Ontology-Driven Decision Support Systems for Health Care

S. Shridevi, V. Viswanathan and B. Saleena

Abstract Decision support systems (DSSs), as means of diffusing clinical guidelines, are powerful software system that will result in an improvement of medical practices. However, they are not invariably efficient and may suffer from limitations among which are lack of flexibility and weaknesses in the integration of many clinical guidelines for the management of patient's details. Recent research efforts resulted in a vital range of semantic reference systems enriched with vocabularies, thesauri, terminologies, and ontologies. The intensive use of ontologies is included in a new approach to create modern intelligent systems, reusing and sharing pieces of declarative information that plays a significant role in a DSS. A lot of effort has been made to produce standard ontologies for medical knowledge representation. This chapter brings an overview of semantic knowledge representation frameworks such as RDF and OWL for developing ontology and presents a DSS that is enabled by ontology for healthcare domain. A clinical use case is illustrated highlighting the role of ontology in medical DSS.

Keywords Ontology · RDF · RDFS · OWL · Medical ontologies
DSS

1 Introduction

Knowledge illustration presents a vital downside in today's science, notably if this knowledge has to be effectively used for reasoning as a part of the decision support systems (DSSs). Medical domain is defined by the abundance of existing

S. Shridevi (✉) · V. Viswanathan · B. Saleena
VIT University, Chennai, India
e-mail: shridevi.s@vit.ac.in

V. Viswanathan
e-mail: viswanathan.v@vit.ac.in

B. Saleena
e-mail: saleena.b@vit.ac.in

S. Margret Anouncia and U. K. Wiil (eds.), *Knowledge Computing and its Applications*, https://doi.org/10.1007/978-981-10-8258-0_4

professional knowledge, and practically each of its specializations includes a constantly growing and interacting range of relevant guidelines. A long goal is illustration of this knowledge in a form which can be used by systems, supporting medical decision making. An approach is critical which can change systematic illustration of various kinds of medical knowledge that may be used for various reasoning, ranging from offline and online warning systems to planning tending activities.

In this chapter, our aim is to give an overview of different metadata structures and schemes proposed for knowledge depiction, storage, and management of information. Any knowledge representation system must have styles for (i) representation and (ii) inference. Based on that, a brief discussion on two vital Semantic Web ontology representation formalisms—namely RDF-based metadata framework *and* OWL-based metadata framework—is presented. The frameworks are described and illustrated with rules and query languages for handling metadata and repositories. The chapter is structured as follows. Section 2 will discuss RDF-based metadata framework constructs; Sect. 3 will discuss OWL-based metadata framework constructs. Section 4 demonstrates a case study in the application of ontology-based DSS for clinical system by exploiting popular ontological resources of medical domain. Section 5 contains the conclusions and summary.

2 RDF-Based Metadata Framework

The RDF particulars were outlined starting from the earliest stage as a dialect for metadata about Web resources. The metadata model of RDF was proposed by the W3C and comprises of the below specifications:

- RDF/XML in terms of XML namespaces is used as specifications for RDF in XML syntax, XML information sets, and XML-based specifications.
- RDF statements are instances or extensions to the specifications for RDF schema (RDF(S)), which is utilized to characterize RDF vocabularies or models.
- The SPARQL is a query language and protocol for retrieving RDF instances from RDF information stores or data stores.

2.1 Resource Description Framework

The Resource Description Framework (RDF) [2] offers a basic yet valuable semantic model based on graph structure. Generally, RDF is a meta-language for characterizing declarations or statements about resources and relations or links among them. Uniform Resource Identifiers (URIs) are utilized for recognizing such statements and connections. RDF gives constructs for characterizing resources and properties using classes. Statements are framed using these classes that declare facts

about resources. RDF utilizes its own constructs (RDF schema or RDFS) for designing a schema for a resource. RDF is less expressive than RDFS, and RDFS incorporates subclass/superclass connections and restrictions or constraints on the resources complying with the schema. The motive of RDF's theoretical model is to break down data, and every little piece has unmistakably characterized semantics in such a way machine can comprehend it and derive meaning with it. Presently, utilizing RDF's constructs follows below rules:

Rule #1: Information should be considered as statements; each statement must be in the form of Subject–Predicate–Object, and change of this triple order is not allowed.
Rule #2: Uniform Resource Identifier (URI) must be the way to name or identify a resource and it must be universal.

RDF utilizes URIs rather than words for identifying resources and properties. Besides, all the URIs in such a vocabulary typically share a namespace prefix for this vocabulary as mentioned below:

http://www.w3.org/1999/02/22-rdf-sentence structure ns#

RDF vocabulary is summarized in Table 1:

The rest of the RDF Syntax names are: datatype, Description, parseType, ID, about, resource, li and nodeID with rdf: prefixed for all. Hereby, rdf:name is utilized to demonstrate RDF vocabulary constructs and the URI of rdf:type is given as below:

http://www.w3.org/1999/02/22-rdf-sentence structure ns#type

2.1.1 Syntax and Examples for Basic RDF Constructs Follows

Consider a basic sentence: The name of "sickperson1" is "Alex" which has the below parts:

- http://www.hospital.org/Sickpersons/sickperson1 as Subject or Resource,
- Name as Predicate or Property,
- "Alex" as Object value.

