Taylor & Francis
Taylor & Francis Group

Cybernetics and Systems
An International Journal

ISSN: 0196-9722 (Print) 1087-6553 (Online) Journal homepage: http://www.tandfonline.com/loi/ucbs20

Extended Reflexive Ontologies for the Generation of Clinical Recommendations

Eider Sanchez, Carlos Toro, Manuel Graña, Cesar Sanin & Edward Szczerbicki

To cite this article: Eider Sanchez, Carlos Toro, Manuel Graña, Cesar Sanin & Edward Szczerbicki (2015) Extended Reflexive Ontologies for the Generation of Clinical Recommendations, Cybernetics and Systems, 46:1-2, 4-18, DOI: 10.1080/01969722.2015.1007723

To link to this article: http://dx.doi.org/10.1080/01969722.2015.1007723

Cybernetics and Systems: An International Journal, 46:4–18

ISSN: 0196-9722 print/1087-6553 online
DOI: 10.1080/01969722.2015.1007723

Extended Reflexive Ontologies for the Generation of Clinical Recommendations

EIDER SANCHEZ[1,2,3], CARLOS TORO[1,2], MANUEL GRAÑA[3], CESAR SANIN[4], and EDWARD SZCZERBICKI[5]

[1]*Vicomtech-IK4, San Sebastian, Spain*
[2]*Biodonostia Health Research Institute, San Sebastian, Spain*
[3]*Computational Intelligence Group, University of The Basque Country UPV/EHU, Computer Science Faculty, San Sebastian, Spain*
[4]*School of Engineering, The University of Newcastle, Newcastle, Australia*
[5]*The Gdansk University of Technology, Gdansk, Poland*

Decision recommendations are a set of alternative options for clinical decisions (e.g., diagnosis, prognosis, treatment selection, follow-up, and prevention) that are provided to decision makers by knowledge-based Clinical Decision Support Systems (k-CDSS) as aids. We propose to follow a "reasoning over domain" approach for the generation of decision recommendations by gathering and inferring conclusions from production rules. In order to rationalize our approach, we present a specification that will sustain the logic models supported in the knowledge bases we use for persistence. We introduce first the underlying knowledge model and then the necessary extensions that will convey toward the solution of the reported needs. The starting point of our approach is the proposition of Reflexive Ontologies (RO). Here, we go a step further, proposing an extension of RO that includes the handling and reasoning that production rules provide. Our approach speeds up the recommendation generation process.

KEYWORDS autopoiesis, fast query system, reflexive ontologies, rule engine

Address correspondence to Eider Sanchez, Vicomtech-IK4, Mikeletegi Pasealekia 57, 20009 San Sebastian, Spain. E-mail: esanchez@vicomtech.org

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/ucbs.

INTRODUCTION

Clinical Decision Support Systems (CDSS) are active intelligent systems that use patient clinical data to generate case-specific advice (Liu et al. 2006). According to Greenes (2011), the main task of CDSS consists of the retrieval of relevant knowledge and patient data (coming from medical devices, evidence provided by the medical community, and clinical guidelines and protocols), and their analyses to generate recommendations. CDSS cover a wide span of tools based on several technologies and approaches. Particularly, Power (2008) identifies knowledge-based CDSS (k-CDSS) as tools with specialized problem-solving expertise that allows them to provide decision recommendations to users. Decision recommendations are a set of alternative options for actions (e.g., diagnoses) that have been inferred by the system according to previously established criteria. Recommendations are ranked and presented to system users so that they can easily analyze the different suggested choices, as well as the evidence supporting them. In this regard, k-CDSS act as black boxes that output decision recommendations for a given input dataset. They benefit from a symbolic representation of knowledge about a particular domain, and the ability for reasoning about solutions of problems within that domain (Kalogeropoulos et al. 2003). They are also endowed with the ability to learn from experience (Toro et al. 2012).

In this article, we use a *reasoning over domain* approach for the generation of recommendations. Our aim is to gather and infer conclusions from production rules. We present our reasoning system and the process of generating decisional recommendations. In order to rationalize our approach, we propose a specification that will sustain the logic models supported in the knowledge bases we use for persistence. We introduce first the underlying knowledge model and then the necessary extensions toward the solution of the reported needs. The starting point of our approach is the work of Toro et al. (2008) on reflexive ontologies (RO). We propose an extension of RO by including the handling and reasoning that production rules provide. The reasoning process is detailed in a specification, which allows us to introduce and discuss in depth implementation aspects already achieved in the course of the realization of the research projects that supported this article.

