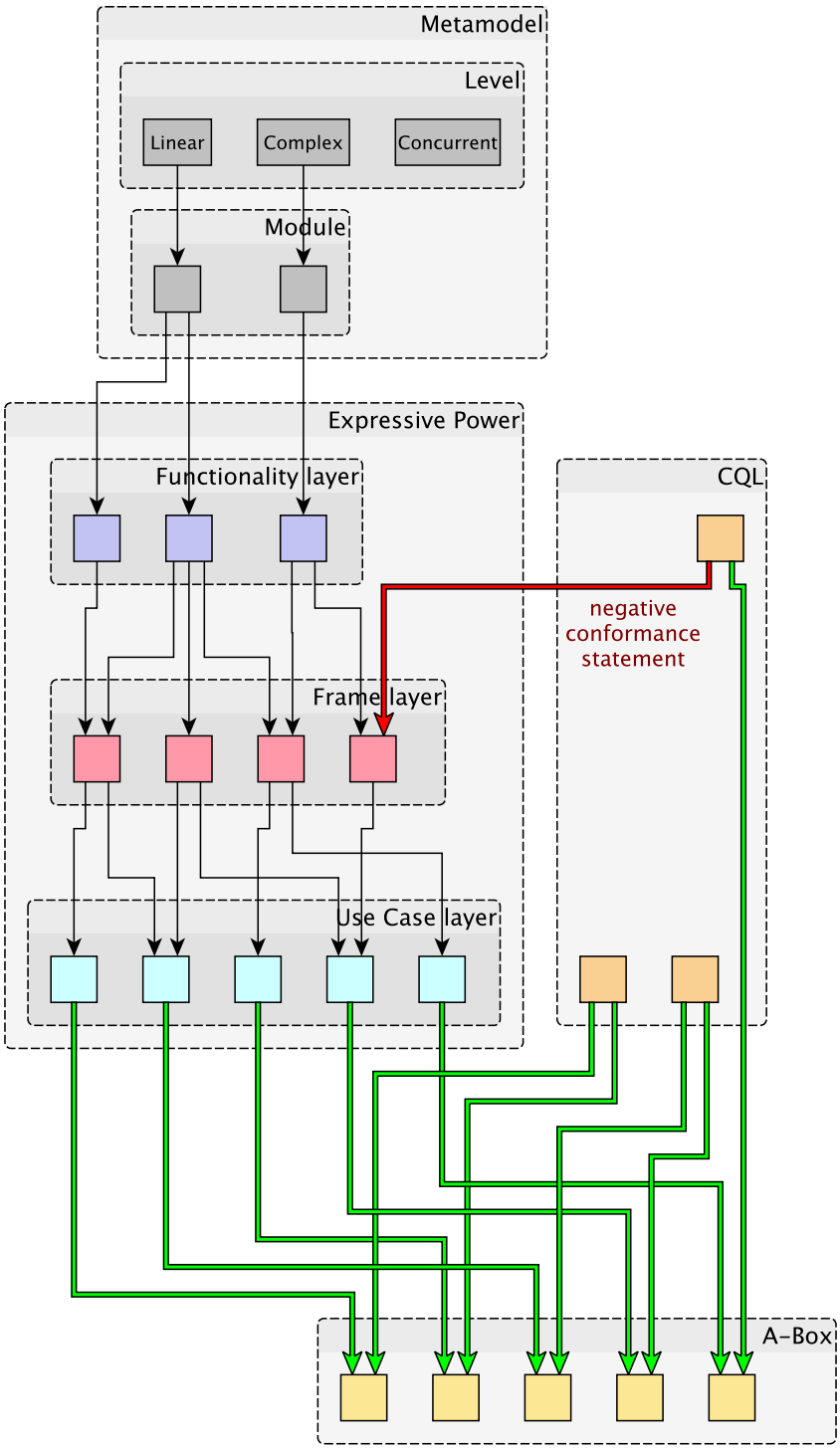

Figure 1 — General structure of the CQLF Ontology

5.3 CQLF Metamodel

The theoretical concept of modules as developed by [4] and standardized in the context of CQLF-1 (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**
  NOTE 1: This is the abstract root concept of the CQLF Metamodel taxonomy.
- **Level** ⊑ Metamodel
  NOTE 2: This is the abstract root concept of all CQLF levels.
- **Module** ⊑ Metamodel
  NOTE 3: This is the abstract root concept of all CQLF modules.

