Representing, Querying, and Visualizing Health-Insurance Knowledge in a Cost-Sharing Estimator

Zainab Almugbel[1] and Harold Boley[2]

[1] Department of Computing, University of Dammam, Dammam, Saudi Arabia
zhalmugbel[AT]uod[PT]edu[PT]sa
[2] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada
harold[PT]boley[AT]unb[PT]ca

Abstract. Cost-Sharing Estimator (CostShEs) 1.0 is a prototype for health-insurance knowledge representation, querying, and visualization. CostShEs 1.0 allows client-side consultation about the cost of a health-care service for a client with/without insurance. The input to this prototype system is a form-based inquiry by a client about the cost of a service. Based on the client's entered membership number, desired service, etc., CostShEs 1.0 generates a logical query. The output is the calculated cost of the service, optionally as a text or a visualization. For logical reasoning, CostShEs 1.0 requires (1) an RDFS taxonomy, (2) webized Rule Markup Language (RuleML) data, and (3) RuleML decision rules. The taxonomy helps the system to find the client's relatives that can take advantage of his/her insurance. The webized RuleML data are relational ground facts that link to the relevant information on web pages, such as a client's web page. Based on (1) and (2), the decision rules answer the query. These three components form the knowledge base that CostShEs uses for reasoning with the Object Oriented Java Deductive Reasoning Engine for the Web (OO jDREW). The visualization of the system's output in Graph Inscribed Logic (Grailog) is obtained via GrailogKSViz 2.0 translation of RuleML/XML to Scalable Vector Graphics (SVG/XML).

1 Introduction

Increasing numbers of insurances play a major role for individuals, groups, and organizations, hence for the world economy, in attempts to capture largely unpredictable events. Insurance is defined here as a form of risk management primarily used to guard against the risk of conditional loss. Health insurance is the most common insurance in today's health-concerned societies. This is because it is devised to protect individuals and families against the increasing and often unexpected costs of healthcare services [IMA13].

As an example, the coverage of essential health services has increased in Saudi Arabia, according to the latest World Health Organization (WHO) figures. For instance, family planning has increased from 58.9% in 2000 to 70.5% in 2015; likewise, child immunization, from 95% to 98% for the same interval. In addition, the latest figures show that there were 26.5 hospital beds for each 10,000 persons in 2014. These factors indeed influence the demand for healthcare services.

In an attempt to cover the whole population, public healthcare can be combined with private healthcare, although the latter may not be affordable to the majority of people. Thus, cooperative health-insurance companies may be a solution. In this type of insurance, three main parties cooperate: the insured, the insurance company (insurer), and the service provider (hospital). This model can provide the insured (beneficiary) with a health service at a lower cost. Such lower cost may be the result of "cost sharing", referring to the portion of health-service costs not borne by the insurer but by the insured. This "out-of-the-client's-pocket" share of costs generally includes deductibles, coinsurance, and copayments, or similar charges.[3] In order to obtain insurance, the insured must pay a "premium", i.e. the amount charged by the insurer to the insured. For our purpose, the insured is any individual who obtains insurance via directly registering at an insurance company. For example, according to the second article of "Implementing Regulation of the Cooperative Health Insurance Law"[4], employees of the public sector cannot take advantage of cooperative health insurance unless they personally register for it.

There have been various Artificial Intelligence efforts in insurance. Some are related to uncertainty calculations in insurance, specifically regarding risks and premiums [AV13]. Other AI approaches have been used for life insurance processing and application assessment [PH15]. The present work focuses on knowledge representation, specifically for webized knowledge as used in the Semantic Web for designing linked (open and corporate) data. For this reason, linked data is discussed here from two perspectives: its languages and its applications.

Re languages, the importance of the Resource Description Framework (RDF) and RDF Schema (RDFS) for webizing is elaborated in [HvH08]. They state that RDF is designed to take advantage of Unique Resource Identifiers (URIs) in knowledge representation, thus enabling linked data. For instance, Positional-Slotted Language (POSL) [Bol10] is building on concepts of linked data. Webizing in POSL includes individuals, slot names, types, functions, and relations. POSL uses a presentation syntax corresponding to the Hornlog subset of the XML-based web rule language RuleML [Bol16][5].