This article is structured as follows: the next section introduces k-CDSS and RO. "Knowledge Model Specification" presents a specification of the underlying knowledge model of the CDSS. "Generation of Recommendations" details the process of generating decision recommendations and is followed by "Specification of Reflexive Ontologies." The section following that proposes a specification of "Extended Reflexive Ontologies." "Generation of Recommendations over Extended Reflexive Ontologies" presents the reasoning process generation over the extended reflexive ontologies paradigm. Next, an "Implementation of Extended Reflexive Ontologies" is

presented. Finally, "Conclusions and Future Work" discusses relevant aspects of our approach.

BACKGROUND CONCEPTS

In this section, we introduce relevant concepts of our approach, which is based on the application and extension of RO to support the process of generating decision recommendations within k-CDSS.

Knowledge-Based CDSS

Knowledge-based CDSS (k-CDSS) have been broadly reported in the literature. Examples are the Bayesian reasoning for general CDSS, Iliad, presented by Warner (1989) and the diagnostic mammography system, Mammonet, based on Bayesian networks presented by Kahn et al. (1995). Among other works based on production rules, we can mention the IMM/Serve immunological CDSS built by Miller et al. (1996). More recently, Wicht et al. (2013) presented a web-based CDSS that follows a case-based approach in which editors were provided for knowledge manual maintenance; Aleksovska-Stojkovska and Loskovska (2011) described an architectural and data model for CDSS, integrated to the clinical system; (Ceccarelli et al. 2008) presented a knowledge-based CDSS for oncology, for which both an ontology and a ruleset were proposed; in Bouamrane et al. (2009), a Web Ontology Language Description Logics (OWL DL) for a preoperative risk assessment CDSS was presented. The proposed system was based on a DL reasoner and a rule engine that provided patient preoperative risk assessments.

The general model of knowledge-based CDSS proposed by Berner and La Lande (2007) consists of four elements:

1. CDSS Input: The CDSS input consists of the patient clinical data for which recommendations are requested. Such data are generally specified in a controlled vocabulary, in which the different variables and their possible values are previously established.
2. CDSS Output: The CDSS output is usually provided as a list of possibilities ranked in some order of probability, such as the most likely, the least likely, and the most safe or risky. Depending on the application domain and the purpose of the system, the most likely possibility could not be interesting for clinicians, as such could be trivial or immediate for them. However, clinicians are, in general, interested in having a broader spectrum of alternatives to consider. Hence, some knowledge-based CDSS are focused on providing less likely options together with the evidence supporting such recommendations.

3. Knowledge Base: The knowledge base consists of medical knowledge. The representation of such knowledge might be obtained by means of the application of different techniques, depending on the technology of the *reasoning engine*. A very common technique is the modeling of the knowledge domain in an ontology, which is defined by Guarino as the explicit and partial account of a conceptualization (Guarino and Giaretta 1995). This variant contains the description of the different elements included in the domain, their relationships and instances. The codification of the criteria for solving the different decisions and aspects of the domain may also be included.
4. Reasoning Engine: The reasoning engine combines the input data and the medical knowledge according to some logical scheme, for generating the output.

Reflexive Ontologies

Reflexive ontologies (RO) define an abstract knowledge structure (i.e., an ontology and its instances) endowed with the capacity to maintain an updated image of every query performed (Toro et al. 2008). That is, the RO maintains the history of queries and the actual collection of instances that answer each query. The purported advantage of RO is that of speeding up query response. It also implies that some knowledge generation can be produced, i.e., new rules can be generated, on the basis of query interaction. This potential behavior was termed *autopoietic* in the original proposal (Toro et al. 2008), following the biological inspiration of Maturana in his seminal work (Maturana and Varela 1980).

Figure 1 shows the logical structure of an RO, which is, basically, a conventional ontology extended with a reflexive structure (mainly composed by

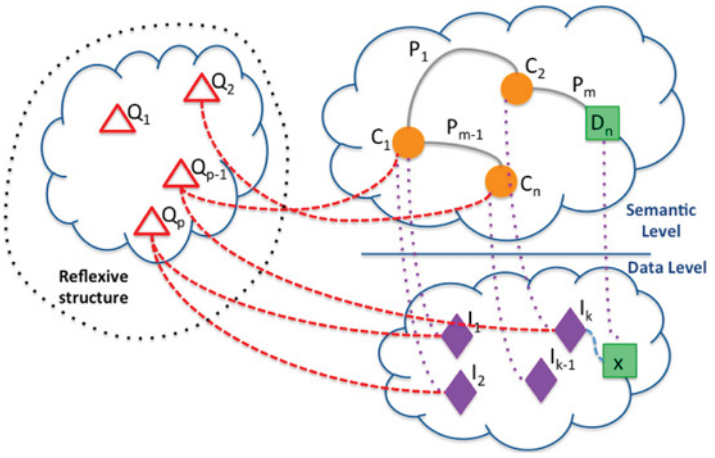

FIGURE 1 Schematic representation of the structure of an RO.

the query instances in the left part of the image). As can be seen, every query Q_p is related to at least one class of the ontology C_i and one—or more—instance I_k.