CQLF levels:

- **Linear** ⊑ Level
  NOTE 4: plain-text search as well as search in segmented data
- **Complex** ⊑ Level
  NOTE 5: search in data annotated with hierarchical structures and/or dependency information, or querying simple annotations by means of containment-based queries
- **Concurrent** ⊑ Level
  NOTE 6: search in multiple concurrent (overlapping, intersecting and often conflicting) annotations built upon a single data stream

CQLF modules:

- **PlainText** ⊑ Module ⊓ Linear
  NOTE 7: segmentation-independent string search
- **SimpleAnnotation** ⊑ Module ⊓ Linear
  NOTE 8: segmentation-based search for annotations describing primary data stream; understood more generally as search for annotations of individual objects in the context of CQLF-2
- **Segmentation** ⊑ Module ⊓ Linear
  NOTE 9: search for segmental annotation, in particular tokens and token sequences
- **Hierarchical** ⊑ Module ⊓ Complex
  NOTE 10: tree-based representations, e.g. for phrase-structure description
- **Dependency** ⊑ Module ⊓ Complex
  NOTE 11: identification of relationships in which objects function as nodes linked by directed arcs
- **SpanContainment** ⊑ Module ⊓ Complex
  NOTE 12: non-recursive simplified hierarchical relationships encoded as character span containment
- **Paradigmatic** ⊑ Module ⊓ Concurrent
  NOTE 13: different annotation layers provide data packages describing the same location
- **Overlapping** ⊑ Module ⊓ Concurrent
  NOTE 14: 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 rather than CQLF modules should be used as entry points for navigation of the ontology.

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. For this reason, CQLF modules are not a suitable entry point for navigation of the ontology.

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** ⊑ ExpressivePower

Functionalities:

- **Annotation** ⊑ Functionality
  find individual objects based on their linguistic annotation
  Annotation ⊑ SimpleAnnotation ⊔ Paradigmatic
- **ConstraintCombination** ⊑ Functionality
  Boolean operators for combining constraints on objects
  ConstraintCombination ⊑ Metamodel
  EXAMPLE 1: conjunction (&), disjunction (|), negation (!, !=), difference (−)
- **Containment** ⊑ Functionality
  containment of an object in a specific context
  Containment ⊑ SpanContainment ⊔ Overlapping
- **ExternalResource** ⊑ Functionality
  reference to externally encoded structured data
  ExternalResource ⊑ Metamodel
  EXAMPLE 2: reference from annotation object to dictionary, database of speakers, etc.
- **FuzzySearch** ⊑ Functionality
  approximate string matching
  FuzzySearch ⊑ Segmentation ⊔ PlainText ⊔ SimpleAnnotation
- **GraphRelation** ⊑ Functionality
  relationships of objects as nodes linked by directed arcs
  GraphRelation ⊑ Dependency
- **LinearRelation** ⊑ Functionality
  horizontal relationships of objects
  LinearRelation ⊑ Segmentation ⊔ PlainText
  EXAMPLE 3: precedence/co-occurrence of objects
- **MatchingStrategy** ⊑ Functionality
  define specific matching modality
  MatchingStrategy ⊑ Metamodel
  EXAMPLE 4: match the first object in a sequence, match the whole span between two objects, greedy vs. non-greedy search
- **Metainformation** ⊑ Functionality
  query metainformation associated with the primary data

Metainformation ⊑ Metamodel
EXAMPLE 5: text genre, publication date, author sex

- **PartialMatch** ⊑ Functionality
match plain text or linguistic annotation value against a generalized pattern
PartialMatch ⊑ PlainText ⊔ SimpleAnnotation
EXAMPLE 6: prefix/suffix search, match against regular expression
- **PlainTextSearch** ⊑ Functionality
find segmentation-independent strings
PlainTextSearch ⊑ PlainText
EXAMPLE 7: "[...]ad a little la[...]", "[...]mber 47 said to num[...]"
- **Position** ⊑ Functionality
relative position of an object with respect to another
Position ⊑ PlainText ⊔ Segmentation ⊔ SpanContainment ⊔ Hierarchical ⊔ Dependency
EXAMPLE 8: containment at left, distance between two objects
- **Quantification** ⊑ Functionality
numeric range specification on a constraint
Quantification ⊑ Metamodel
EXAMPLE 9: at least *N*, at most *M*
- **Repetition** ⊑ Functionality
multiple occurrence of an object
Repetition ⊑ PlainText ⊔ Segmentation
- **Sensitivity** ⊑ Functionality
treatment of special character features
Sensitivity ⊑ PlainText ⊔ Annotation
EXAMPLE 10: case-insensitive, ignore diacritics
- **Size** ⊑ Functionality
specify the size of an object
Size ⊑ PlainText ⊔ Segmentation ⊔ Hierarchical ⊔ Dependency
EXAMPLE 11: string length, number of objects in sequence, arity of tree branch or graph vertex
- **TreeRelation** ⊑ Functionality
vertical relationships of objects as nodes linked by hierarchically directed arcs
TreeRelation ⊑ 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:

