Innovation in Medicine and Healthcare 2014
M. Graña et al. (Eds.)
IOS Press, 2014
© 2014 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-61499-474-9-234

Specification of Extended Reflexive Ontologies in the context of CDSS

Eider SANCHEZ [a,b,c,1], Carlos TORO [a,b], Manuel GRAÑA [c], Cesar SANÍN [d] and Edward SZCZERBICKI [d]

[a] *Vicomtech-IK4 Research Centre, San Sebastian, Spain*
[b] *Biodonostia Health Research Institute, eHealth Group, Bioengineering Area, San Sebastian, Spain*
[c] *University of the Basque Country UPV/EHU, Computational Intelligence Group, Computer Science Faculty, San Sebastian, Spain*
[d] *School of Engineering, The University of Newcastle, Newcastle, Australia*

Abstract. 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 towards the solution of the reported needs. The starting point of our approach is the work of Toro et al. [13] on Reflexive Ontologies (RO). We also propose an extension of RO, by including the handling and reasoning that production rules provide. Our approach speeds-up the recommendation generation process.

Keywords. Reflexive ontologies, fast query system, rule engine, autopoiesis

1. Introduction

Clinical Decision Support Systems (CDSS) are active intelligent systems that use patient clinical data to generate case specific advice [9]. According to [5], 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 analysis to perform some action, often the generation of recommendations. CDSS cover a wide span of tools based on several technologies and approaches. Particularly, [12] identifies knowledge-based CDSS (k-CDSS) as tools with specialized problem-solving expertise which allows them to provide decision recommendations to users. Decision recommendations are a set of alternative options for actions (e.g. diagnosis) that have been calculated by the system according to previously established crite-

[1]Corresponding Author: Eider Sanchez, Vicomtech-IK4 Research Centre, Mikeletegi Pasealekua 57, 20009 San Sebastian, Spain; E-mail: esanchez@vicomtech.org

ria. Recommendations are ranked and presented to system users, so that they can easily analyze the different suggested choices, as well as their proofs. In this sense k-CDSS act as black boxes that output decision recommendations for a given input data set. They benefit from a symbolic representation of knowledge about a particular domain, and the ability for reasoning about solutions of problems within that domain [8].

In the context of 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 generation of 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 that will convey towards the solution of the reported needs. The starting point of our approach is the work of Toro et al. [13] on Reflexive Ontologies (RO). We propose an extension of RO, by including the handling and reasoning that production rules provide.

This paper is structured as follows: Section 2 introduces CDSS and Reflexive Ontologies. Section 3 presents an specification of the underlying knowledge model of the CDSS. Section 4 details the process of generation of decision recommendations. Section 5 proposes a specification of Reflexive Ontologies. Section 6 proposes a specification of Extended Reflexive Ontologies. Section 7 presents the reasoning process generation over the Extended Reflexive Ontologies paradigm. Finally, Section 8 discusses some relevant aspects of our approach.

2. Background concepts

2.1. Knowledge-based CDSS

Knowledge-based CDSS have been broadly reported in the literature. Examples are the Bayesian reasoning for general CDSS Iliad, presented by Warner [14]; the diagnostic mammography system Mammonet based on bayesian networks presented by Kahn et al [7]. Amongst other works based on production rules we can mention the IMM/Serve immunological CDSS built by Miller et al [11]. More recently, [15] presented a web based CDSS that follows a case-based approach, in which editors were provided for knowledge manual maintenance; [1] described an architectural and data model for CDSS, integrated to the clinical system; [4] presented a knowledge-based CDSS for Oncology, where both an ontology and a ruleset were proposed; in [3] an OWL DL ontology 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 et al. [2], consists of 4 elements: (i) an input, (ii) an output, (iii) a Knowledge Base and (iv) a reasoning engine.

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 stablished.

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 less likely, and the most save 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.

Knowledge Base The Knowledge Base consists of some form of medical knowledge. The representation of such knowledge may be obtained by means of applying 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 [6]. 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.

Reasoning Engine The Reasoning Engine combines the input data and the medical knowledge, according to some logical scheme, for generating the output.

2.2. *Reflexive Ontologies*

Reflexive Ontologies (RO) define an abstract knowledge structure (i.e. an ontology and its instances) endowed with the capacity of maintaining an updated image of every query performed [13]. 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 [13], following the biological inspiration of Maturana in his seminal work [10]. Figure 1 shows the logical structure of a RO, which is, basically, a conventional ontology extended with a reflexive structure (mainly composed by 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).