Among others, RO is based on the concept of autopoiesis, meaning *self-creation* or *self-production*. The RO displays an autopoietic behavior because its structure is regenerated in response to external changes, such as the launching of new queries or the modification of the information stored in the ontology. Moreover, the ontology is capable of storing the history of performed queries, which allows some interesting operations, such as knowing which parts and concepts of the ontology are consulted more regularly. Accordingly, the autopoietic behavior ensures the integrity of the whole RO. When a new individual is created, modified, or removed from the ontology, the reflexive structure is updated. The updating process consists of modifying or generating new references to individuals for each query instance related with the change. By creating new connections (pointers to individuals) as a result of external perturbations, the system behaves as an autopoietic system or organism, according to the definition given by Maturana and Varela (1980).

KNOWLEDGE MODEL SPECIFICATION

This section presents a specification of the knowledge model of a CDSS.

Domain Ontology

Let $O = < C,P,I >$ denote a domain ontology, whose elements are a set of classes $C = \{C_1, C_2, \ldots, C_{N-1}, C_N\}$, a set of properties $P = \{P_1, P_2, \ldots, P_{N-1}, P_N\}$, and a set of instances $I = \{I_1, I_2, \ldots, I_{N-1}, I_N\}$.

- A class *Ci* defines a group of individuals that share common properties. Classes in C can be hierarchically organized.
- A property P_i defines relationships either (i) between sets of individuals, or (ii) from set of individuals to data types. When P_i relates instances of two different classes or instances of the same class it is called an Object Property, P_i^o. Likewise, when P_i relates instances of a class to instances of data types (e.g., integer, string, float), it is called a Datatype Property, P_i^d.
- An individual I_i defines an instance of a class C_i; we use the notation $I_i \in C_i$ to specify this instantiation. Properties P_i between classes are mapped homomorphically to properties relating individuals.

Querying the Ontology

Individuals of an ontology are instances of the classes in the knowledge structure, so that searching in the space of instances is enhanced by the

possibility of reasoning at the semantic level of classes and properties. Hence, an ontology provides semantic enrichment of the data. A query is a search within the ontology that returns a collection of instances satisfying a set of clauses. We specify these ideas as follows: a query is a pair $Q_i = \left(q_i, I_{Q_i}\right)$ where q_i are the clauses specifying the characteristics of the search, and I_{Q_i} is the subset of the ontology individuals matching the query clauses. In fact, a query is a map of the form: $\sigma(q_i) = I_{Q_i} = \{I_k \in I | M(q_i, I_k)\}$, where $M(q_i, I_k)$ is a very general predicate that is true when query clause q_i is satisfied by the assignment of values to variables in an individual I_k (i.e., I_k matches q_i. A query clause q_i can be simple or complex, denoted q_i^s or q_i^c, respectively. A simple query clause is specified by a tuple $q_i^s = V_i, m_i, v_i$, where V_i is a variable, m_i is the comparison operator (i.e., $>$, $<$, $=$), and v_i is a value of the range of V_i. A complex query q_i^c is specified by n simple queries, combined by logical operators, θ, (i.e., $\cup, \cap, \neg$), which define the relationships among consecutive simple queries: $q_i^c = \left\{\left(\theta_n, q_n^s\right)\right\}_{\forall n}$, where θ_n is the nth logical operator (i.e., $\cup, \cap, \neg$), assuming $\theta_O = \emptyset$.

Rules

The atomic knowledge encoding is the *rule*, which states the consequences of the search performed on the semantically enhanced data. Rule consequents are actions involving variable value assignments or recommendations. A rule r_k is composed of a query clause and the consequent actions. Each rule is formalized as a tuple $r_k = < A_k, S_k, L_k, W_k, B_k >$, where

1. A_k is the set of conditional clauses (antecedents), that are equivalent to the q_i part of the queries,
2. S_k is the set of actions corresponding to the THEN consequents,
3. L_k is the set of actions corresponding to the ELSE consequents,
4. W_k is the rule salience (a.k.a. weight), defined as a real number $W_k \in [0,1]$, and
5. B_k is a generic notation for application-dependent ancillary information that can be associated to the rule.