Re applications, Cloudant[6] provides a webized database for global sharing by transforming data into RDF triples. Since the friend-of-a-friend schema is employed, webizing is mostly used for classes and property (slot) names. Another application is the Linked Data Fragments[7] project. This project modifies the concept of client-server architecture. It aims to explore what happens when data load is distributed between clients and servers in order to obtain an "intelligent client". This "intelligent client" is a web browser that allows users' queries without

[3] https://www.healthcare.gov/glossary/cost-sharing/
[4] http://www.sama.gov.sa
[5] http://wiki.ruleml.org/index.php/RuleML_Home
[6] https://developer.ibm.com/dwblog/2014/webizingdatabasejson/#top
[7] http://client.linkeddatafragments.org/

depending only on data stored in a server. It comes with several interfaces to measure the impact of the redistributed linked data on servers, clients and caches.

Furthermore, linked data can be used to enhance the web's linking mechanism because it enables the reuse of data between web applications that were not connected previously [Ver14].

The Cost-Sharing Estimator prototype system[8] attempts to determine how much a client would need to pay for the costs of a health service: full, partial, or none, via a webized knowledge base. It presumes the following:
1- Data and knowledge is available for describing the required aspects of insurers, hospitals, and insureds.
2- This data and knowledge is used to determine the classes, slots, and web references of the represented RuleML data facts from the websites (of insurers, hospitals, and insureds).
3- A taxonomy of relatives is used to help CostShEs in deciding which client's relatives are covered by his/her insurance.
4- Knowledge about cost sharing is available and is formalized as RuleML rules for the system.
5- The Java-based reasoning engine (OO jDREW) is used for reasoning with the taxonomy, data, and rules.

2 CostShEs 1.0 Design

The design of Cost-Sharing Estimator (CostShEs) 1.0 involves two sections: system design and knowledge base design. The system design consists of three main processes shown in Fig. 1. CostShEs 1.0 starts when the user fills out an input form and submits a cost-sharing inquiry. The first process, "Logic Generator", obtains RuleML data facts from several web pages. The selection of these data facts is based on the "Input Form". It also uses the "Input Form" to generate the RuleML query. The second process, "Answer Generator", finds the RuleML answers for the user inquiry. The last process, "Visualization Generator", transforms the RuleML answers into Scalable Vector Graphics (SVG/XML) via GrailogKSViz 2.0. The knowledge base design takes the three main parties of the cooperative health-insurance companies into account.

Based on the first, second, and fifth articles of "Implementing Regulation of the Cooperative Health Insurance Law", the client's insurance must cover his/her dependent(s). Such an eligible dependent is defined in the Law as being partitioned into "spouse", "son up to the age of twenty-five" and "unmarried daughter". In order to expand the insurance coverage to include these different types of persons, an RDFS taxonomy is used to classify "Person" into "Insured" vs. "NonInsured". The "Insured" class partitions into "Self" (the client) and (the client's) "Dependent"(s). Finally, "Dependent" partitions into "Spouse", "Son"

[8] The experimental CostShEs 1.0 system was prototyped for research in health-insurance knowledge representation and processing rather than for actual consulting in, e.g., health insurance. The exemplary facts (data) and rules (programs) in this document are used for explanatory purposes only.

and "Daughter" (the age and marital-status constraints on the latter two will be added as rules in Subsection 3.3). The "Insured"-complementing "NonInsured" class is not given subclasses in the taxonomy because it includes all other relatives of the client, e.g. a sister, and all other persons.

In general, the knowledge base contains the data for and relationships amongst the three mentioned parties. On top of these data facts, it contains rules that realize the decision model of Fig. 2. These are used to calculate the cost of a health service by taking three main decisions.

- The first decision is about whether the client has insurance.
  1. If no, the client has to make full payment.
  2. If yes, the second decision, which is regarding the provider (hospital) of the health service, is taken. It includes checking if the hospital is a partner of the insurance company.
     - If no, the client has to make full payment.
     - If yes, the third decision regarding the cost-sharing amount (payment) is taken by comparing the compensation (the amount of money paid by the insurance company) with the treatment cost:
       (a) If the compensation is greater than or equal to the treatment cost, the client does not make any payment.
       (b) If the compensation is less than the treatment cost, the client makes partial payment (cost sharing).

