Published online: 2021-06-30

A Comparison of Arden Syntax and Clinical Quality Language as Knowledge Representation Formalisms for Clinical Decision Support

Andrey Soares[1] Robert A. Jenders[2] Robert Harrison[3] Lisa M. Schilling[1]

[1] Department of Medicine, University of Colorado, Aurora, Colorado, United States
[2] Department of Medicine, University of California, Los Angeles, California, United States
[3] University of Colorado Health (UCHealth), Aurora, Colorado, United States

Address for correspondence Andrey Soares, PhD, Department of Medicine, University of Colorado, 13199 East Montview Boulevard, Suite 210, Aurora, CO 80045, United States (e-mail: andrey.soares@cuanschutz.edu).

Appl Clin Inform 2021;12:495–506.

Abstract

Objectives This article presents a comparative study of two Health Level Seven International (HL7) standards for clinical knowledge representation, the Arden Syntax and the Clinical Quality Language (CQL), regarding their expressiveness and utility to represent knowledge for clinical decision support (CDS) systems.

Methods We compiled a concatenated set of features from both languages and made descriptive comparisons of 27 categories covering areas of language characteristics, data, control statements, and operators.

Results Both Arden and CQL have similar constructs that can be used for representing CDS knowledge but also have unique constructs that could support distinct use cases. They have constructs that fully or partially address several of the categories used in the comparison, except for data models and terminologies in Arden and event triggering and iteration statements in CQL.

Conclusion These standards can facilitate the sharing, management, and reuse of computable knowledge, and permit knowledge to be represented with their languages and converted to a machine-friendly executable code that can be shared and reused by other systems. Having support for standard data models and terminologies will continue to be a differential for adoption of a language. The HL7 working groups responsible for developing these standards can direct future development to enhance the functions of the standard and address the gaps identified in this study.

Keywords

- clinical decision support
- Clinical Quality Language
- Arden Syntax
- comparison
- knowledge representation

Background and Significance

With the passage of the Health Information Technology for Economic and Clinical Health Act (HITECH) of 2010, the United States government authorized incentive payments, referred to as "Meaningful Use" (MU, now Promoting Interoperability), to clinicians and hospitals for the use of electronic health record (EHR) systems to improve clinical care delivery.[1] The HITECH Act specified that MU advance the adoption of EHR functions that included: measuring and

received
January 15, 2021
accepted after revision
May 3, 2021

DOI https://doi.org/10.1055/s-0041-1731001.
ISSN 1869-0327.

Georg Thieme Verlag KG, Rüdigerstraße 14, 70469 Stuttgart, Germany

reporting clinical quality measures; clinical decision support (CDS); electronic prescribing; health information exchange; and patient-tailored health and disease management tools. HITECH, and the substantial monetary resources supporting it, led to several initiatives that resulted in informatics advances driven by a goal to harmonize and develop technical solutions that could be applied to both quality measurement and quality improvement, including CDS interventions. The solutions include: (1) the Center for Medicaid and Medicare Service defined electronic clinical quality measures (eCQMs),[2] (2) the Quality Data Model (QDM) which defines such things as logical operators, criteria for population classifications, and required terminologies and value sets,[3,4] (3) Healthcare Quality Measure Format[5] an extensible markup language (XML) document that represents the data elements, logic, and definitions of the eCQMs based on the QDM model standards, (4) Value Set Authority Center, a repository of value sets used for eCQMS,[6] and (5) Clinical Quality Language (CQL), a programming language to support the creation of executable CDS rules and algorithms.[7]

Other solutions that predated HITECH addressed the need to represent computable clinical knowledge explicitly and in a standard representation to facilitate implementation of CDS and reduce its costs and complexity through knowledge sharing, leading to solutions such as the Arden Syntax for Medical Logic Systems that provides "explicit representation of the data and logic used in clinical reasoning in a standard executable format."[8]

In this article, we provide a comparative analysis study of two Health Level Seven International (HL7) language standards, the Arden Syntax (version 2.10[9]) and the CQL (version 1.4[7]), for regarding their expressiveness and formalism to represent knowledge for CDS systems. We will henceforth refer to them as Arden and CQL. The goal of this article is to provide detailed function-based comparison that will help CDS technologists to better understand their options, and the pros and cons of each standard. Our goal is not to provide a recommendation about what standard to choose. This choice is a complex decision that goes beyond the standards and may include several factors such as organizational culture, resources availability, and project scope. Thus, we provide facts and our analysis about the standards, which can assist stakeholders involved in CDS development make an informed decision about adopting a language for either a specific project or their CDS system.

Objectives

We undertook the present study to identify, characterize, and compare Arden and CQL in regards to their utility for representing CDS knowledge resources (i.e., artifacts).[10] We focus on constructs and characteristics that support the representation of clinical knowledge (statements about the clinical domain such as laboratory results) and control knowledge (instruction on how to apply the clinical knowledge).[11]

Methods

We analyzed the Arden and CQL specification documents to identify programming constructs[12] (e.g., sequence, selection and repetition, procedure, operators, and data types) and specific language characteristics. We compiled a concatenated set of 27 categories (►Table 1) and provided a description of the categories and identified snippets of code to illustrate the explanations comparing the languages. We provided references where additional sample use cases can be found: the HL7 Arden Syntax Implementation Guide, Release 3[13] and the "Cooking with CQL series."[14]

Results

We report on our characterized Arden and CQL constructs and our comparisons organized by category. A list of constructs used for the comparison is summarized in ►Table 2.