$$\text{Functionality} \equiv \top$$

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** ⊑ ExpressivePower

Every Frame *A* shall make additional subsumption assertions towards one or more Functionalities F_i and/or other Frames A_j:

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

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

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 5.5). 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** ⊑ 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 corpus query languages whose expressive power is to be described with respect to the CQLF 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 the 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: a CQL might 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 5.5.

- **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 1 For the parameterized CQP [5] query expression `[lemma = "①"]`, the cqlf:parameters annotation might state that parameter ① is a regular expression in PCRE 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.

EXAMPLE 2 See Annex A for some illustrative examples of positive conformance statements.

6.2 Negative conformance statements

Negative conformance statements document missing capabilities *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 assertion

$$L \sqcap F \equiv \bot$$

shall be made.

If *L* cannot support any instantiation of a Frame *A* because of its design limitations, the assertion

$$L \sqcap A \equiv \bot$$

shall be made (unless it is already implied).

If L is known not to satisfy a Use Case U in its entirety, the assertion

$$L \sqcap U \equiv \bot$$

shall be made (unless it is already implied).

NOTE 1 As a logical consequence of the negative conformance statement $L \sqcap F \equiv \bot$, 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 $L \sqcap A \equiv \bot$, 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 $L \sqcap U \equiv \bot$ (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 indeterminate whether or not F satisfies U. The description of any CQL L with respect to the 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 2 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.

7 RDF/XML serialization (informative)

A complete RDF/XML serialization of the normative part of the CQLF Ontology is available on GitHub at the following URL:

https://github.com/cqlf-ontology/cqlf/blob/master/templates/CQLF-2.owl

Annex A
(informative)

Illustrative examples of non-normative elements in the CQLF Ontology

A.1 Example ontology

A larger example ontology is available on GitHub at the following URL:

https://raw.githubusercontent.com/cqlf-ontology/cqlf/master/examples/CQLF-2.owl

The ontology fragment detailed below is visualized in Figure B.1 at the end of this annex.

A.2 Frames

A.2.1 $A_1 \sqsubseteq$ Frame $\sqcap$ Annotation $\sqcap$ TreeRelation

- rdfs:label = Annotation(TreeRelation)
- cqlf:searchNeed = tree relation with a specified annotation value

A.2.2 $A_2 \sqsubseteq$ Frame $\sqcap$ Annotation $\sqcap$ PartialMatch

- rdfs:label = RegEx(Annotation)
- cqlf:searchNeed = object with annotation value matching regular expression

A.2.3 $A_3 \sqsubseteq$ Frame $\sqcap A_1 \sqcap A_2$

- rdfs:label = RegEx(Annotation(TreeRelation))
- cqlf:searchNeed = tree relation with partially matched annotation value

A.2.4 $A_4 \sqsubseteq$ Frame $\sqcap$ Annotation

- rdfs:label = Annotation(Object)
- cqlf:searchNeed = annotation object where a given attribute has a specific value

A.2.5 $A_5 \sqsubseteq$ Frame $\sqcap$ TreeRelation

- rdfs:label = Domination(Object, Object)
- cqlf:searchNeed = dominance relation between two objects in a tree

A.2.6 $A_6 \sqsubseteq$ Frame $\sqcap A_3 \sqcap A_4 \sqcap A_5$

- rdfs:label = RegEx(Annotation(Domination))(Annotation(Object), Annotation(Object))
- cqlf:searchNeed = domination relation with functional annotation matched by regular expression between two tree nodes with specific annotation values (single attribute=value constraints)

A.3 Use Cases

A.3.1 $U_1 \sqsubseteq$ UseCase $\sqcap A_6$

- rdfs:label = immediate dominance matching regex ① between phrase node of category ② and token with POS tag ③
- cqlf:searchNeed = find a phrase node *A* of category ② in a syntactic parse tree and a token *B* with part-of-speech tag ③ such that *A* is the immediate parent of *B* and the dominance relation is annotated with a function matching regular expression ①

A.4 CQLs

A.4.1 CQP $\sqsubseteq$ CQL

- rdfs:label = CQP
- cqlf:description = Query syntax of the corpus query processor (CQP) of IMS Open Corpus Workbench version 3.5, see https://cwb.sourceforge.net/. Documentation is provided by the CQP Query Language Tutorial at https://cwb.sourceforge.net/files/CQP_Tutorial.pdf.