Representing a sick person using RDF expects all resources to be considered as URIs. The doctor of this sick person is also a resource. Doctor name is Kishore with kishore@clinic.org as his mail id, and he treats http://www.clinic.org/Sickpersons/

Table 1 RDF constructs

Classes	rdf:Statement, rdf:Property, rdf:XMLLiteral, rdf:Seq, rdf:Bag, rdf:List, rdf:Alt
Properties	rdf:type, rdf:predicate, rdf:object, rdf:subject rdf:first, rdf:rest, rdf:_n, rdf:value
Resources	Rdf:nil

sickperson1. This could be taken as "http://www.clinic.org/Sickpersons/sickperson1" is dealt by somebody with name Kishore and email as kishore@-clinic.org. This can be taken as two sentences: The person identified as http://www.clinic.org/Physicians/physician1 has name Kishore with kishore@clinic.org as his mail id. The sick person http://www.clinic.org/Sickpersons/sickperson1 was dealt by this doctor. From the aspect of data integration, RDF promotes three different container objects such as Bag, sequence and alternative where URIs are used to identify same nodes and RDF graphs can be merged using the same.

2.1.2 RDF Serialization

Multiple syntaxes are followed in serializing RDF data model. Among them, RDF/XML and triples-based syntax are the two commonly used syntaxes. Consider the example discussed earlier. The XML serialization of that example is as follows.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:about="http://www.hospital.org/Sickpersons/sickperson1">
<treated-by rdf:resource="http://www.hospital.org/Physicians/physican1"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.hospital.org/Physicians/physician1">
<Name>Kishore</Name>
<Email>kishore@hospital.org</Email>
</rdf:Description>
</rdf:RDF>
```

The triples-based serialization for the same example is as follows.

```
<http://www.hospital.org/Sickpersons/sickperson1> <treated-by> <http://www.
hospital.org/Physicians/physican1>.
<http://www.hospital.org/Physicians/physician1> <Name> "Kishore".
<http://www.hospital.org/Physicians/physician1> <Email> "kishore@hospital.
org".
```

2.2 Core Elements of RDFS Schema

The section discusses the constructs of RDF schema (RDFS). RDFS vocabulary contains terms that are used to define properties and classes for a domain-specific application. Similar to RDF constructs, all the RDFS terms are also identified by pre-defined URIs with the following leading string:

http://www.w3.org/2000/01/rdf-schema#

The above URI is the namespace prefix for RDF/XML format with the prefix string RDFS. The terms are grouped as classes, properties, and utilities.

- classes

The class constructs of RDFS that are used to define classes are rdfs:Resource, rdfs: Class, rdfs:Literal, rdfs:Datatype.

- properties

The property constructs of RDFS that can be used to define properties prefixed with rdfs are range, domain, subClassOf, subPropertyOf, label, and comment.

- utilities

The miscellaneous terms are rdfs:seeAlso and rdfs:isDefinedBy.
The core schema terms are illustrated in Fig. 1 based on their purposes.
A portion of sample rdfs HOSPITAL ontology is given below for better understanding of RDF and RDFS terms usage.

```
<rdfs:Class rdf:about="http://www.hospital.org/Hospital.owl#Cancer">
  <rdfs:subClassOf
rdf:resource="http://www.hospital.org/Hospital.owl#Disease"/>
 </rdfs:Class>
<rdfs:Class
rdf:about="http://www.hospital.org/Hospital.owl#Dermatologist">
  <rdfs:subClassOf
rdf:resource="http://www.hospital.org/Hospital.owl#Doctor"/>
 </rdfs:Class>
```

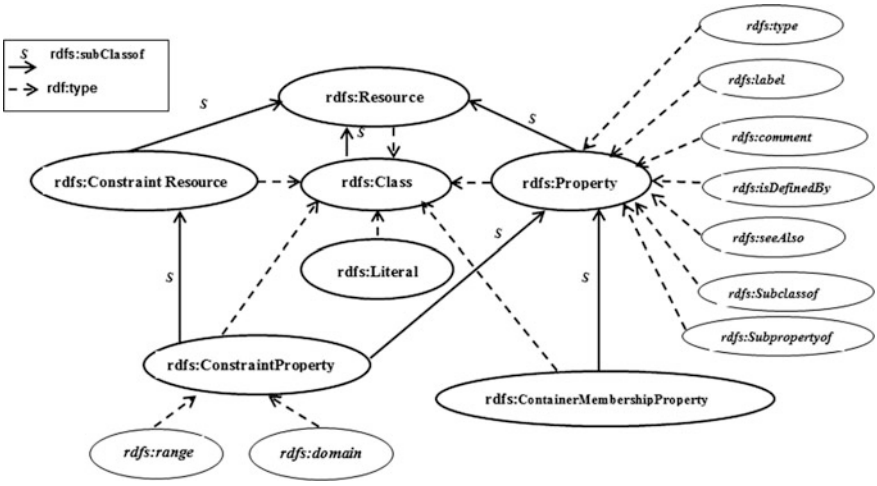


Fig. 1 RDFS vocabulary