Language Characteristics

Grammar: Arden and CQL each use a context-free grammar expressed in the Backus–Naur form[15] or an extension of it (i.e., ANother Tool for Language Recognition[16]) and are intended to express medical knowledge in a English-like format to allow users to read and understand the logic represented in the artifact.

Machine-friendly executable: Both standards support translation of the logic into a machine-readable artifact (i.e., Arden Syntax Markup Language [ArdenML] and Expression Logical Model [ELM] XML) that can be shared and used by other systems such as Drools.[17]

Code execution: Code execution in Arden is executed sequentially (top to bottom) according to the evaluation steps of the slots (i.e., *data*, *evoke*, *logic*, *action*). Data read in the *data* slot is available in subsequent slots. The expressions in the *logic* slot are executed in sequence, and the execution can end at any time using the *Conclude* statement. If no *Conclude* statement is executed, the execution ends after the last statement in the *logic* slot, and the *action* slot is not executed.[9] In CQL, a preprocessing phase reads the initial data to be processed, followed by an evaluation phase that evaluates the expressions.[7] Declarations are executed in sequence but the *define* and *function* statements can be written in any sequence within a context (i.e., *Patient*, *Practitioner*, or *Unfiltered*) as they are executed based on their interdependencies.

Modularization: Like in software engineering, modularization is an important step to manage complexity and to separate the functionality of a program into independent modules. Both Arden and CQL have an *include* statement to support modularization (►Fig. 1), but they have different approaches to it. In Arden, code can be separated into different Medical Logic Modules (MLM) files that can then be called from other MLMs. Each functionality would be stored on its own MLM file, which can receive parameters with data to be processed or use the *data* slot to access the data needed. The *interface* statement also allows calling external functionalities created with other languages. In

Table 1 Feature categories and description used for comparison

Language characteristics
• **Grammar**: Support a natural language-like expressions to facilitate understanding of the logic • **Machine-friendly executable**: Convert the logic of artifacts to be used by other systems • **Code execution**: Support production rule and procedural knowledge • **Modularization**: Separate complex logic into subunits • **Event triggering**: Monitor actions and conditionally launch knowledge execution • **Maintenance / Library**: Documentation about the purpose, creation, and maintenance of the artifacts
Data
• **Data types**: Covering a variety of data types • **Data models**: Facilitate use of different data models • **Data context**: Facilitate access to data from a specific patient or provider perspective • **Data conversion**: Convert data types and units from one format to another • **External resources**: Access external data and resources beyond the patient database • **Terminologies**: Access to standard vocabularies and ontologies
Control statements
• **Conditional statements**: Control statements for conditions • **Iteration statements**: Control statements for repetition
Operators
• **Logical operators**: Evaluation of logical expressions • **Arithmetic/Numeric operators**: Evaluation of numeric expressions • **Comparison operator**: Compare operands and return a logical value • **Date/Time operators - Construction**: Create dates and times • **Date/Time operators - Extraction**: Extract information from dates and times • **Date/Time operators - Arithmetic**: Evaluate arithmetic expressions with dates and times • **Date/Time operators - Duration and Differences**: Calculate duration and differences of dates and times • **List operators**: Perform operations on a list • **Aggregate operators**: Perform mathematical operations • **String operators**: Perform operations on strings • **Query operators**: Retrieve data from data stores • **Object operators**: Create objects and access their attributes • **Uncertainty**: Handle inexact or probabilistic information

Table 2 Comparison between Arden and CQL regarding language characteristics and constructs

Category	Arden Syntax v2.10	CQL v1.4
Language characteristics		
Grammar	Backus-Naur Form (BNF)	Antlr4, an Extended Backus-Naur Form (EBNF)
Machine- friendly executable	ArdenML	Expression Logical Model (ELM)
Code execution	Data slot, Evoke slot, Logic slot, Action slot. - sequential, top to bottom, ends with conclude	Library, Using, Include, Codesystem, Valueset, Code, Concept, Parameter, Context, Define, Function. - preprocessing phase (initial data), evaluation phase (result of expressions) - follow sequence of declarations
Modularization	Call, Mlm, Interface, Event, Include	Define functions, Include libraries
Event triggering	Event, Evoke slot	–
Maintenance / Library	Title, Mlmname, Version, Arden Syntax Version, Institution, Author, Specialist, Date, Validation, Purpose, Explanation, Keywords, Citations, Links	Name, Version

(*Continued*)

Table 2 (*Continued*)