A special kind of action is the assignment of a value to a variable, i.e., $V_1 = v_1$. In the context of a rule, this action is restricted to individual instances fulfilling the antecedent clause of the rule, for the THEN consequent, or its negation, for the ELSE consequent. We assume that the foregoing assignment expression is equivalent to $I_k \cdot V_l = v_l$, where the dot notation specifies the fact that the variable is an attribute of the individual instance, which may fulfill the antecedent clause or not, as discussed before. To identify each of the different types of decisions (recommendations) that can be produced by the search and reasoning over the semantically enhanced data, we

introduce the *decision domain*, denoted d_i. Each d_i is associated to a property P_i in the ontology; we denote this association as follows: $d_i \hookrightarrow P_i$, because it is not strictly a map. We say that a rule r_k is oriented toward a decision domain d_i, when the THEN and ELSE consequents S_k and L_k, respectively, refer to the Property P_i associated with d_i. Each rule r_k is oriented toward some d_i and both consequents, S_k and L_k, must refer to the same set of d_i.

GENERATION OF RECOMMENDATIONS

When inputted a request $J=(I_J, D_J)$, where $I_J \in I$ is a set of individuals for which recommendations are requested, and $D_J \in D$ are the decision domains of those recommendations, the reasoner $\Re$ outputs a set of recommendations $K=\{K_{ij}\}$ for a given ontology O and ruleset R such that j different recommendations are provided for each individual request $J_i=(I_i, d_i)$.

A recommendation is a tuple $K_{ij}=<G_{ij}, W_{ij}, R^{Kij}>$ computed in response to a couple (I_i, d_i) where

- The recommendation consequent G_{ij}, is a collection of output domain assignments u_k associated to rules $r_k \in R^{K_i}$, being $U_k=\{(V_d, v_d)\}$, a collection of domain variable value assignments where d is the domain indicator associated to the consequent of r_k. The value assignment affects the individual instance of the request, i.e., $I_i \cdot V_d = v_d$.
- The weighted probability $W_{G_{ij}} \in [0,1]$ is computed for recommendation G_i,
- A subset of rules $R^{K_{ij}} = \{r_k | M(A_k, I_i)\}, R^{K_{ij}} \in R$, provides the supporting evidence for the recommendation consequent G_{ij}.

The output recommendations in K are not ordered, however, they are given a different weighted probability $W_{G_{ij}} \in [0,1]$ computed from the respective weights W_k of the rules endorsing each recommendation G_{ij}. After all recommendations K_{ij} for a couple (I_i, d_i) are calculated, the weighted probabilities $W_{G_{ij}}$ are normalized to guarantee $\sum_j W_{G_{ij}} = 1$. Each K_{ij} is built by analyzing the sets of instances matching antecedents of rules r_k, whose consequents refer to the same d_i queried in the input request J_i; i.e., $r_k \in R^{Kij}$.

- The matching for each individual I_i and rule r_k is done by translating A_k into a query specification q_i and obtaining the subset of individuals $I_{q_i} \subset I$ that match q_i in ontology O. Let $\overline{I_{q_i}}$ be the set of individuals that do not match q_i in ontology O, such that $I_{q_i} \cup \overline{I_{q_i}} = I$ and $I_{q_i} \cap \overline{I_{q_i}} = \phi$.
- For each individual in I_{q_i}, the domain value assignment $V_d = v_d$ in S_k is selected as recommendation consequent G_{ij}.

- However, for individuals in $\overline{I_{q_i}}$, we select domain value assignment $V_d = v_d$ in L_k.
- Then, W_k is added to $W_{G_{ij}}$ and r_k to $R^{K_{ij}}$.

SPECIFICATION OF REFLEXIVE ONTOLOGIES

In this section, we present the first actual attempt to provide a formal specification of ROs, which, though remaining abstract, is concrete enough to discuss the consequences and degree of implementation. An *RO* is a tuple $RO = < O, Q^t >$, where O is a domain ontology and $Q^t = \{Q_1, Q_2, \ldots, Q_{N\text{-}1}, Q_N\}$ is the set of queries that have been performed over the set of instances, I, and classes, C of the ontology up to time t (see Figure 1). Therefore, the RO is a time varying structure in two senses:

1. Its query set Q^t will be growing in time; each new query will be added to it.
2. Changes in the instance layer, i.e., by the addition of an individual, will be reflected on the queries that include it.

The properties of RO are specified as follows.