A.4.2 ANNIS $\sqsubseteq$ CQL

- rdfs:label = ANNIS
- cqlf:description = ANNIS Query Language (AQL) version 3.6, see https://corpus-tools.org/annis/. Documentation is provided by the ANNIS User Guide available at https://korpling.github.io/ANNIS/3.6/user-guide/.

A.5 Conformance statements

A.5.1 Positive conformance statement for ANNIS

$x_1 \in$ ANNIS $\sqcap U_1$

- rdfs:label = `cat = "②" & pos="③" & #1 >[func=/①/] #2`
- cqlf:parameters =
  - ① is a basic regular expression, which is automatically anchored at the beginning and end of the annotation value
  - forward slashes in ① shall be escaped by backslashes
  - double quotes in ② and ③ shall be escaped by backslashes
- cqlf:example = `cat = "NP" & pos="ADJA" & #1 >[func=/NK.*/] #2`
- rdfs:comment = The precise flavour of regular expression syntax is not specified in the ANNIS documentation and can be implementation-specific. It is assumed that all elements of POSIX.1 Basic Regular Expression syntax are supported.

A.5.2 Negative conformance statement for CQP

CQP $\sqcap$ TreeRelation = $\bot$

NOTE CQP does not implement any capabilities corresponding to the TreeRelation Functionality. As a consequence, it cannot satisfy any Frames or Use Cases subsumed by TreeRelation, namely A_1, A_3, A_5, A_6 and U_1 in the present fragment.