Category	Arden Syntax v2.10	CQL v1.4
Data		
Data types	Boolean, String, List, Null, Number, Time, Duration, Term, Query results, Object, Time-of-day, Day-of-week, Truth value, Fuzzy	Boolean, Integer, Decimal, String, Date, Datetime, Time, Quantity, Ratio, CodeSystem, ValueSet, Code, Concept, Tuple, Interval
Data models	–	multiple data models such as Quality Information and Clinical Knowledge (QUICK), Quality Data Model (QDM), Fast Healthcare Interoperability Resources (FHIR) and Virtual Medical Record (vMR)
Data context	–	Patient, Provider, Unfiltered
Data/Type conversion	As String, As Number, As Time, As Truth value	As String, As Integer, As Code, As Quantity, As <Resource>, Cast..As, ToBoolean(String), ToInteger(String), ToDecimal(Integer), ToDecimal (String), ToQuantity(Decimal), ToQuantity(Integer), ToQuantity(String), ToRatio(String), ToDate (String), ToDate(Datetime), ToDatetime(Date), ToDatetime(String), ToTime(String), ToString (Boolean), ToString(Integer), ToString(Decimal), ToString(Quantity), ToString(Ratio), ToString (Date), ToString(Datetime), ToString(Time), ToConcept(Code), ToConcept(List<Code>), Integer From/To Decimal, Integer From/To Quantity, Decimal From/To Quantity, Date From/To Datetime, Code From/To Concept, Convert..To
External resources	Read, Destination, Interface, Event, Write	External functions
Terminologies	–	Valueset, Codesystem, Code, Concept
Control statements		
Conditional statements	If-then-else, Switch-case	If..then...else, Case..when...then...else
Iteration statements	While loop, For loop	–
Operators		
Logical operators	And, Or, Not	And, Or, Not, Xor, Implies
Arithmetic/Numeric operators	**, +, –, *, /, Ceiling, Floor, Truncate, Round, Abs, Exp, Log, Arcos, Arcsin, Arctan, Cosine, Sine, Tangent, Log10, Int, Sqrt	+, –, *, /, Div, Mod, Ceiling, Floor, Truncate, Abs, – (negate), Round, Ln, Log, Exp, ^, Max, Min, HighBoundary, LowBoundary, Precision, Predecessor, Successor
Comparison operators	=, <>, <, <=, >, >=, After, Before, Ago, Is [not] equal, Is [not] less than, Is [not] greater than, Is [not] less than or equal, Is [not] greater than or equal, Is [not] within... to, Is [not] within... preceding, Is [not] within... following, Is [not] within... surrounding, Is [not] within past, Is [not] within same day as, Is [not] before, Is [not] after, Is [not] in, Is [not] present, Is [not] null, Is [not] boolean, Is [not] number, Is [not] string, Is [not] time, Is [not] time of day, Is [not] duration, Is [not] list, [not] In, Is [not] object, Is [not] <objecttype>, Is [not] fuzzy, Is [not] crisp, Occur [not]	=, !=, >, <, >=, <=, Between, ~, !~, In, Same as, Before, After, Meets before, Meets after, Meets, Overlaps before, Overlaps after, Overlaps, Begins, Included in, Includes, Ends; Starts/Ends: On or, Or on, Less than, More than, Or less, Or more, Within... of, Occurs, During, Same or before, Same or after

Table 2 (*Continued*)

Category	Arden Syntax v2.10	CQL v1.4
	equal, Occur [not] within... to, Occur [not] within... preceding, Occur [not] within... following, Occur [not] within... surrounding, Occur [not] within past, Occur [not] within same day as, Occur [not] before, Occur [not] after, Occur [not] at	
Date/Time operators - Construction	Time, Time of day [of], Time of object [of], Attime, Replace year [of]... with, Replace month [of]... with, Replace day [of]... with, Replace hour [of]... with, Replace minute [of]... with, Replace second [of]... with	Year, Month, Day, Hour, Minute, Second, Millisecond, Timezone offset, Now, Today, TimeOfDay
Date/Time operators - Extraction	Day of week, Extract year, Extract month, Extract day, Extract hour, Extract minute, Extract second	Date from, Time from, Year from, Month from, Day from, Hour from, Minute from, Second from, Millisecond from, Timezoneoffset from
Date/Time operators - Arithmetic	+, –	+, –
Date/Time operators - Duration and Differences	From, Year, Month, Week, Day, Hour, Minute, Second	Between, Difference, Duration, Year(s), Month(s), Week(s), Day(s), Hour(s), Minute(s), Second(s), Millisecond(s)
List operators	, (Comma), Merge, Sort, Add... to... [At...], Remove... from..., Minimum (Min)... from, Maximum (Max)... from, First... from, Last... from, Sublist... elements [Starting at...] From ..., Increase, Decrease, % Increase, % Decrease, Earliest... from, Latest... from	Contains, Distinct, =, Except, Exists, Flatten, First, In, Includes, Included in, [], IndexOf, Intersect, Last, Length, ~, !=, !~, Properly Includes, Properly included in, Singleton from, Skip, Tail, Take, Union
Aggregate operators	Count, Exist, Average, Median, Sum, Stddev, Variance, Minimum (Min), Maximum (Max), Last, First, Any, All, No, Latest, Earliest, Element, Extract Characters..., Seqto, Reverse	Count, Sum, Stddev, Median, Variance, AllTrue, AnyTrue, Avg, GeometricMean, Max, Min, Mode, PopStdDev, PopVariance, Product
String operators	||, Formatted with, String..., Trim, Find in string, Localized, Substring..., Length, Uppercase, Lowercase, Matches, Matches pattern	Combine, +, &, EndsWith, [], LastPositionOf, PositionOf, ReplaceMatches, Split, StartsWith, Substring, Length, Upper, Lower, Matches, SplitOnMatches
Query operators	Where, Nearest... from, Index nearest... from, Index of... from..., At least... [ISTrue|AreTrue] from..., At most... [IsTrue|AreTrue] from..., Slope, Interval	Where, Sources, Let clause, With or Without, Return, Sort
Object operators	New, Dot, Clone, Extract attribute Names..., Attribute... from..., Using	–
Uncertainty	Fuzzy set..., Fuzzified by, Defuzzified..., Applicability [of]...	Uncertainty

Abbreviation: CQL, Clinical Quality Language.