Query retrieval. The RO must be able to detect and store every new query—and subquery—performed on it. Let us denote Q_{i^*} a new query posted by the user.

$$\neg \exists q_i \in Q^t s.\text{t.} q_i = q_{i*} \Rightarrow Q^{t+1} = Q^t \cup \{Q_i\}.$$

However, if the query has been already posted and answered, an updated answer will be provided:

$$\exists q_i \in Q^t s.\text{t.} q_i = q_{i*} \Rightarrow I_{Q_{i*}} = I_{Q_i}.$$

Integrity update. The system must be able to actualize the query set every time a new individual is added to, removed from, or modified within the ontology. Let us denote I_k^t the variable value assignment of the kth data instance at time t. The integrity update means that, at any time, if an instance satisfies the clause of a query, then it belongs to the data associated to the query:

$$\forall q_i \in Q^t s.t. M\left(q_i, I_k^t\right) \Rightarrow I_k^t \in I_{Q_i}^t.$$

This specification is purely declarative. If we want to advance something on the mechanism that might implement such property, we can state what happens

for each change in the instance layer. For the sake of notation, let us assume that the introduction of a new instance at time t can be formalized as $I_k^{t-1} = \emptyset$ and $I_k^t \neq \emptyset$. Also, the following holds always: $M(\emptyset, q_i) = F$. Hence, the integrity update can be specified as follows:

$$I_k^{t-1} \neq I_k^t \Rightarrow \begin{cases} \neg M\left(q_i, I_k^{t-1}\right) \wedge M\left(q_i, I_k^t\right) I_{Q_i}^t = I_{Q_i}^{t-1} \cup \left\{I_k^t\right\} \\ M\left(q_i, I_k^{t-1}\right) \wedge M\left(q_i, I_k^t\right) I_{Q_i}^t = I_{Q_i}^{t-1} - \left\{I_k^{t-1}\right\} \cup \left\{I_k^t\right\}. \\ M\left(q_i, I_k^{t-1}\right) \wedge \neg M\left(q_i, I_k^t\right) I_{Q_i}^t = I_{Q_i}^{t-1} - \left\{I_k^{t-1}\right\} \end{cases}$$

The three possibilities specify all possible casuistry. The first case is when the data instance was not included in the past version of the query, but its new values do match the query clause; then the instance is added to the query data. The second case is when the data instance was already in the query, but it has changed; then the instance must be updated in the query (i.e., the old version removed and the new one added). Finally, when the instance no longer matches the query clause, then it must be removed from the query data.

Self-reasoning over the query set. This property states the ability to perform some kind of query result mining. Some possible ways of self-reasoning are as follows:

1. Discover patterns of queries. As an example, assume that some pair of queries Q_{i1} and Q_{i2} have a nonempty intersection of their corresponding data instances, i.e., $I_{Q_{i_1}} \cap I_{Q_{i_2}} \neq \emptyset$; then we can add a new query corresponding to this intersection $Q_{i*} = \left(q_{i_1} \wedge q_{i_2}, I_{Q_{i_1}} \cap I_{Q_{i_2}}\right)$.
2. Recommend ontology refinement based on the queries performed over the system. As an example, consider the case when some class is never searched by any query, it might well be denoted obsolete or redundant; i.e., if $\forall I, I_{Q_i} \cap C_j = \emptyset$, then we may propose to remove C_j from the ontology.

Remaining properties. The support for logical operators is considered in the definition of the rule system, and the autopoietic behavior is a property that is related to the second order reasoning over the ontology and the alignment with third-party tools generating synonymy, equivalent concept matching, statistical and fuzzy analysis.

EXTENDED REFLEXIVE ONTOLOGIES

In this article, we propose the extended reflexive ontologies (ROX) whose main feature is the maintenance of the rule and recommendation history

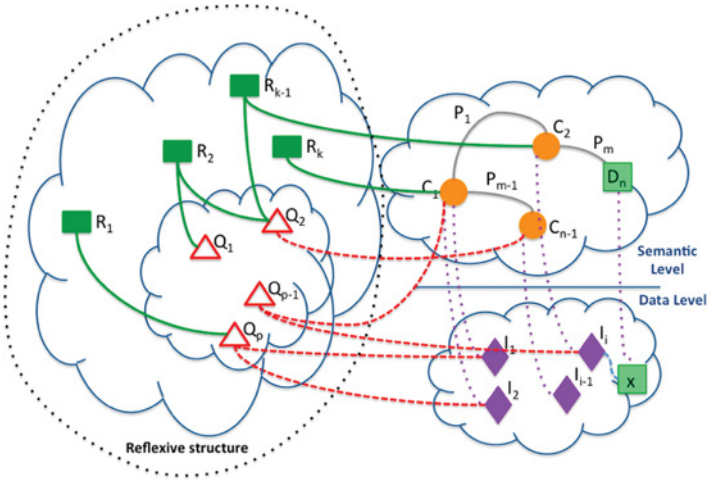

FIGURE 2 Extended reflexive ontologies.

along with the query history already kept by the RO. Figure 2 shows the structure of the ROX.