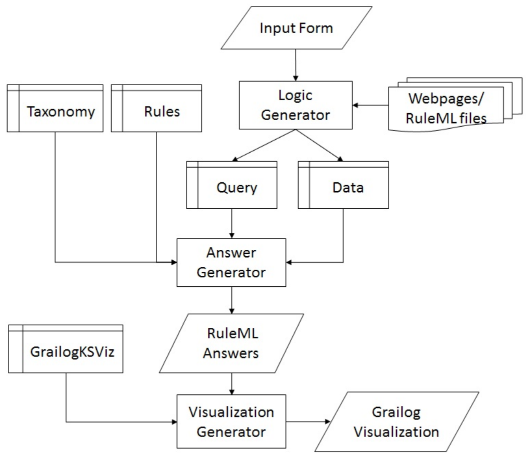


Fig. 1. CostShEs 1.0 system design.

3 CostShEs 1.0 Implementation

CostShEs 1.0 uses an RDFS taxonomy, a webized RuleML subset for relational data, and Naf Datalog RuleML for decision rules. For the webized relational

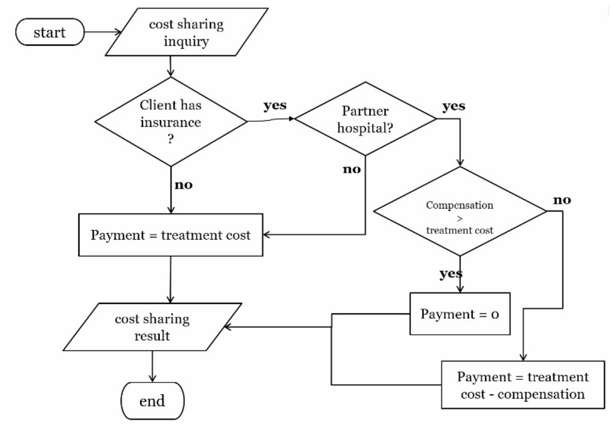


Fig. 2. CostShEs 1.0 decision model.

data, it uses RuleML ground-fact relationships with the optional XML attribute "iri" on "Ind" arguments of relations. The attribute values extend RuleML by linking to properties via CURIEs[9]. When CostShEs 1.0 identifies a CURIE that itself is a URL property, it dereferences its URL content to parse the RuleML atomic facts that are stored at the retrieved web page. Thus, in our knowledge base, data facts use only ground (variable-less) atoms, while queries can use non-ground (variable-containing) atoms, and decision rules (implications) can use non-ground atoms anywhere in their conditions (bodies) and conclusions (heads). Fig. 3 illustrates the hierarchical structure of facts and rules.

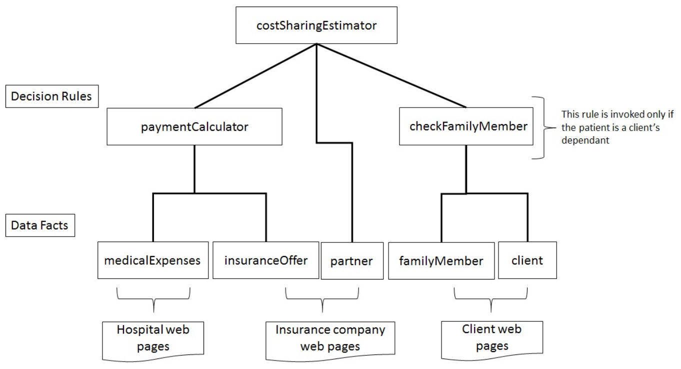


Fig. 3. Hierarchical structure of facts and rules.

[9] https://www.w3.org/TR/curie/

The following subsections describe the taxonomy, and the atomic facts (data) in three main kinds of web pages (for hospitals, insurance companies, and clients).

3.1 Taxonomy

The RDFS taxonomy represents the categories of insurance coverage beneficiaries. For instance, the following RDFS class descriptions illustrate that both "Daughter" is subClassOf "Dependent" and "Dependent" is subClassOf "Insured". Therefore, a client's daughters may be covered by the parent's insurance. However, there are some additional constraints, e.g. a daughter must be unmarried to receive her parent's insurance coverage. Those additional constraints are considered as part of the decision rules (see Subsection 3.3).

```
<rdf:Description rdf:ID="Dependent">
 <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Insured"/>
</rdf:Description>

<rdf:Description rdf:ID="Daughter">
 <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Dependent"/>
</rdf:Description>
```

3.2 Web Page Data Facts

The data facts include medicalExpenses, insuranceOffer, client, and partner. They are represented in RuleML/XML for linked-data access on three kinds of web pages, as explained in the following subsections.