CQL

```
include "10_Year_ASCVD_Risk" version '1.0.0' called ASCVDRisk
```

Arden

```
ASCVDRisk := MLM '10_Year_ASCVD_Risk.mlm' FROM INSTITUTION "my institution";
INCLUDE ASCVDRisk;
…
Result := CALL ASCVDRisk;
```

Fig. 1 Example of *include* and *call* statements to invoke other libraries. Clinical Quality Language (CQL) allows the use of any expressions from the included library, while Arden returns the result of processing the Medical Logic Modules (MLM).

Arden

```
data:
   penicillin_storage := EVENT {store penicillin order};
   cephalosporin_storage := EVENT {store cephalosporin order};
;;
priority: 90;;
urgency: 50;;
evoke:
   penicillin_storage OR Cephalosporin_storage;;
   TODAY ATTIME 15:00 AFTER TIME OF penicillin_storage
   every 2 hours for 1 day starting time of Cephalosporin_storage
```

Fig. 2 Example of setting and evoking an event in Arden.[9]

CQL, code can also be separated into different libraries, which can be called by other libraries. Each library can define multiple functions that can accept parameters as inputs and can be called independently using the assigned name of the included library. For instance, a developer could create a library with various functions to support data conversion.

Event triggering: Arden has dedicated slots to support event triggering, including occurrence of some event (e.g., data stored in the patient database), time delay after an event (e.g., check condition status after a few days of starting a new medication), periodically after an event (e.g., monitor drug reaction every 2 days during the 2-week regiment), constant time trigger (e.g., trigger on Monday at 10 a.m.), and constant periodic time trigger (repeat the execution of a MLM every Monday at 10 a.m. for 5 months).[9] The event is defined in the *data* slot and the condition under which the MLM becomes active is described in the *evoke* slot[9] (►Fig. 2). The *priority* slot determines the order of execution and the *urgency* slot determines the importance of the action of the MLMs.[9] When any of the defined events occur, the MLM is triggered. CQL does not have constructs to support event triggering and it would depend on other services to handle the trigger of CQL procedures.

Maintenance/Library: Arden has slots to specify meta-level information about the artifact's knowledge, creation, and purpose, including *mlmname*, *title*, *version* of the artifact and *arden* version, *author*, *date* of last revision, and *validation*, which refers to whether the artifact is approved for production, research, testing, or is expired (►Fig. 3). In addition, it represents brief information about the *purpose* and *explanation* about the artifact, with *citations* to relevant literature and *links* to external sources of information that

Arden

```
maintenance:
   title:       10-Year ASCVD Risk Assessment;;
   mlmname:     10_Year_ASCVD_Risk;;
   arden:       Version 2.9;;
   version:     1.00;;
   institution: My Health Institution;;
   author:      John Doe, PhD;;
   specialist:  John Doe, PhD;;
   date:        2018-05-17;;
   validation:  testing;;
```

Fig. 3 Example of Arden's Maintenance category.

CQL

```
using FHIR version '4.0.0'

define DOB: Patient.birthDate
```

Arden

```
data:
(X_TOKEN)   := ARGUMENT;
MData       := READ {GetMData where token=X_TOKEN and
                    SearchBackAmount=365 and
                    HoursOrDays=Days};
MedName     := MData.MMeds.Name;
Dose        := MData.MMeds.Dose;
DoseUnit    := MData.MMeds.DoseUnit;
Route       := MData.MMeds.Route;
AdminTime   := MData.MMeds.InstantTaken;
Medication := OBJECT [MedName, Dose, DoseUnit, Route, AdminTime];
```

Fig. 4 Example of Clinical Quality Language (CQL) using a data model and Arden using *object*.

support the artifact. CQL only has meta-information regarding the unique name of the library and its *version*.

Data

Data types: Both standards support a variety of data types, including primitive data types and some special types. They have equivalent types (e.g., *Boolean*, *String*, *List*, and *Null*), and some differences on how they handle numbers, dates, and times. Arden uses a single number type that does not distinguish between integer and floating point numbers, while CQL has separate constructs for *Integer* and *Decimal* numbers. They also have differences regarding *Date* and *Time*. Arden uses *Time* to store date and time together, with functions available to extract subparts, while CQL uses different constructs for *date*, *time*, and *datetime*. In addition, the standards have different data types with no equivalent in the other language. Arden, for example, has *truth value* and *fuzzy set* to represent the degree of certainty, and CQL has *code and concept*, which are used to define a terminology declaration.

Data models: Arden does not facilitate use of references to specific data models, while CQL supports use of multiple data models such as QDM and Fast Healthcare Interoperability Resources (FHIR).[18] To use a data model, Arden developers must create related objects in the *data* slot and assign values with the data retrieved with the *read* statement (►Fig. 4). References to local data stores are enclosed in curly braces for easy identification and revision during the knowledge sharing process involving another organization using a different data model.

Data context: The logic in an artifact can be applied to a single patient, or multiple patients. Arden does not have a specific construct to specify the context and it typically focuses on a single patient, however, because of the curly braces it could also support multiple patients.[19] CQL uses the *context* declaration to evaluate the logic against data of *patient*, *practitioner*, or *unfiltered* (data not constrained by patient or provider).