A ROX is a tuple $ROX = < RO, R^t >$, where RO is a reflexive ontology and $R = \{R_1, R_2, \ldots, R_{K-1}, R_K\}$ the historical set of rules applied at least once to obtain a recommendation. Let a rule recommendation R_K be defined as a tuple $R_K = < r_k, u_k >$, where

1. r_k is a rule such that $r_k = < A_k, S_k, L_k, W_k, B_k >$, where the antecedent A_k is a query, $Q_k = \left(q_k, I_{Q_k}\right)$, and consequents S_k and L_k are corresponding actions taken in the THEN and ELSE parts of the rule, and
2. u_k are the output domain assignments associated to r_k, where $u_k = \left\{\left(I_U^d, V_d, v_d\right)\right\}$ is a collection of domain variable value assignments V_d-v_d, where d is the domain indicator and I_U^d is a set of individuals affected by the domain value assignment $I_{k'} \cdot V_d = v_d, \forall I_{k'} \in I_U^d$.

GENERATION OF RECOMMENDATIONS OVER EXTENDED REFLEXIVE ONTOLOGIES

The ROX approach stores every rule r_k applied to the ontology into a pool of rules that have been applied $R = \{R_1, R_2, \ldots, R_{K-1}, R_K\}$, such that $R_K = < r_k, u_k >$, $r_k = < A_k, S_k, L_k, W_k, B_k >$, and generated domain recommendations $u_k = \left\{\left(I_U^d, V_d, v_d\right)\right\}$. The basic recommendations generation by a reasoner $\Re$ is performed by taking as input a request $J = (I_J, D_J)$, the current extended reflexive ontology, ROX^t, and ruleset R. Then the reasoner $\Re$ outputs a set of

recommendations $K = \{K_{ij}\}$ for each request $J_i = (I_i, D_i)$. A recommendation is a tuple $K_{ij} = < G_{ij}, W_{ij}, R^{Kij} >$, as defined previously.

The K_{ij} is built by first analyzing the stored rules in R. After that, the process recalls the reasoning on the remaining rules R–R^t. For each $I_i \in I_J$, we follow this process:

1. For each u_k in R^t such that $I_i \in I_{u_k}$, we have two situations:
   a. a previously created recommendation K_{ij} such that its recommendation G_{ij} refers to the same v_d as u_k, then we add the rule r_k and weight W_k to $R^{K_{ij}}$ and $W_{G_{ij}}$, respectively;

   b. otherwise, we create recommendation K_{ij} such that its recommendation G_{ij} is $V_d = v_d$, the rule set $R^{K_{ij}} = \{r_k\}$, and weight $W_{G_{ij}} = W_k$.
2. For each r_k in R- R, if $M(I_i, A_k)$ we compute a new recommendation K_{ij} with $G_{ij} = (V_d, v_d)$ as specified by the consequent of rule r_k, the rule set $R^{K_{ij}} = \{r_k\}$, and weight $W_{G_{ij}} = W_k$. Also, we update the rule pool of the ROX, as follows: $R^{t+1} = R^t \cup \{(r_k, u_k)\}$, with $u_k = \{(I_i, V_d, v_d)\}$.
3. After computing the recommendations for the given collection $J = (I_J, D_J)$, we might need to perform a compaction process in R^{t+1} because we may have some redundant u_k, which differ only in I_{u_k} and can be compacted into one.

IMPLEMENTATION OF EXTENDED REFLEXIVE ONTOLOGIES

The classes and properties needed for extending an ontology O as a ROX are depicted in Figure 3, and are described as

- Classes
  - Class **Rule** is the object storing each rule R_i applied.
  - Class **Query** is the object storing each query Q_i performed to the base ontology.
  - Class **QuerySpecification** is the object storing each query specification q_i.
  - Class **OutputRecommendation** is the object storing a couple individuals/values, (I^u, v).
- Object type properties
  - Property *ruleOutputsRecommendations* relates **Rule** instances with instances of **OutputRecommendation**. It is an inverse functional property.
  - Property *ruleHasAntecedent* relates **Rule** instances with **Query** instances. It is a functional property, because each rule contains a unique antecedent.