```
<rdfs:Class rdf:about="http://www.hospital.org/Hospital.owl#Oral">
 <rdfs:subClassOf
rdf:resource="http://www.hospital.org/Hospital.owl#Treatment"/>
</rdfs:Class>
<rdfs:Class rdf:about="http://www.hospital.org/Hospital.
owl#Sickperson">
 <rdfs:subClassOf
rdf:resource="http://www.hospital.org/Hospital.owl#Person"/>
 </rdfs:Class>
<rdfs:Class rdf:about="http://www.hospital.org/Hospital.owl#Person">
   <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
 </rdfs:Class>
<rdfs:Class rdf:about="http://www.hospital.org/Hospital.owl#Surgeon">
  <rdfs:subClassOf
rdf:resource="http://www.hospital.org/Hospital.owl#Doctor"/>
</rdfs:Class>
```

2.3 The SPARQL Query Language for RDF Data

SPARQL is a W3C recommended query language and a protocol for accessing RDF datasets. SPARQL is "data-oriented" in that it only queries the information held in the RDF models; there is no inference in the query language itself. An RDF graph is a set of triples, and the serialization is just a way to write the triples down as there are multiple ways to encode the same RDF graph. The structure of a SPARQL query is followed as below:

```
#prefix declarations
PREFIX foo: http://example.com/resources/
…
#dataset definition
FROM …
#result clause
SELECT…
Query pattern
WHERE {
….
}
#Query modifiers
ORDER BY….
```

A SPARQL query consists of the following, in order:

- Declarations of Prefixes to abbreviate URIs.
- Definitions of datasets to specify the RDF graph(s) that are being queried.

- A result clause to identify the information returned from the query.
- The query pattern that specifies what to query from the mentioned dataset.
- Query modifiers, slicing, ordering, and otherwise rearranging query results.

Figure 2 illustrates the working of SPARQL query language.

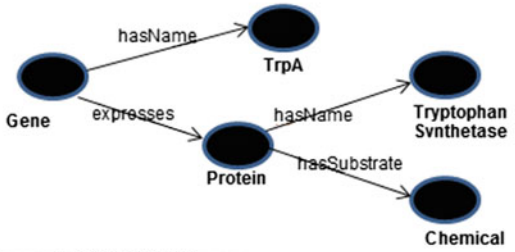
The below query example returns the blood test results for a sick person named "Alex" using a simple graph pattern.

```
Prefix foo:<http://www.hospital.org/Hospital.owl>
SELECT? Result
WHERE {
?sickperson name "Alex".
?sickperson hasbloodTestResult ?Result
}
```

Below query uses value constraints to list only male patients from the whole sick people list.

```
SELECT ?name ?sex
WHERE {
?sickperson sex ?sex.
FILTER (?sex="male").
?sickperson name ?name.
}
```

ID	Subject	Predicate	Object
1	<Gene>	<hasName>	TrpA
2	<Gene>	<expresses>	<Protein>
3	<Protein>	<hasName>	Tryptophan Synthetase
4	<Protein>	<hasSubstrate>	<Chemical>



Example SPARQL Query

```
SELECT ?x WHERE
{
 ?x<hasName>"Tryptophan Synthetase"
 ?x <hasSubstrate> <Chemical>
```

Answer: <Protein>

Fig. 2 SPARQL query execution

3 OWL-Based Metadata Framework

Web ontology language (OWL) [1] is a Semantic Web language which was developed in 2004. It is the formal way of representation of knowledge about concepts/entities, and relationship between the concepts. Representing the concepts and their interdependencies is called ontology. OWL is an extension of RDF syntax with additional features. The knowledge captured in OWL can be easily extracted by programs written in any language as it is a logic-based language.

OWL has a greater number of features for capturing and communicating the meaning and semantics when compared with XML, RDF, and RDFS, and hence, OWL has the ability to represent machine-understandable content on the Web. OWL is a modification of the DAML+OIL Web ontology language consolidating lessons gained from the plan and utilization of DAML+OIL. OWL includes rich vocabulary for portraying attributes and classes among others, relations between classes (e.g., disjointness), cardinality (e.g., "precisely one"), fairness, richer typing of properties and characteristics of properties (e.g., symmetry), and enumerated classes.

3.1 OWL Semantics

OWL leverages upon the strength of Description Logics (DLs), a family of logic-based formal knowledge representation that enables us to capture and represent the concepts in any application domain. The DL models the concepts aka "classes" in RDF, roles aka "properties" in RDF and individuals. Individuals are the atomic elements of the domain; concepts (such as OWL) describe sets of individuals sharing the same characteristics; and describing the nature of relationships between pairs of individuals is done by roles.

Constructors are building blocks for any given DL that describes complex concepts and roles derived from simpler ones. The class constructors accessible in OWL incorporate the Booleans such as "and," "or," and "not," which in OWL parlance are called as intersectionOf, unionOf, and complementOf. The limited types of existential and universal evaluation and the corresponding OWL representation are called someValuesFrom and allValuesFrom restrictions.

OWL also supports transitive properties; in OWL, someValuesFrom limitations are utilized to depict classes, the cases of which are connected by means of an offered property to examples of some different class. Conversely, allValuesFrom limitations compel the conceivable objects of a given property and are ordinarily utilized as a sort of confined range limitation. OWL likewise takes into account property progressions, extensionally characterized classes utilizing the oneOf constructor, backwards properties utilizing the inverseOf property constructor, cardinality limitations utilizing the minCardinality, maxCardinality, and cardinality constructors.

OWL provides three increasingly expressive sublanguages evolved over time addressing specific needs by appropriate communities of implementers and adopters.

- *OWL Lite:* For those users whose primary needs are around classification hierarchy and other simple constraints. For example, restricted permitted value in cardinality constraints values is 0 or 1.
- *OWL DL* supports for advanced users who want to leverage the maximum expressiveness offered in OWL yet wanting to retain the completeness in computation.
- *OWL Full* is meant for the extreme power users who want the maximum expressiveness and the freedom associated with syntactic of RDF with no computational guarantees whatsoever.

Each of these sublanguages encompasses the simpler predecessor, both in what can be legally expressed and holding the validity of the conclusion. The following set of forward progressive relations hold and not their inverses.

- All legal OWL Lite ontology is a subset of OWL DL ontology.
- All legal OWL DL ontology is a subset of OWL Full ontology.

The list of OWL language constructs is given in Table 2.

3.2 *Few Examples of OWL Constructs in Healthcare Domain*

An important feature of OWL is the property restrictions, which can be used to set the constraints on values and specify the cardinalities.

- The Value constraints are allValuesFrom, hasValue, and someValuesFrom,
- The cardinality constraints are cardinality, maxCardinality, and minCardinality.

These are called "qualified cardinality restrictions" (QCRs), is constraining the number of values a particular property type can take on, since it is easy to express constraints like "Each individual has at most one SSN."

(a) **A Surgery Team consists of any person qualified as a doctor.**