Data conversion: Data can be converted or cast from one type to another in both standards. Their common conversion is the *as String* operator. Arden also can convert data *as number*, *as time*, and *as truth value*. CQL has various operators (i.e., *as*, *to*, *convert to*, and *from/to*) to support data conversion. It can also convert units (e.g., "g" to "kg") and time units (e.g., days to weeks).

External resources: The logic module in Arden facilitates access to data and resources to/from sources using the curly braces approach. For instance, the *read* statement can perform a query to a database or a FHIR service (►Fig. 5). The *interface* statement can call an external function, and the *write* statement can send text or a message to a destination such as an email address. CQL uses local identifiers that represent external references (e.g., object identifier, a uniform resource identifier, or any other identification system) with value sets, standardized vocabularies, or ontologies, using the identifiers *valueset* and *codesystems* to define terminology declarations (e.g., codesystem "SNOMED": 'http://snomed.info/sct'). However, "CQL does not interpret the external id."[7] CQL can reference elements from a data model (►Fig. 4), reference functions from other CQL libraries (►Fig. 5), and invoke external functions. However, the CQL specification states that "the use of external functions is discouraged because they hinder one of the foundational benefits of CQL, which is data exchange."[7]

Terminologies: Arden does not have constructs to support the use of controlled vocabularies and terminology standards (e.g., Systematized Nomenclature of Medicine (SNOMED),[20] Logical Observation Identifiers Names and Codes,[21] etc.). Arden "does not include a vocabulary, nor does it dictate that a particular vocabulary or vocabularies must be used."[22] CQL has terminology declarations such as *codesystem*, *valueset*, *code*, and *concept*, which permits codes to be used within the artifact (►Fig. 2).

CQL

```
include CMS153_Common version '2' called Common

define SexuallyActive:
   exists (Common.ConditionsIndicatingSexualActivity)
   or exists (Common.LaboratoryTestsIndicatingSexualActivity)
```

Arden

```
data:
(X_TOKEN)      := ARGUMENT;
PatientBundle  := READ {fhir:Patient?identifier=X_TOKEN};
DOB            := PatientBundle.entry.resource.Patient.birthDate.value;
Gender         := PatientBundle.entry.resource.Patient.gender.value;

MData          := READ {GetMData where token=X_TOKEN and
                        SearchBackAmount=365 and
                        HoursOrDays=Days};
MedName        := MData.MMeds.Name;
Dose           := MData.MMeds.Dose;
```

Fig. 5 Example of referencing function from another Clinical Quality Language (CQL) library (source: https://cql.hl7.org/02-authorsguide.html), and a *data* slot in Arden demonstrating the read of data from a Fast Healthcare Interoperability Resources (FHIR) service and a database.

CQL

```
codesystem "ICD-10-CM": 'http://hl7.org/fhir/sid/icd-10-cm'

valueset "Diabetes VS":
'https://cts.nlm.nih.gov/fhir/ValueSet/2.16.840.1.113883.3.464.1003.103.12.1001'

code "Familial hypercholesterolemia code": 'E78.01' from "ICD-10-CM" display 'Familial
hypercholesterolemia'
```

Fig. 6 Example of Clinical Quality Language (CQL) *codesystem*, *valueset*, and *code* constructs used to reference terminologies.

Statements

Conditional statements: Arden and CQL support condition execution. They have an *if-then-else* statement and some sort of case statement such as the *switch-case* statement (Arden) and the *case-when-then* statement (CQL).

Iteration statements: Only Arden has statements to support iteration, including the *while* loop and *for* loop. However, CQL, like Arden, can loop through a list and perform operations using elements from the query, list, arithmetic, comparison, and aggregate operators.

Operators

Logical operators: Arden and CQL have similar logical operators (i.e., *and*, *or*, *not*). CQL also has the *xor* (i.e., exclusive or) and *implies* operators.

Arithmetic/Numeric operators: The standards have similar arithmetic operators (e.g., +, –, *, *Ceiling*, *Floor*) and additional operators not found in both. Arden has some operators such as *sine*, *cosine*, *arcsin*, and *sqrt* that are not available in CQL. Similarly, CQL has operators not available in Arden such as *mod*. Arden and CQL have similar functions represented by different symbols. For example, the power operator in Arden is ** and in CQL is ^). Some functions in both formalisms have similar names but different functionalities. For example, the operator *Log* in Arden is used to return the natural logarithm of its argument, but in CQL, the *Log* operator returns the logarithm of the first argument using the second argument as the base. In CQL, the natural logarithm is the *Ln* operator.

Comparison operators: Both standards have a variety of comparison operators that can be used on numbers, dates, times, and intervals (e.g., =, <, <=). Arden has a larger selection of named functions, including the *Is* and *Occur* comparison operators. Regarding the comparison of intervals, CQL has a more comprehensive set of operations, compared with Arden, including *same as*, *before*, *after*, *meets before*, *meets after*, *meets*, *overlaps before*, *overlaps after*, *overlaps*, *begins*, *included in*, *includes*, and *ends*.

Date/Time operators: Arden and CQL have an extensive list of operators to handle data and time operations for construction, extraction, arithmetic operations, and calculation of duration and differences. As in other operators, they have similar constructs with same name and function (e.g., +, *year*, *now*) and different names and similar functionalities (e.g., *Time* in Arden and *DateTime* in CQL store both date and time information).

List operators: Both standards have several operations to create, compare, and compute data on lists, including accessing an item, finding the index of an item based on a value, the first or last item from a list, and the length of the list.