Amongst others, RO is based on the concept of autopoiesis, meaning "self-creation" or "self-production". The RO display an autopoietic behavior since 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, for instance, 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 [10].

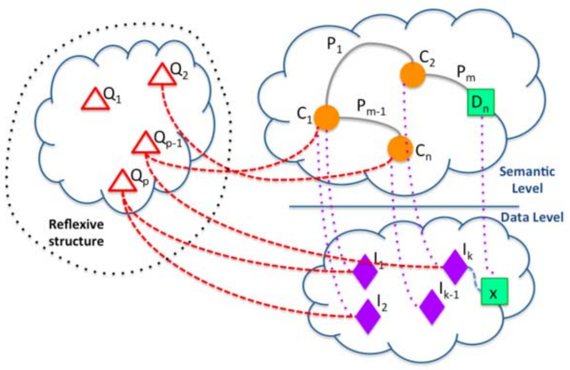

Figure 1. Schematic representation of the structure of a RO

3. Knowledge model specification

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

3.1. Domain Ontology

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

- A class C_i 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.

3.2. 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 $Q_i = (q_i, I_{Q_i})$ where q_i are the clauses specifying the characteristics of the search, and $I_{Q_i} \subset I$ is the subset of the ontology individuals matching the query clauses. In fact, a query is a map of the form:

$$\sigma(q_i) = I_{Q_i} = \{I_k \in I \,|M(q_i, I_k)\}, \tag{1}$$

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 $q_i^s = \langle V_i, m_i, v_i \rangle$, where V_i is a variable, m_i is the comparison operator (i.e. $>, <, =$) and v_i 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. $\vee$, $\wedge$ and $\neg$) which define the relationships among consecutive simple queries:

$$q_i^c = \{(\theta_n, q_n^s)\}_{\forall n}, \tag{2}$$

where θ_n is the n-th logical operator (i.e. $\vee$, $\wedge$ and $\neg$), assuming $\theta_0 = \emptyset$.

3.3. 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 assignment or recommendations. A rule r_k is composed of a query clause and the consequent actions. Each rule is formalized as a tuple $r_k = \langle A_k, S_k, L_k, W_k, B_k \rangle$, where

(i) A_k is the set of conditional clauses (antecedents), that are equivalent to the q_i part of the queries,

(ii) S_k is the set of actions corresponding to the **THEN** consequents,

(iii) L_k is the set of actions corresponding to the **ELSE** consequents,

(iv) W_k is the rule salience (aka weight), defined as a real number $W_K \in [0,1]$, and

(v) B_k is a generic notation for application-dependent ancillary information that can be associated to rule.

A special kind of action is the assignment of a value to a variable, i.e. $V_l = v_l$. 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 towards 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 towards some d_i and both consequents, S_k and L_k, must refer to the same set of d_i.

4. Generation of recommendations

When inputted a request $J = (I_J, D_J)$, where $I_J \subset I$ are a set of individuals for which recommendations are requested, and $D_J \subset D$ 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 $K_{ij} = \langle G_{ij}, W_{ij}, R^{K_{ij}} \rangle$ computed in response to a couple (I_i, d_i) where:

- The recommendation consequent G_{ij}, which 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 $W_{G_{ij}} \in [0,1]$ computed for recommendation G_i,
- A subset of rules $R^{K_{ij}} = \{r_k | M(A_k, I_i)\}, R^{K_{ij}} \subset R$, that provide 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 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. $r_k \in R^{K_{ij}}$.

- 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 $I_{q_i} \cup \overline{I_{q_i}} = \mathrm{I}$ and $I_{q_i} \cap \overline{I_{q_i}} = \varnothing$.
- 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}.
- On the other hand, 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 $R^{K_{ij}}$.

5. Specification of Reflexive Ontologies

In this Section, we present the first actual attempt to provide a formal specification of Reflexive Ontologies, which, though remaining abstract, is concrete enough to discuss the consequences and degree of implementation.