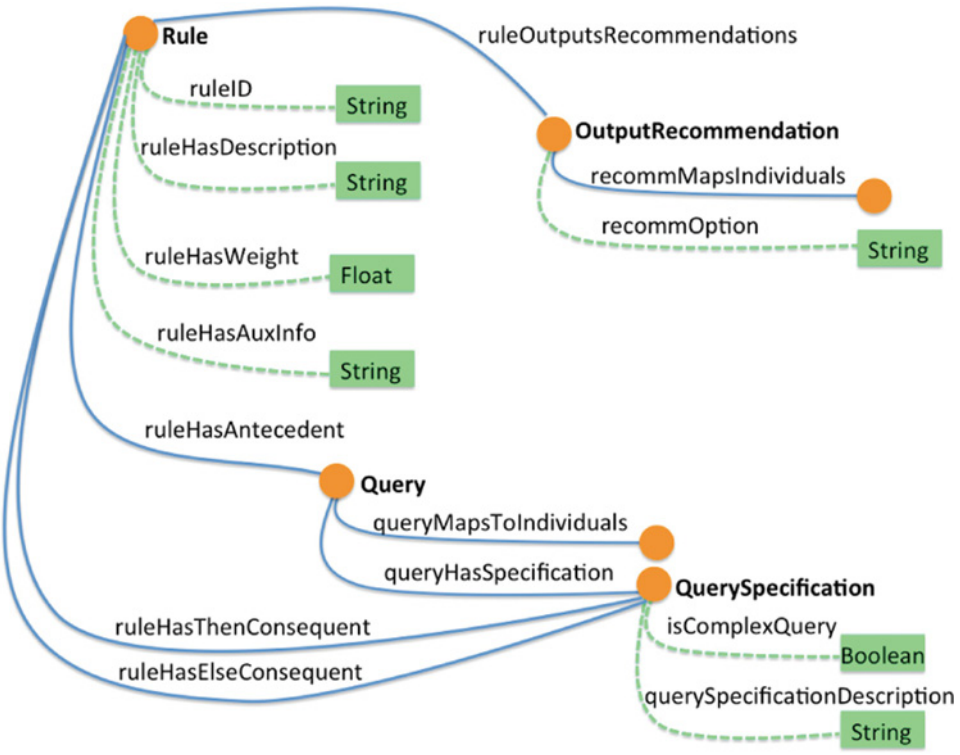

FIGURE 3 Implementation details of extended reflexive ontologies.

- Property *ruleHasThenConsequent* relates **Rule** instances with **QuerySpecification** instances. It is a functional property, because each rule contains a unique rule then-type consequent.
- Property *ruleHasElseConsequent* relates **Rule** instances with **QuerySpecification** instances. It is a functional property, because each rule contains a unique rule else-type consequent.
- Property *recommendationMapsToIndividuals* relates instances of **OutputRecommendation** with instances of classes of the base ontology.
- Property *queryMapsToIndividuals* relates **Query** instances with instances of classes of the base ontology.
- Property *queryHasSpecification* relates **Query** instances with **QuerySpecification** instances. It is a functional property, because each query has a unique specification.

- Datatype properties
  - Property *ruleID* relates **Rule** instances with String data values. It is a functional property, because each rule has a unique ID.
  - Property *ruleHasDescription* relates **Rule** instances with String data values. It is a functional property, because each rule has a unique description.

- Property *ruleHasWeight* relates **Rule** instances with Float data values. It is a functional property, because each rule has a unique weight.
- Property *ruleHasAuxInfo* relates **Rule** instances with String data values.
- Property *recommendationOption* relates **OutputRecommendation** instances with String data values. It is a functional property, because each option has a unique recommended value.
- Property *isComplexQuery* relates **QuerySpecification** instances with Boolean data values. It is a functional property.
- Property *querySpecificationDescription* relates **QuerySpecification** instances with String data values. It is a functional property, because each query has a unique specification.

The implementation of such extension is done by applying the Protégé-OWL API,[1] which is based on the JENA Ontology API.[2] First, the URI corresponding to the target ontology is opened and the OWL model is retrieved. Then, each of the four classes above (i.e., **Rule, Query, QuerySpecification**, and **OutputRecommendation**) are created. Following, the object type properties (i.e., *ruleOutputsRecommendations, ruleHasAntecedent, ruleHasThenConsequent, ruleHasElseConsequent, recommendationMapsToIndividuals, queryMapsToIndividuals*, and *queryHasSpecification*) and the data type properties (i.e., *ruleID, ruleHasDescription, ruleHasWeight, ruleHasAuxInfo, recommendationOption, isComplexQuery* and *querySpecificationDescription*) described previously are created. Then, the generated ontology is saved. Finally, the ROX can be fed with instances.

CONCLUSIONS AND FUTURE WORK

In this article we presented a specification of the fast-querying technique Reflexive Ontologies. We also proposed an extension of such technique to allow fast rule-based reasoning. We called our new approach Extended Reflexive Ontologies (ROX). The enhancement of an ontology in providing self-contained rules and recommendations relies on the following aspects:

1. *Speed up of the process of recommendation generation*. Each rule r_k, as well as the recommendations u_k provided to each decisional domain d by r_k, are both stored in the ROX. Thus, when applying a rule that is already contained in ROX, recommendations do not need to be recalculated. They are calculated only in the case where the rule has never been applied before and are then added to the rule reflexivity class.

[1] goo.gl/NmGirH.

[2] Jena Home Page: goo.gl/Zru2Ct.