```
:Surgery Team
 a OWL: Class ;
 OWL: equivalentClass
 [ a OWL: Restriction ;
  OWL: onProperty : Doctor
  OWL: someValuesFrom : Person
 ]
```

Table 2 Details of OWL constructs

RDF schema		
Name	Syntax	Remark
Class	rdfs:Class	Kinds of things/generic concept of a type or category
SubClass	*rdfs:subClassOf*	Specialization of another general class
Property	*rdf:Property*	Characterizes those classes of things
SubProperty	*rdfs:subPropertyOf*	Relates a property to one of its super properties
Domain	*rdfs:domain*	Specifies object to which properties are applied
Range	*rdfs:range*	Specifies the range of a property P
Individual	*Individual*	Facts about class, property values, and identity
(In)Equality		
EquivalentClass	OWL:equivalentClass	Properties applied to two or more classes will have same values but different meaning
EquivalentProperty	OWL:equivalentProperty	To show two properties have the same property span
SameAs	OWL:sameAs	Links an individual to an individual. Defining mappings between ontologies. Defines class equality, same values and same meaning
DifferentFrom	OWL:differentFrom	Links an individual to an individual, refers to different individuals
AllDifferent	OWL:AllDifferent	All individuals in the given list are different from each other
DistinctMembers	OWL:distinctMembers	Links an individual of OWL:AllDifferent to a list of individuals
Property characteristics		
ObjectProperty	OWL:ObjectProperty	To show the link between individuals
DatatypeProperty	OWL:DatatypeProperty	Individuals are linked to data values

(continued)

Table 2 (continued)

RDF schema		
Name	Syntax	Remark
InverseOf	OWL:inverseOf	Defines inverse relation between properties
TransitiveProperty	OWL:TransitiveProperty	If (a,b) and (b,c) are an instances of X, this infers that (x,z) is also an instance of X
SymmetricProperty	OWL:SymmetricProperty	If (a,b) is an instance of X, then (b,a) is also an instance of X
FunctionalProperty	OWL:FunctionalProperty	Property that defines at most one unique value for each object
InverseFunctionalProperty	OWL: InverseFunctionalProperty	Property that defines two different objects will not have same value
Property restrictions		
Restriction	OWL:Restriction	Describes constraints for the classes to satisfy
OnProperty	OWL:onProperty	Triplet for linking restriction to particular property
AllValuesFrom	OWL:allValuesFrom	Describes all possible values for property
SomeValuesFrom	OWL:someValuesFrom	Describes at least one possible value for property
Restricted cardinality		
minCardinality	OWL:minCardinality	All objects of class must have *at least* N semantically unique values for property
maxCardinality	OWL:maxCardinality	All objects of class must have a *at most* N semantically unique values for property
Cardinality	OWL:cardinality	Restriction applied to a data value belonging to the range of the XML schema datatype
Datatypes		
xsd datatypes	rdf:datatype = ”&xsd; < type>”	Includes Boolean, numerical, string, time-related, URI, etc.

(b) **A Team for a Cyst prostatectomy should have one male doctor.**

```
:Surgery Team
 a OWL: Class ;
 rdfs: subClassOf
  [ a OWL: Restriction ;
   OWL:onProperty :Doctor ;
   OWL:allValuesFrom :  Person
  ] .
 rdfs: subClassOf
  [ a OWL: Restriction ;
   OWL:onProperty : Doctor ;
   OWL:someValuesFrom : MalePerson
  ]
```

(c) **A Team for a C-Section Surgery should have at least three female doctors**