A reflexive ontology RO is a tuple $RO = \langle O, Q^t \rangle$, where O is a domain ontology and $Q^t = \{Q_1, Q_2, ..., Q_{N-1}, Q_N\}$ 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 edition 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.

$$\neg\exists q_i \in Q^t \text{ s.t. } q_i = q_{i^*} \Longrightarrow Q^{t+1} = Q^t \cup \{Q_i\}. \tag{3}$$

On the other hand, if the query has been already posted and answered, an updated answer will be provided

$$\exists q_i \in Q^t \text{ s.t. } q_i = q_{i^*} \Longrightarrow I_{Q_{i^*}} = I_{Q_i}. \tag{4}$$

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 k-th 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:

$$\forall q_i \in Q^t \text{ s.t. } M\left(q_i, I_k^t\right) \Longrightarrow I_k^t \in I_{Q_i}^t \tag{5}$$

This specification is purely declarative. If we want to advance something on the mechanism that may 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 $I_k^{t-1} = \varnothing$ and $I_k^t \neq \varnothing$. Also, the following holds always $M(\varnothing, q_i) = F$. Hence, the integrity update can be specified as follows:

$$I_k^{t-1} \neq I_k^t \Longrightarrow \begin{cases} \neg M\left(q_i, I_k^{t-1}\right) \wedge M\left(q_i, I_k^t\right) & I_{Q_i}^t = I_{Q_i}^{t-1} \cup \left\{I_k^t\right\} \\ M\left(q_i, I_k^{t-1}\right) \wedge M\left(q_i, I_k^t\right) & I_{Q_i}^t = I_{Q_i}^{t-1} - \left\{I_k^{t-1}\right\} \cup \left\{I_k^t\right\}. \\ M\left(q_i, I_k^{t-1}\right) \wedge \neg M\left(q_i, I_k^t\right) & I_{Q_i}^t = I_{Q_i}^{t-1} - \left\{I_k^{t-1}\right\} \end{cases} \tag{6}$$

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:

i) discover patterns of queries. As an example, assume that some pair of queries Q_{i_1} and Q_{i_2} have a non empty intersection of their corresponding data instances, i.e. $I_{Q_{i_1}} \cap I_{Q_{i_2}} \neq \varnothing$, then we can add a new query corresponding to this intersection $Q_{i^*} = \left(q_{i_1} \wedge q_{i_2}, I_{Q_{i_1}} \cap I_{Q_{i_2}}\right)$.

ii) 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 may well be denoted obsolete or redundant, i.e. if $\forall i, I_{Q_i} \cap C_j = \varnothing$ 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.

6. 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 along with the query history already keep by the RO. Figure 2 shows the structure of the Extended Reflexive Ontologies (ROX).

A ROX is a tuple $ROX = \langle RO, \mathscr{R}^{\mathrm{t}}\rangle$, where RO is a Reflexive Ontology and $\mathscr{R}^t = \{\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_{K-1}, \mathscr{R}_K\}$ the historical set of rules applied at least once to obtain a recommendation.

Let a rule-recommendation $\mathscr{R}_K$ be defined as a tuple $\mathscr{R}_K = \langle r_k, u_k\rangle$, where

(i) r_k is a rule such that $r_k = \langle A_k, S_k, L_k, W_k, B_k\rangle$, 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

(ii) u_k are the output domain assignments asociated 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, I_U^d is a set of individuals affected by the domain value assignment $I_{k'}.V_d = v_d, \forall I_{k'} \in I_U^d$.

7. Generation of recommendations over Extended Reflexive Ontologies

The Extended Reflexive Ontologies approach stores every rule r_k applied to the ontology, into a pool of rules that have been applied $\mathscr{R}^t = \{\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_{K-1}, \mathscr{R}_K\}$, such that $\mathscr{R}_k = \langle r_k, u_k\rangle$, $r_k = \langle A_k, S_k, L_k, W_k, B_k\rangle$, and generated domain recommendations $u_k = \left\{\left(I_{u_k}^d, V_d, v_d\right)\right\}$.

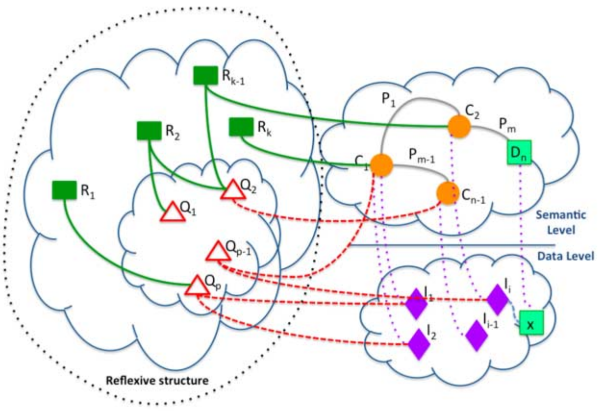

Figure 2. Extended Reflexive Ontologies

The basic recommendations generation by a reasoner $\mathfrak{R}$ 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 $\mathfrak{R}$ outputs a set of recommendations $K = \{K_{ij}\}$ for each for each request $J_i = (I_i, d_i)$. A recommendation is a tuple $K_{ij} = \langle G_{ij}, W_{ij}, R^{K_{ij}} \rangle$ as defined above.