3. *Incremental nature of ROX*. From the analysis of the previously applied rules and the corresponding attached actions, new rules could be discovered and added to ROX. In this article, such analysis is performed by experience-mining processes executed over a history of stored decisional events.

The application of ROX in Clinical Decision Support Systems (CDSS) provides a considerable speed up of the process of generation of decision recommendations. Particularly during patient-recommendations generation, many rules are applied to the underlying knowledge bases of the CDSS. Because rules tend to be the same for every patient, each time a new patient data is introduced in ROX, the applying rules and recommendations are automatically calculated by the reasoner ℜ. Thus, when requesting for recommendations, they will be readily available.

REFERENCES

Aleksovska-Stojkovska, L. and S. Loskovska. "Architectural and Data Model of Clinical Decision Support System for Managing Asthma in School-Aged Children." Paper presented at the IEEE International Conference on Electro/Information Technology (EIT), Mankato, MN, 15–17 May 2011.

Berner, E. S. and T. J. La Lande. "Overview of CDSS." In *Clinical Decision Support Systems: Theory and Practice* (2nd ed.), edited by E. S. Berner, 3–22. New York, NY, USA: Springer-Verlag, 2007.

Bouamrane, M. M., A. L. Rector, and M. Hurrell. "Development of an Ontology for a Preoperative Risk Assessment Clinical Decision Support System." Paper presented at the 22nd IEEE International Symposium on Computer-Based Medical Systems, Albuquerque, NM, August 2009.

Ceccarelli, M., A. Donatiello, and D. Vitale. "Kon3: A Clinical Decision Support System, in Oncology Environment, Based on Knowledge Management." *IEEE 20th International Conference on Tools with Artificial Intelligence* 2, (2008): 206–210.

Greenes, R. A. *Clinical Decision Support: The Road Ahead*. Burlington, MA: Elsevier Science, 2011.

Guarino, N. and P. Giaretta. "Ontologies and Knowledge Bases: Towards a Terminological Clarification." In *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, 25–32. Amsterdam, Netherlands: IOS Press, 1995.

Kahn, C. E. Jr., L. M. Roberts, K. Wang, D. Jenks, and P. Haddawy. "Preliminary Investigation of a Bayesian Network for Mammographic Diagnosis of Breast Cancer." Paper presented at the Proceedings of the Annual Symposium on Computer Applications in Medical Care, New Orleans, LA, October 28–November 1, 1995. 208–212. 1995.

Kalogeropoulos, D. A., E. R. Carson, and P. O. Collison. "Towards Knowledge-Based Systems in Clinical Practice: Development of an Integrated Clinical Information and Knowledge Management Support System." *Computer Methods and Programs in Biomedicine* 72, no. 1 (2003): 65–80.

Liu, J., J. C. Wyatt, and D. G. Altman. "Decision Tools in Health Care: Focus on the Problem, Not the Solution." In *BMC Medical Informatics and Decision Making* 6, no. 4 (2006).

Maturana, H. R. and F. J. Varela. "Autopoiesis and Cognition: The Realization of the Living." In *Boston Studies in the Philosophy and History of Science*, 42. Berlin: Springer, 1980.

Miller, P. L., S. J. Frawley, F. G. Sayward, W. A. Yasnoff, L. Duncan, and D. W. Fleming. "Imm/Serve: A Rule-Based Program for Childhood Immunization." Paper presented at the AMIA Annual Fall Symposium, Washington, DC; 26–30 October, 1996. 184–188. 1996.

Power, D. J. "Decision Support Systems: A Historical Overview." In *Handbook on Decision Support Systems, Part I – Foundation of Decision Support Systems*, 121–140. Berlin: Springer, 2008.

Toro, C., E. Sanchez, E. Carrasco, L. Mancilla-Amaya, C. Sanin, E. Szczerbicki, M. Graña, P. Bonachela, C. Parra, G. Bueno, F. Guijarro. "Using Set of Experience Knowledge Structure to Extend a Rule Set of Clinical Decision Support System for Alzheimer's Disease Diagnosis." *Cybernetics and Systems* 43, no. 2 (2012): 81–95.

Toro, C., C. Sanin, E. Szczerbicki, and J. Posada. "Reflexive Ontologies: Enhancing On- tologies with Self-Contained Queries." *Cybernetics and Systems: An International Journal* 39, (2008): 171–189.

Warner, H. R. Jr. "Iliad: Moving Medical Decision-Making into New Frontiers." In *Methods of Information in Medicine* 28, no. 4 (1989): 370–372.

Wicht, A., T. Wetter, and U. Klein. "A Web-Based System for Clinical Decision Support and Knowledge Maintenance for Deterioration Monitoring of Hemato-Oncological Patients." *Computer Methods and Programs in Biomedicine* 111, no. 1 (2013): 26–32.