Hospital These pages contain (applications of) the medicalExpenses predicate, thus storing the expenses/price of receiving a service in a specific hospital. In particular, the following ground atom about an xray service for teeth illustrates the arguments of the medicalExpenses predicate (in RuleML/XML, but omitting namespace declarations, here and later on):

```
<Atom>
  <Rel>medicalExpenses</Rel>
  <Ind iri="h:provider">saudiGermanHospital</Ind>
  <Ind iri="m:bodyPart">teeth</Ind>
  <Ind iri="h:serviceType">xray</Ind>
  <Ind iri="h:priceRange" type="Integer">300</Ind>
</Atom>
```

This fact has four arguments: the provider where the client will get the health service, what body part needs the service, the kind of required service, and the service costs. The example thus means that it costs 300 Saudi Riyal to take an xray of teeth in the Saudi German Hospital.

Insurance Company These pages contain two predicates. First, the insuranceOffer predicate stores the available health-insurance offers of a specific insurance company. These insurance offers are categorized based on the insurance type: silver, gold, and platinum. For example, the following ground atom illustrates the arguments of the insuranceOffer predicate:

```
<Atom>
  <Rel>insuranceOffer</Rel>
  <Ind iri="o:provider">tawuniya</Ind>
  <Ind iri="pm:programName">silver</Ind>
  <Ind iri="m:bodyPart">teeth</Ind>
  <Ind iri="h:priceRange" type="Integer">1000</Ind>
</Atom>
```

This fact has four arguments: (name of) insurance company, type of insurance, what body part the insurance covers, and the compensation for treating the specified body part. The example means that if a client has silver insurance from Tawuniya, the company compensates up to 1000 Saudi Riyal for treating his/her teeth.

Second, the partner predicate stores the url of the parties involved in the partnership between hospitals and insurance companies. For example, the following ground atom illustrates the arguments of the partner predicate:

```
<Atom>
  <Rel>partner</Rel>
  <Ind iri="o:legalName">tawuniya</Ind>
  <Ind iri="o:url">http://localhost:90/tawuniya.xml</Ind>
  <Ind iri="h:legalName">dallahHospital</Ind>
  <Ind iri="h:url">http://localhost:90/dallahHospital.xml</Ind>
</Atom>
```

This fact has four arguments: the name and the url of partners. This means there is a partnership between Tawuniya and the Dallah Hospital; therefore, Tawuniya covers its clients' treatment in the Dallah Hospital.

Client These pages contain two predicates. First, the client predicate stores information about the membership program a client may have with a specific insurance company. For example, the following ground atom illustrates the arguments of the client predicate:

```
<Atom>
  <Rel>client</Rel>
  <Ind iri="pm:membershipNumber">11122</Ind>
  <Ind iri="p:name">ali</Ind>
  <Ind iri="pm:hostingOrganization">tawuniya</Ind>
  <Ind iri="pm:url">http://localhost:90/tawuniya.xml</Ind>
  <Ind iri="pm:programName">silver</Ind>
  <Ind iri="pm:description">health Insurance membership</Ind>
</Atom>
```

This fact has six arguments: the membership number for the registered client, the client name, the insurance company name, its url, the client's insurance type, and a description about the membership.

Second, the familyMember predicate stores information about the client's relatives. For example, the following ground atom illustrates the arguments of the familyMember predicate:

```
<Atom>
  <Rel>familyMember</Rel>
  <Ind iri="pm:membershipNumber">11122</Ind>
  <Ind iri="p:children">child</Ind>
  <Ind iri="p:gender">male</Ind>
  <Ind iri="p:givenName">saleh</Ind>
  <Ind iri="p:birthDate" type="Integer">1992</Ind>
  <Ind iri="c:maritalStatus">married</Ind>
</Atom>
```

This fact has six arguments: the membership number for the registered client and information about the relative. This includes the relationship between the relative and the client, the relative's gender, his/her name, his/her date of birth, and his/her marital status.