Aggregate operators: Arden and CQL support summaries and statistical calculations on lists, and have similar operations (e.g., *count*, *sum*, *stddev*, *variance*, *median*, etc.). CQL has additional operators for calculating the geometric mean, population standard deviation, and population variance.

String operators: The manipulation of strings is very similar in both standards with operations to concatenate string, find positions and expression patterns within a string, split a string, convert a string to upper or lower case, and access specific index of a string.

Query operators: Both standards allow querying and retrieving data and performing operations on results. Arden uses the *where* clause (similar to the *select…where* clause of relational databases) and various query aggregation operators to handle data from a list and query results. CQL uses a SQL-like approach to query data scoped to the retrieve *context*.[7] A query can be performed on one or multiple sources (i.e., *with/without* clauses) and the result filtered (i.e., *where* clause), sorted (i.e., *sort* clause), and removed/included as needed (i.e., *return* clause).[7]

Object operators: Arden can work with complex data structures by creating a new *Object* with attributes that can be accessed using the *dot* operator (e.g., person.name). Objects can be cloned (deep copy) and have the content of its attributes retrieved. CQL has the construct *Tuple* that allows creating structured values, but CQL can also access the data models objects and attributes using the *dot* notation (e.g., Patient.birthDate from FHIR).

Uncertainty: Although both standards facilitate management of uncertainty, they take different approaches to it. Arden uses *fuzzy set* and *fuzzy logic* to handle uncertain data and imprecise reasoning found in patient's data, medical knowledge, and clinical reasoning.[8,23–25] The fuzzy sets can be used with numbers, time, and duration data types, and the ranges can be extended with the fuzzified statements. For example, for the fuzzy condition "*blood_glucose_level is within 180 fuzzified by 50 to 250 fuzzified by 50*", "a blood glucose level of 275 would return the fuzzy truth value 0.5 while a level of 251 would return the fuzzy truth value 0.98, which could be interpreted as 'almost true'."[25] Arden can also define the degree of applicability (a number between 0 and 1) of a value. CQL defines uncertainty to specify the semantics for date, time, duration, and number comparisons. However, it does not have constructs to represent or calculate gradual transition or degree of applicability. Uncertainty can be represented with a range of values, like an interval. For example, the expressions "*days between Date(2014, 1, 15) and Date(2014, 2)*" cannot be calculated accurately and will result in several days between 17 (to the first day of February) and 44 (to the last day of February).[7]

Discussion

This study compared two HL7 standard formalisms for coding CDS knowledge into computable artifacts. Both Arden and CQL facilitate the sharing, management, and reuse of

Table 3 Arden and CQL implications on utility

	Category	Arden	CQL	Implication
Language	Grammar	★★	★★	Both standards allow the creation of a logic module consisting of English-language-like statements
	Machine-friendly executable	★★	★★	Both standards can convert the logic into an XML/JSON format to be used by other systems
	Code execution	★★	★★	Arden executes the expressions in sequence and CQL evaluates all the expressions based on their interdependencies (like in production rules)
	Modularization	★★	★★	Both standards have constructs to break the logic into modules. In Arden, each function must be a separate MLM file imported and called individually. Arden has special constructs to also access external logic modules and resources
	Event triggering	★★	☆☆	Only Arden has constructs to support event triggering
	Maintenance/Library	★★	★☆	Arden has a more robust metadata library that would better support integration with knowledgement databases and key metadata uses. CQL leaves metadata about the library to be stored outside of the artifact
Data	Data types	★★	★★	

(*Continued*)

Table 3 (*Continued*)

	Category	Arden	CQL	Implication
				Arden and CQL can support basic primitive data types and some specific data types
	Data models	☆☆	★★	CQL alignment with several quality and data exchange models facilitates technical integration and sharable resources
	Data context	★☆	★★	Both standards can work with EHR data with different contexts, but only CQL has a language construct to control the context such as patient, provider, and unfiltered
	Data conversion	★★	★★	Both standards can convert data from one type to another
	External resources	★★	★☆	Arden facilitates access to external logic modules and resources, while CQL provides references to value sets and code systems and also definition of functions from external libraries
	Terminologies	☆☆	★★	Only CQL has terminology declarations to represent codesystem, valueset, code, or concept, which can be used anywhere within the artifact
Control statements	Conditional statements	★★	★★	Both standards support conditional statements
	Iteration statements	★★	☆☆	Only Arden has loop statements
Operators	Logical operators	★★	★★	Both standards support logical operators
	Arithmetic/Numeric operators	★★	★★	Both standards support a variety of arithmetic operators
	Comparison operators	★★	★★	Both standards support a variety of comparison operators
	Date/Time operators - Construction	★★	★★	Both standards support a variety of date/time operators to create and compare date and time
	Date/Time operators - Extraction	★★	★★	Both standards support a variety of date/time operators to extract information from date and time
	Date/Time operators - Arithmetic	★★	★★	Both standards support date/time operators to perform arithmetic operations with date and time
	Date/Time operators – Duration and Differences	★★	★★	Both standards support date/time operators to perform calculations of duration and differences between dates and times
	List operators	★★	★★	Both standards support a variety of operations with lists
	Aggregate operators	★★	★★	Both standards support a variety of mathematical operations
	String operators	★★	★★	Both standards support a variety of string operators
	Query operators	★★	★★	Both standards support query operators, with CQL being closer to a SQL-like query operations
	Object operators	★★	★☆	Arden has several operators to handle objects, while CQL has only tuple as an object. CQL, however, can access objects and attributes from the data models
	Uncertainty	★★	★☆	Both standards support uncertainty. Arden uses fuzzy set and fuzzy logic to handle vague data and imprecise reasoning, and CQL uses