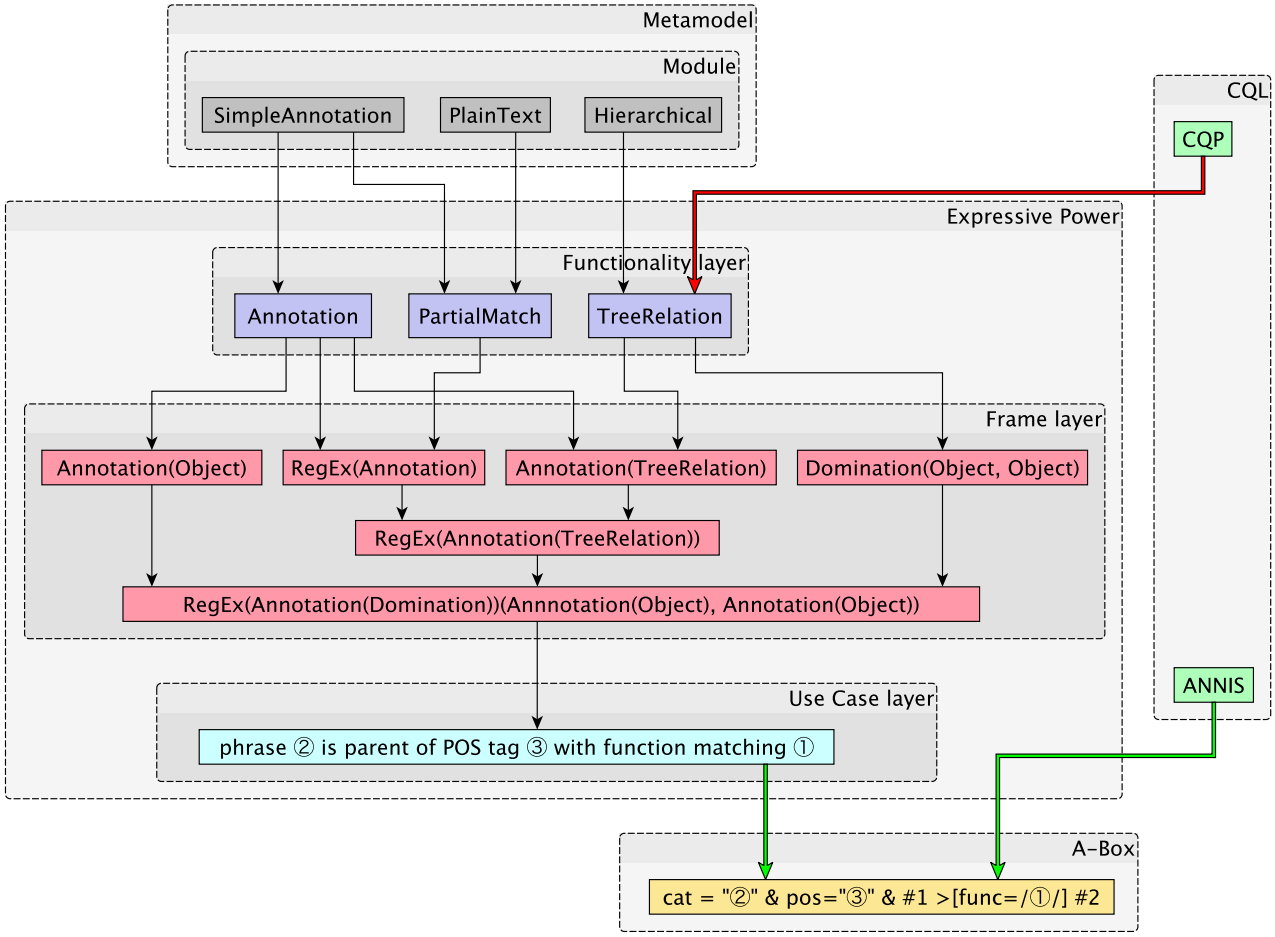

Figure B.1 — An example fragment of non-normative content in the CQLF Ontology

Annex B
(informative)

CQLF Ontology: Moderated community process

Recall that only the top layer of the Expressive Power taxonomy of the CQLF Ontology is in the normative scope of this document. The lower layers of Frames and Use Cases – as well as conformance statements for different CQLs – are expected to be supplied by the community in a moderated process of extending the ontology. As an informative annex to this document, an initial version of the extended ontology, documenting the CQP [5] and ANNIS [7] CQLs, is provided by a GitHub organization at the following URL:

https://github.com/cqlf-ontology/

The community is expected to extend the ontology with new Frames and Use Cases but will not be able to modify or retract existing ones. The list below enumerates the main features of the envisioned community process, on the assumption that GitHub will be used as the platform.

- Authentication:
  - via GitHub, the ontology can only be edited directly by users whose accounts are members of the GitHub organization cqlf-ontology;
  - other users can submit pull requests, e.g. with conformance statements for a new (or newly added) CQL, or with entries for new Frames and Use Cases.
- Version control: all submissions are automatically recorded together with a date stamp and the account name of the submitter.
- Moderation: members of the GitHub organization cqlf-ontology review pull requests and ensure that they meet all requirements laid down in the present document; existing Frames and Use Cases will only be modified or deleted in exceptional circumstances by the moderators.
- It is expected that users ensure the well-formedness of the ontology with their modifications before they initiate a pull request.
- Moderation, error reporting and the verification of the submitted conformance statements will be driven by the ticketing system automatically coordinated with pull requests.

Bibliography

[1] ISO 24613-1, *Language resource management — Lexical markup framework (LMF) — Part 1: Core model*

[2] ISO 24617-1, *Language resource management — Semantic annotation framework (SemAF) — Part 1: Time and events (SemAF-Time, ISO-TimeML)*

[3] ISO 24622-1, *Language resource management — Component Metadata Infrastructure (CMDI) — Part 1: The Component Metadata Model*

[4] Bański, P., Frick, E., and Witt, A. (2016). Corpus query lingua franca (CQLF). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation* (LREC 2016), pages 2804–2809, Portorož, Slovenia. European Language Resources Association (ELRA).

[5] Evert, S. and Hardie, A. (2011). Twenty-first century corpus workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 Conference*, Birmingham, UK.

[6] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P., and Rudolph, S. (2012). OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, 11 December 2012. (Latest version available at http://www.w3.org/TR/owl2-primer/.)

[7] Krause, T. and Zeldes, A. (2016). ANNIS3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities* 31(1): 118–139.

[8] Krötzsch, M., Simancik, F., and Horrocks, I. (2012). A description logic primer. arXiv:1201.4089

[9] Patel-Schneider, P. F. and Motik., B. (2012). OWL 2 Web Ontology Language: Mapping to RDF Graphs (Second Edition). W3C Recommendation, 11 December 2012. (Latest version available at http://www.w3.org/TR/owl2-mapping-to-rdf/.)

[10] Beckett, D. (2014). RDF 1,1 XML Syntax. W3C Recommendation, 25 February 2014. (Latest version available at https://www.w3.org/TR/rdf-syntax-grammar/.)