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

1. For each u_k in $\mathscr{R}^t$ such that $I_i \in I_{u_k}$ we have two situations
   (a) we have a previusly 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 $R - \mathscr{R}^t$, 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 $W_{G_{ij}} = W_k$. Besides we update the rule pool of the ROX, as follows, $\mathscr{R}^{t+1} = \mathscr{R}^t \cup \{(r_k, u_k)\}$, with $u_k = \{(I_i, V_d, v_d)\}$.

After computing the recommendations for the given collection $J = (I_J, D_J)$, we may need to perform a compaction process in $\mathscr{R}^{t+1}$ because we may have some redundant u_k which differ only in I_{u_k} and can be compacted into one.

8. 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 in the following aspects:

- 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 Extended Reflexive Ontology (ROX). Thus, when applying a rule that is already contained in ROX, recommendations do not need to be recalculated. They are only calculated in the case where the rule has never been applied before and are then added to the rule reflexivity class.
- 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.

In the context of this article 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. As 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 $\mathfrak{R}$. Thus, when requesting for recommendations, they will be readily available.

References

[1] L. Aleksovska-Stojkovska and S. Loskovska. Architectural and data model of clinical decision support system for managing asthma in school-aged children. In *Electro/Information Technology (EIT), 2011 IEEE International Conference on*, pages 1–6, 2011.

[2] Eta S. Berner and Tonya J. La Lande. *Clinical Decision Support Systems, Theory and Practice*, chapter Overview of Clinical Decision Support Systems. Number 1. Springer, New York, second edition, 2007.

[3] Matt-Mouley Bouamrane, Alan L. Rector, and Martin Hurrell. Development of an ontology for a preoperative risk assessment clinical decision support system. In *CBMS*, pages 1–6. IEEE, 2009.

[4] Michele Ceccarelli, Antonio Donatiello, and Dante Vitale. Kon3: A clinical decision support system, in oncology environment, based on knowledge management. *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, 2:206–210, 2008.

[5] R.A. Greenes. *Clinical Decision Support: The Road Ahead*. Elsevier Science, 2011.

[6] N. Guarino and P. Giaretta. Ontologies and Knowledge Bases: Towards a Terminological Clarification. *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32, 1995.

[7] C E Jr Kahn, L M Roberts, K Wang, D Jenks, and P Haddawy. Preliminary investigation of a bayesian network for mammographic diagnosis of breast cancer. *Proc Annu Symp Comput Appl Med Care*, pages 208–212, 1995.

[8] Carson E.R. Collison P.O. Kalogeropoulos, D.A. Towards knowledge-based systems in clinical practice: Development of an integrated clinical information and knowledge management support system. *Computer Methods and Programs in Biomedicine*, 72:65–80, 2003.

[9] Joseph Liu, Jeremy C Wyatt, and Douglas G Altman. Decision tools in health care: focus on the problem, not the solution. *BMC Med Inform Decis Mak*, 6(4), Jan 2006.

[10] Varela F.J. Maturana, H.R. *Autopoiesis and Cognition: The Realization of the Living*, volume 42 of *Boston Studies in the Philosophy and History of Science*. Springer, 1980.

[11] P L Miller, S J Frawley, F G Sayward, W A Yasnoff, L Duncan, and D W Fleming. Imm/serve: a rule-based program for childhood immunization. *Proc AMIA Annu Fall Symp*, pages 184–188, 1996.

[12] D.J Power. *Decision Support Systems: A Historical Overview*, chapter Part I – Foundation of Decision Support Systems, pages 121–140. Handbook on Decision Support Systems. Springer, 2008.

[13] Sanín C. Szczerbicki E. Posada J. Toro, C. Reflexive ontologies: Enhancing ontologies with self-contained queries. *Cybernetics and Systems: An International Journal*, 39:171–189, 2008.

[14] H R Jr Warner. Iliad: moving medical decision-making into new frontiers. *Methods Inf Med*, 28(4):370–372, Nov 1989.

[15] Andreas Wicht, Thomas Wetter, and Ulrike 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(1):26–32, 2013.