Table 3 (*Continued*)

	Category	Arden	CQL	Implication
				intervals for date, time, duration, and numbers, but does not have a construct to represent or calculate gradual transition or degree of applicability

Abbreviations: CQL, Clinical Quality Language; EHR, electronic health record; MLM, Medical Logic Modules.
Note: ★★ = addressed; ★☆ = partially addressed; ☆☆ = not addressed.

computable knowledge, and both allow knowledge to be represented with their languages and converted to a machine-friendly executable code that can be shared and reused by other systems. ►Table 3 presents a summary of the implications on utility of these standards regarding each category. They have constructs with similar functions that can be used for representing CDS knowledge, but also have unique constructs that could support distinct use cases. They each have constructs that fully or partially address several of the categories used in this study, except for data models and terminologies in Arden and event triggering and loop statements in CQL.

The utility of a single standards development organization offering two different solutions for computable clinical knowledge representation can be unclear. However, both Arden and CQL have evolved under different influences to occupy distinct niches. Arden, intended as a general-purpose CDS programming language that incorporates features of CDS workflow, such as event detection and triggering as well as alert messaging, is an international realm standard. It has been implemented in several countries including France, Germany, Austria, Sweden, Korea, and the United States, and has been incorporated by EHR system vendors in their CDS offerings. Its use has addressed a wide variety of clinical scenarios, including laboratory observation monitoring and interpretation, medication decision support, clinical practice guideline dissemination, and cohort identification for clinical trial recruitment. In contrast, CQL was created by HL7 as a U.S. realm standard. As its name suggests, one rationale that informed development of CQL was a desire in the U.S. to standardize the representation of clinical quality measures as part of a focus on quality improvement in health care. With a focus by developers to create just an expression language, CQL lacks some of the CDS process features such as event trigger and messaging constructs as well as bibliographic and other metadata that can support clinical reasoning. However, the intent was that CQL expressions would be included in a wrapper, such as the FHIR PlanDefinition resource, which would help provide key details such as explanations, triggers, messages, and the like that already are present in an Arden Syntax MLM. Accordingly, while Arden can be used to implement quality measures, and CQL employed for use cases beyond quality measures, these formalisms support use cases that reflect their distinct origins. However, to provide robust yet nonduplicative tools to vendors and health care communities, convergence or harmonization of standards, may ultimately result. Other important standards to consider for harmonization include those that support business process modeling such as the Object Management Group's Business Process Model and Notation (BPMN),[26] Case Management Model and Notation (CMMN),[27] and Decision Model and Notation (DMN)[28] specification.

Regardless of future harmonization directions, key to the functionality and shareability of knowledge representation formalisms is the use of standard data models. In this light, with support by several of the largest EHR vendors in the U.S. (e.g., Epic, Cerner, Allscripts, NextGen, athenahealth, etc.), and being required by the Office of the National Coordinator for Health IT's 21st Century Cures Act Final Rule as a standardized application programming interface to support interoperability,[29] FHIR support will be an important criterion for selecting languages for developing CDS systems. Currently, only CQL supports FHIR as a standard data model. However, the Arden Syntax Work Group will introduce a standard data model in the Arden Syntax version 3.0, and FHIR is the candidate data model, which "would enhance standardization of data access and eliminate or reduce use of the curly braces."[13]

This study has several limitations. We did not analyze and compare every single construct between the two languages but instead focused on the constructs relevant to CDS. The analysis was solely based on the content of the standard specifications and not tested on live systems.

Conclusion

Both Arden and CQL can be used to represent CDS knowledge. They both support a variety of operations, can access external data, and invoke other modules to represent complex CDS artifacts. Because there is overlap in functionality between Arden Syntax and CQL, and because they evolved to fulfill different missions, there is confusion over which formalism is best used in which circumstances and to accommodate different CDS use cases. CQL's constructs to support use of standard external data models and terminologies will continue to be a facilitator to adoption. However, the HL7 working groups responsible for the development of the Arden standards can direct future development to enhance these functions.

Clinical Relevance Statement

This article provides a comparative analysis of two HL7 standard formalisms (i.e., Arden and CQL) for coding CDS knowledge into computable artifacts and uncovers their similarities and differences. Standard formalisms support

CDS dissemination and implementation by facilitating knowledge sharing, reuse, and scalability. This in turn promotes the improved clinical and process outcomes, such as care quality, safety, efficiency, and cost that can be realized through the implementation of CDS.

Multiple Choice Questions

1. What Arden construct can perform a query to a database?
   a. Read
   b. Query
   c. Destination
   d. Write

   Correct Answer: The correct answer is option a. The Read statement can read data from external data sources using a database query written within curly braces.

2. What the CQL construct *valueset* is used for?
   a. To set the value of a logical variable
   b. To define terminology declarations
   c. To read the value of an item in a list
   d. To define library declarations

   Correct Answer: The correct answer is option b. The *valueset* is used to define terminology declarations with a list of codes and terms from a codesystem.

Protection of Human and Animal Subjects

Human and animal subjects were not included in this project.