```
:Surgery Team
 a OWL: Class ;
 rdfs: subClassOf
  [ a OWL: Restriction ;
    OWL: onProperty : Doctor ;
    OWL: allValuesFrom : Person
  ] .
 rdfs:subClassOf
  [ a OWL: Restriction ;
   OWL: onProperty : Doctor ;
   OWL: someValuesFrom : FemalePerson
   OWL: minCardinality "3"^^xsd:integer
  ]
```

3.3 Reasoning Using OWL

The OWL ontology requires an inference engine to reason over the knowledge base to discover the hidden relationships from the ontologies. Some of the well-known reasoners include Pellet, Hermit, Fact++, and RacerPro [2]. The availability of such reasoners was the key motivation of W3C to base OWL as a DL. Ontology IDE tools (like SWOOP, code, Protégé 4, and TopBraid Composer) provide feedback to developers on the logical implication of their designs using DL reasoners.

Reasoners can help us identify any missing relationships that of type subclass. For example, When Fact++ was used against SNOMED, it helped discovering 180 missing relationships that were of type subclass.

For instance, researchers utilize ontologies, (for example, the gene ontology and the biological pathways exchange ontology) for annotating information from gene sequencing tests, enabling them to answer complex queries, (for example, "What DNA binding products interact with insulin receptors?"). For the reasoner to answer to not just recognize people that are (maybe just verifiably) cases of DNA-restricting items and of insulin receptors yet to distinguish which sets of people are connected. Rules have to be written using ontology development tools to do the appropriate reasoning. One such language proposed for defining rules for Semantic Web is discussed in Sect. 4.1.

3.3.1 The Semantic Web Rule Language (SWRL)

SWRL is the proposed language to express rules and logics in Semantic Web. Rules help us to infer additional information from the dataset. SWRL allows users to write rules expressed in terms of OWL concepts to reason about OWL individuals. Using the rules, new knowledge can be inferred from existing knowledge bases. There are many built-ins which will provide an extension mechanism whereby the modeling language can be enhanced with domain-specific built-ins. The predicates can be class expressions, property expressions, data range restrictions, sameIndividual, differentIndividuals, core SWRL built-ins, and user-defined SWRL built-ins. Table 3 lists the available built-ins which are defined for various comparisons.

Consider the following indications recorded in the database by patients.

indication(fever), indication(running_nose), indication(headache), indication (body_ache), indication(sore_throat), indication(cough), indication(chills), indication(conjunctivitis)

Table 3 SWRL built-ins for comparisons

Built-ins	Syntax with comment
swrlb:equal	swrlb:equal(?x,?y) If the argument1 and the argument2 are same
swrlb:notEqual	swrlb:notEqual(?x,?y) If argument1 and the argument2 are not same. It is the negation of swrlb:equal
swrlb:lessThan	swrlb:lessThan(?x,?y) If the argument1 is less than the argument2
swrlb:lessThanOrEqual	swrlb:lessThanOrEqual(?x,?y) If the argument1 is less than or equal to the argument2
swrlb:greaterThan	swrlb:greaterThanOrEqual(?x,?y) If the argument1 is greater than the argument2
swrlb:greaterThanOrEqual	swrlb:greaterThanOrEqual(?x,?z) if the argument1 is greater than or equal to the argument2

Suppose doctor gives a rule for Viral_Fever as "If indications are fever, cough, running nose and chills then patient probably have Viral Fever." The rule can be written as

Diagnosis (Viral_Fever) :- indication(fever), indication(cough), indication (running_nose), indication(chills)

The rules written in SWRL can be integrated through a Rule Editor (protégé), and the additional information can be inferred from the patient database.

Indication(?x) ^Indication(?y) ^Indication(?z) ^Indication(?v)←Diagnosis(?X)

Another way of adding rules to the ontology is through JENA. JENA can not only read ontologies, it can also reason out information from the ontology, like other reasoners RacerPro, Pellet.

4 Clinical Decision Support System (CDSS)

Clinical decision support system majorly helps in improving clinical practices. It is a computer-based information system which helps in clinical decision making. The diagnostic features of a patient are mapped to a computerized clinical database using which patient-specific assessment or suggestions are disclosed to the clinician or the patient for decision making [3].

A large quantity of data from medical devices, diagnosing patients, and medical imaging has already been produced periodically in health clinic and health centers [4]. This will continue to happen which creates the need for digitalization of health through high-tech devices and information systems. Due to this explosive growth of data, there is a need to discover or uncover valuable information from that database in order to transform such data into knowledgebase that could help in improving healthcare practice and develop better biomedical products [5]. Due to many challenges, this ultimate goal persists to be a tedious task.

Interoperability is one of the major challenges in managing CDSS [6]. Generally, the distributed data repositories store the generated data from different sources. This database will have inconsistency in naming, structure and format. This distributed data should be transformed into a common format which is accurate, manageable, and efficient in processing the data to integrate with other systems [7].

Since the cost of data integration is high, many hospitals and medical centers are now trying to use Semantic Web Technologies to integrate the data. The advantage of Semantic Web includes the integration of heterogeneous data using explicit semantics, simplified annotation, and sharing of findings. In order to help the organization to adopt Semantic Web, the World Wide Web Consortium (W3C) has formed the Semantic Web for Health Care and Life Sciences Interest Group (HCLS IG). The HCLS IG was established to increase the use of Semantic Web Technologies to support collaboration, innovation, and adoption of Semantic Web in the domains of Health Care and Life Sciences [11].

4.1 Medical Ontologies

BioPortal is a comprehensive repository of biomedical Ontologies [12]. Few of them are discussed below:

4.1.1 SNOMED CT

To develop a comprehensive high-quality clinical/health information in a consistent and reliable manner, SNOMED CT places an integral part [8]. It helps in automatic interpretation of clinical phrases selected by the clinician in a more standardized way. SNOMED CT supports evidence-based care as it is clinically validated and semantically rich, thus benefitting individual patients and clinicians in any location. Processing and presenting the same clinical information to serve different purposes is one of the major advantages of SNOMED CT.

4.1.2 LOINC

Generally, clinical information consists of health measurement, observations, test results, document, etc. LOINC as a common language helps in creating different codes for measurements, observations, test, etc., that has a clinically different vocabulary. To do that, LOINC codes distinguish a given observation across six dimensions called component, property, time, system, scale, and method. It consists of 192,372 classes and 152 properties. It is available only in the Unified Medical Language System (UMLS) format.

4.1.3 MedDRA

Some clinicians are convenient with using and understanding clinical information in their native languages. MedDRA supports in successfully achieving this purpose. It is a multilingual terminology allowing the clinicians and patients to use their native languages [9]. MedDRA uses a hierarchical structure to analyze individual medical events or issues involving a system, organ, or etiology. It uses its multiaxial hierarchy and a set of standardized queries (MedDRA queries) to detect and monitor clinical syndromes whose symptoms consist of numerous systems or organs. A list of few more ontologies which are currently available in BioPortal and number of classes in the ontology, properties, and ontology format are summarized in Table 4.

Table 4 List of ontologies in BioPortal

Name of the ontology	Number of classes	Number of properties	Format
LOINC	1,92,372	152	UMLS
SNOMEDCT	3,27,128	152	UMLS
MedDRA	69,107	16	UMLS
FMA	1,00,080	188	OWL
ICD10	12,445	1	UMLS
RADLEX	46,433	91	OWL
NBO	1,91,799	195	OWL
DRON	4,34,663	20	OWL
MFOEM	899	29	OWL
VO	6211	137	OWL

4.2 *Architecture of Clinical Decision Support System (CDSS)*

Zenuni et al. [10] proposed the system architecture for CDSS depicted in Fig. 3 that includes four modules: the inference engine, the graphical user interface, the SWRL rule, and the ontologies.

In the CDSS architecture, the user can interact through a graphical user interface and place request for any test or diagnostic for a specific illness in the system. The inference engine uses the rules in the knowledge base as the reasoning component through ontologies and ultimately infers a diagnostic for the specific illness tested in

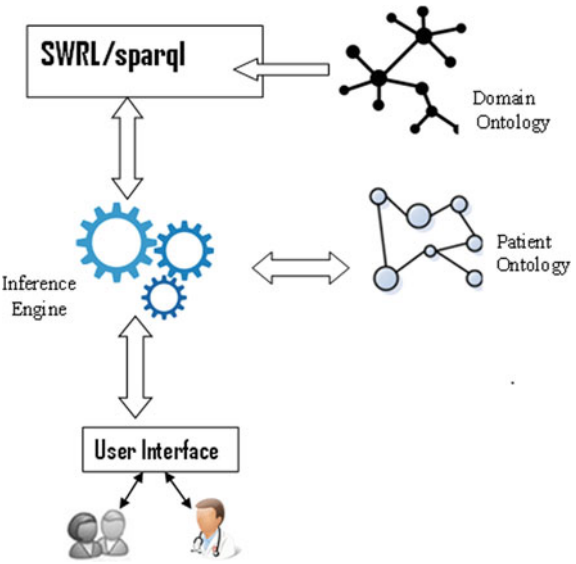


Fig. 3 Architecture of clinical decision support system (CDSS)

the system. The rules in the knowledge base are constructed from the expertise in the medical domain, and these rules are expressed in SWRL format after mapping the domain ontology concepts. During this process, the ontologies feed the reasoning component(s) with the necessary concepts and match their relationships which help the inference engine to combine the rules with concept instances while inferring the diagnostic.

4.2.1 Domain Ontology

A portion of domain ontology which is used for translation medicine during the clinical use case scenario is illustrated below. List of classes and relationships modeled in this ontology are: LaboratoryTestOrder class which denotes the order of the laboratory test for a patient. The class Panel contains one or more tests represented in the class Test and the order may be for a Panel of test represented in the class Panel. The class INDAddress represents the receipientAddress and a payorAddress in the order for a laboratory test. INDAddress represent the set of addresses in the country India. The class Patient is a major concept that features the information about the patient such as the results of diagnostic tests and his family history. It is a subclass of Person. Details related to a patient's family and his/her relatives are denoted using the is_related relationship.

If the family of the patient being evaluated has similar symptoms of disease or ailments, it is captured under the class FamilyHistory and this information related to the Patient class through the relationship called hasFamilyHistory. The information about the results of laboratory tests of the patient is captured under the StructuredTestResult class which is connected to the Patient class through the relationship called hasStructuredTest and the Test class through the relationship called associatedResult.

The MolecularDiagnosticTestResult class represents the results of a molecular diagnostic test result. Molecular diagnostics identify mutations represented using the identifiesMutation relationship and indicate diseases represented using the indicatesDisease relationship in a patient.

Gene class represents information about genes and is associated with the class Patient through the relationship called hasGene. Mutation class represents the genetic variants or mutations of a given gene linked to the class Patient through the relationship called hasMutation. The relationship isMutationOf represents the relationship between gene and mutation.

The class Disease represents the conditions of the disease prevailing in the patient who is diagnosed. It is associated to the class Patient through the relationship suffersFrom and is also associated with molecular diagnostic test class through the relationship indicatesDisease.

OWL representation of the above domain ontology in Turtle format is as follows:

```
@prefix OWL: <http://www.w3.org/2002/07/OWL#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
<http://medical/#Person> a OWL:Class .
<http://medical/#Patient>  a OWL:Class ;
 rdfs:subClassOf <http://medical/#Person>, [
  a OWL:Restriction ;
  OWL:onProperty <http://medical/#isRelatedTo> ;
  OWL:allValuesFrom <http://medical/#Relative>
 ] <http://medical/#Relative>  a OWL:Class ;
 rdfs:subClassOf <http://medical/#Person> .
<http://medical/#StructuredTestResult> a OWL:Class ;
 rdfs:subClassOf [
  a OWL:Restriction ;
  OWL:onProperty <http://medical/#hasPatient> ;
  OWL:cardinality ""^^
1 ""^^xsd:nonNegativeInteger ] .
<http://medical/#MolecularDiagnosticTestResult> a OWL:Class .
<http://medical/#FamilyHistory> a OWL:Class .
<http://medical/#Disease> a OWL:Class .
<http://medical/#Gene> a OWL:Class .
<http://medical/#Mutation>  a OWL:Class ;
 rdfs:subClassOf [
  a OWL:Restriction ;
  OWL:onProperty <http://medical/#isMutationOf> ;
  OWL:someValuesFrom <http://medical/#Gene>
 ] .
<http://medical/#LaboratoryTestOrder> a OWL:Class .
<http://medical/#Panel> a OWL:Class .
<http://medical/#Test> a OWL:Class .
<http://medical/#INDAddress> a OWL:Class .
<http://medical/#isRelatedTo>
 a OWL:ObjectProperty, OWL:TransitiveProperty ;
 rdfs:domain <http://medical/#Patient> ;
 rdfs:range <http://medical/#Relative> .
<http://medical/#hasFamilyHistory> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Patient> ;
 rdfs:range <http://medical/#FamilyHistory> .
<http://medical/#associatedRelative>  a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#FamilyHistory> ;
 rdfs:range <http://medical/#Relative> .
```