Funding

Dr. Jenders was supported by NIMHD grants U54MD007598 and S21MD000103 and NCATS grant UL1TR001881 from the National Institutes of Health (USA).

Conflict of Interest

None declared.

Acknowledgments

Drs. Jenders is co-chair and Dr. Harrison is a member of the HL7 Arden Syntax work group. Dr. Jenders additionally is co-chair of the HL7 Clinical Decision Support work group.

References

1 Blumenthal D, Tavenner M. The "meaningful use" regulation for electronic health records. N Engl J Med 2010;363(06):501–504
2 Electronic Specifications for Clinical Quality Measures | CMS. Accessed December 30, 2020 at: https://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/Electronic_Reporting_Spec
3 Behilng D, Davis D, Sherman G, et al. Quality Data Model Based Health Quality Measures Format (eMeasure) Implementation Guide, Release 1 (US Realm) Based on HL7 HQMF Release 2.0.
4 QDM - Quality Data Model | eCQI Resource Center. Accessed December 30, 2020 at: https://ecqi.healthit.gov/qdm
5 HQMF - Health Quality Measure Format | eCQI Resource Center. Accessed December 19, 2020 at: https://ecqi.healthit.gov/hqmf
6 Value Set Authority Center. Accessed December 28, 2020 at: https://vsac.nlm.nih.gov
7 Clinical Quality Language (CQL). Accessed July 23, 2020 at: https://cql.hl7.org.
8 Jenders RA, Adlassnig K-P, Fehre K, Haug P. Evolution of the Arden Syntax: key technical issues from the standards development organization perspective. Artif Intell Med 2018;92:10–14
9 HL7 Standards Product Brief - Arden Syntax v2.10 (Health Level Seven Arden Syntax for Medical Logic Systems, Version 2.10) | HL7 International. Accessed July 23, 2020 at: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=372
10 Lomotan EA, Meadows G, Michaels M, Michel JJ, Miller K. To Share is Human! Advancing evidence into practice through a national repository of interoperable clinical decision support. Appl Clin Inform 2020;11(01):112–121
11 Jenders RA. Chapter 15 - Decision rules and expressions. In: Greenes RA, ed. Clinical Decision Support. 2nd ed. Academic Press; 2014:417–434. ISBN: 9780123984760
12 Robert W. Sebesta. Concepts of Programming Languages. 10th ed. Pearson; 2013. ISBN-13: 9780131395312
13 Jenders RA, Haug P, Adlassnig K. P. Arden Syntax Implementation Guide Release 3. 2019. Accessed December 27, 2020 at: https://confluence.hl7.org/download/attachments/40741751/arden-syntax-IG-R3-2019-09-16-ballot-reconciliation.pdf
14 Cooking with CQL. GitHub. Accessed March 23, 2021 at: https://github.com/cqframework/CQL-Formatting-and-Usage-Wiki
15 Backus-Naur Form - an overview | ScienceDirect Topics. Accessed July 23, 2020 at: https://www.sciencedirect.com/topics/computer-science/backus-naur-form
16 ANTLR. Accessed July 23, 2020 at: https://www.antlr.org
17 Jung CY, Sward KA, Haug PJ. Executing medical logic modules expressed in ArdenML using Drools. J Am Med Inform Assoc 2012;19(04):533–536
18 Overview FHIR. HL7 FHIR. Accessed August 4, 2020 at: https://www.hl7.org/fhir/overview.html
19 Kraus S, Drescher C, Sedlmayr M, Castellanos I, Prokosch H-U, Toddenroth D. Using Arden Syntax for the creation of a multi-patient surveillance dashboard. Artif Intell Med 2018;92:88–94
20 Systematized Nomenclature of Medicine (SNOMED). Accessed March 23, 2021 at: https://www.snomed.org
21 Logical Observation Identifiers Names and Codes (LOINC). LOINC. Accessed March 23, 2021 at: https://loinc.org
22 Hripcsak G, Ludemann P, Pryor TA, Wigertz OB, Clayton PD. Rationale for the Arden Syntax. Comput Biomed Res 1994;27(04):291–324
23 Klaus-Peter Adlassnig Karsten Fehre. Service-Oriented Fuzzy-Arden-Syntax-Based Clinical Decision Support. Indian J Med Inform 2014:75–79. Accessed May 29, 2021 at: http://www.meduniwien.ac.at/kpa/publications/2014_IJOMI_–_Service-Oriented_Fuzzy-Arden-Syntax-Based_Clinical_Decision_Support.pdf
24 de Bruin JS, Steltzer H, Rappelsberger A, Adlassnig K-P. Creating clinical fuzzy automata with fuzzy Arden Syntax. AMIA Annu Symp Proc AMIA Symp 2017;2017:475–484
25 Tiffe S. Defining medical concepts by linguistic variables with fuzzy Arden Syntax. Proc AMIA Symp2002:796–800
26 BPMN Specification - Business Process Model and Notation. Accessed March 23, 2021 at: https://www.bpmn.org
27 Case Management Model and NotationTM (CMMNTM) | Object Management Group. Accessed March 23, 2021 at: https://www.omg.org/cmmn
28 Decision Model and NotationTM (DMNTM) | Object Management Group. Accessed March 23, 2021 at: https://www.omg.org/dmn
29 ONC's Cures Act Final Rule. Accessed May 10, 2020 at: https://www.healthit.gov/curesrule