```
<http://medical/#hasStructuredTestResult> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Patient> ;
 rdfs:range <http://medical/#StructuredTestResult> ;
 OWL:inverseOf <http://medical/#hasPatient> .
<http://medical/#hasMolecularDiagnosticTestResult> a OWL:
ObjectProperty ;
 rdfs:subPropertyOf <http://medical/#hasStructuredTestResult> ;
 rdfs:range <http://medical/#MolecularDiagnosticTestResult> .
<http://medical/#identifiesMutation> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#MolecularDiagnosticTestResult> ;
 rdfs:range <http://medical/#Mutation> .
<http://medical/#indicatesDisease> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#MolecularDiagnosticTestResult> ;
 rdfs:range <http://medical/#Disease> .
<http://medical/#suffersFrom> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Patient> ;
 rdfs:range <http://medical/#Disease> .
<http://medical/#hasMutation> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Patient> ;
 rdfs:range <http://medical/#Mutation> .
<http://medical/#hasGene> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Patient> ;
 rdfs:range <http://medical/#Gene> .
<http://medical/#isMutationOf> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Mutation> ;
 rdfs:range <http://medical/#Gene> .
<http://medical/#recipientAddress>
 a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#LaboratoryTestOrder> ;
 rdfs:range <http://medical/#INDAddress> .
<http://medical/#payorAddress>
 a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#LaboratoryTestOrder> ;
 rdfs:range <http://medical/#INDAddress> .
<http://medical/#testPanel>  a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#LaboratoryTestOrder> ;
 rdfs:range <http://medical/#Panel> .
<http://medical/#test>  a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Panel> ;
 rdfs:range <http://medical/#Test> .
```

```
<http://medical/#associatedResult> a OWL:ObjectProperty ;
 rdfs:domain <http://medical/#Test> ;
 rdfs:range <http://medical/#StructuredTestResult> .
<http://medical/#orderDateTime> a OWL:DataTypeProperty, OWL:
FunctionalProperty ;
 rdfs:domain <http://medical/#LaboratoryTestOrder> ;
 rdfs:range xsd:datetime .
<http://medical/#PatientWithMYH7Gene> a OWL:Class ;
 rdfs:subClassOf [
  a OWL:Restriction ;
  OWL:onProperty <http://medical/#hasGene> ;
  OWL:hasValue <http://medical/#MYH7>
 ] .
<http://medical/#NormalStructuredTestResult> a OWL:Class ;
 rdfs:subClassOf <http://medical/#StructuredTestResult> ;
 OWL:disjointWith <http://medical/#AbnormalStructuredTestResult> .
<http://medical/#AbnormalStructuredTestResult> a OWL:Class ;
 rdfs:subClassOf <http://medical/#StructuredTestResult> ;
 OWL:disjointWith <http://medical/#NormalStructuredTestResult> .
```

4.2.2 Rules for Clinical Decision Support and Integration of Ontologies

In this section, the rules to integrate with the domain ontology to infer new knowledge as recommendation will be discussed. In some cases, all these rules cannot be written in ontology using OWL and hence such rules are expressed using SWRL. Consider the example: if a patient has a structured test result which is indicative of a particular disease, then the patient suffers from that disease. Properties involved in these rules are hasStructuredTestResult, indicatesDisease, and suffersFrom. These types of rules cannot be expressed using OWL axiom, but it can be expressed using SWRL representation as follows.

hasStructuredTestResult(?x,?y) ^ indicatesDisease(?y,?z) ->suffersFrom(?x, ?z)

Consider another example: if the patient has an allergy to fibric acid or has an abnormal liver panel, then the patient is recommended for ZetiaLipdManagemetn therapy. This rule can be written in SWRL as follows:

hasALPValue (?x,?y) ^ swrl:greaterThan(?y, NormalALPValue) ^ isAllergicTo (?x, FibricAcid)->recommendedTherapy(?x, ZetiaLipidManagementTherapy)

With the help of a reasoner, the above rule can be executed with domain ontology in order to infer new knowledge. Once the doctor enters all the details about patient, the system will recommend decision based on the rules written in the knowledgebase and domain ontology.

5 Summary

In this chapter, we tend to confer an in-depth discussion of two major information frameworks based on RDF and OWL specifications for developing ontologies. The information models and query languages of those specifications were given. There are two main kinds of clinical decision support systems. One form of CDSS, which uses a knowledge domain, applies rules to patient information using a reasoning engine and displays the results to the end user. Systems without a knowledge domain, on the other hand, accept machine learning to investigate clinical information. We have given here a DSS solution based on Semantic Web specifications to a clinical use case situation that takes the benefits of using ontologies as its knowledge domain.

References

1. Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (2004). OWL web ontology language semantics and abstract syntax. W3C recommendation.
2. Carroll, J. J., & Klyne, G. (2004). Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation.
3. Castaneda, C., et al. (2015). Clinical decision support systems for improving diagnostic accuracy and achieving precision medicine. *Journal of Clinical Bioinformatics, 5*(1), 4.
4. Gorman, S. P., Greenes, R. A., Haynes, R. B., Kaplan, B., Lehmann, H., & Tang, P. C. (2001). Clinical decision support systems for the practice of evidence-based medicine. *Journal of the American Medical Informatics Association, 8*(6), 527–534.
5. Riañoa, D., Reala, F., López-Vallverdúa, J. A., Campanab, F., Ercolanic, S., Mecoccic, P., et al. (2012). An ontology-based personalization of health-care knowledge to support clinical decisions for chronically ill patients. *Journal of Biomedical Informatics, 45*(3), 429–446.
6. Jaspers, M. W., Smeulers, M., Vermeulen, H., & Peute, L. W. (2011). Effects of clinical decision-support systems on practitioner performance and patient outcomes: A synthesis of high-quality systematic review finding. *Journal of the American Medical Informatics Association, 18*(3), 327–334.
7. Garg, A. X., Adhikari, N. K., McDonald, H., Rosas-Arellano, M. P., Devereaux, P., Beyene, J., et al. (2005). Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: A systematic review. *JAMA, 293*(10), 1223–1238.
8. SNOMEDCT. (May, 2017). Systematized nomenclature of medicine—clinical terms. Retrieved from http://www.ihtsdo.org/snomed-ct.
9. MedDRA (May, 2017). Medical dictionary for regulatory activities. Retrieved from http://www.meddra.org/.
10. Zenuni, X., Raufi, B., Ismaili, F., & Ajdari, J. (2015). State of the art of semantic web for healthcare. *Procedia, Social and Behavioral Sciences, 195,* 1990–1998.

11. HCLSIG. (May, 2017). Semantic web health care and life sciences interest group. Retrieved from http://www.w3.org/blog/hcls/.
12. BioPortal. (May, 2017). The world's most comprehensive repository of biomedical ontologies. Retrieved from https://bioportal.bioontology.org/.
13. LOINC. (May, 2017). LOINC logical observation identifier names and codes. Retrieved from http://loinc.org/.
14. WHO. World Health Organization. (May, 2017). The ICD-10 classification of mental and behavioural disorders: clinical descriptions and diagnostic guidelines. Geneva: World Health Organization. http://www.who.int/classifications/icd/en/.
15. RadLex. Radiology Lexicon. (May, 2017). http://www.rsna.org/RadLex.aspx.