3.3 Decision Rules

Besides the main predicate, costSharingEstimator, there are several other predicates, including paymentCalculator and checkFamilyMember, all defined by rules for decision-making on different levels. The costSharingEstimator, checkFamilyMember, and paymentCalculator decision rules will be discussed in detail in the following subsections.

costSharingEstimator The predicate costSharingEstimator uses seven named variables (indicated by a "?" prefix) for estimating the cost-sharing ?Payment for a client ?Name with a ?clientMembershipID, who may have medical insurance at ?InsCompany and needs a ?RequiredTreatment for a ?BodyPart in a ?Hospital. One of its decision rules thus has this head (also using an anonymous variable, serialized as an empty XML element, to branch on its type, here "Self"):

```
<Atom>
  <Rel>costSharingEstimator</Rel>
  <Var>clientMembershipID</Var>
  <slot><Ind>insured</Ind><Var type="Self"/></slot>
  <Var>Name</Var>
  <Var>InsCompany</Var>
  <Var>Hospital</Var>
  <Var>BodyPart</Var>
  <Var>RequiredTreatment</Var>
  <Var type="Integer">Payment</Var>
</Atom>
```

The costSharingEstimator predicate is defined by three rules based on the type of the insurance-coverage beneficiaries:

1. `type="Self"` means the client is insured and the treatment is for him/herself (Rule 1)
2. `type="Dependent"` means the client is insured and the treatment is for one of his/her dependents (Spouse, Son, Daughter) (Rule 2)
3. `type="NonInsured"` means the client is not insured (Rule 3)

costSharingEstimator Rule 1 and Rule 2. These rules realize the decision model specified in Fig. 2. Although Rule 1 and Rule 2 look almost the same, e.g. using the same facts of Subsection 3.2, there are critical differences as shown in Table 1.

Table 1. Differences between Rule 1 and Rule 2

	Rule 1	*Rule* 2
Rule type:	type="Self"	type="Dependent"
Head arguments:	for the client	for the dependent
Chained subrules:	paymentCalculator	paymentCalculator + checkFamilyMember

costSharingEstimator Rule 3. This rule is invoked when the client is not insured – as determined via type="NonInsured" – at any insurance company. The client has to pay the full cost of the service.

checkFamilyMember This predicate's three rules take advantage of the taxonomy by checking, in their bodies, if a family member would have the client's insurance coverage. The common arguments of the heads of these rules expose three variables: the client membership id, the relationship between the client and the family member, and the family member's name. These variables are bound based on the "Input Form". The below sample head invokes the rule for the client's spouse to check if he/she is eligible for his/her insurance coverage.

```
<Atom>
  <Rel>checkFamilyMember</Rel>
  <Var>clientMembershipID</Var>
  <slot><Ind>insured</Ind><Var type="Spouse"/></slot>
  <Var>name</Var>
</Atom>
```

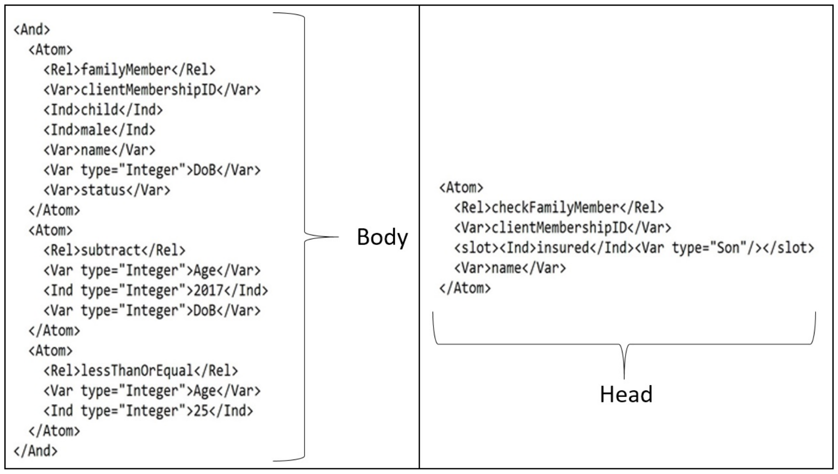
CostShEs employs a checkFamilyMember rule for each insured dependent type (Spouse, Son, and Daughter) because of different constraint levels applicable to these subclasses:

1. The spouse has the client's insurance coverage without any constraints (Rule 1).
2. The son has the client's insurance coverage if his age is less than or equal to 25 years (Rule 2). This rule is further discussed below.
3. The daughter has the client's insurance coverage if she is unmarried (Rule 3).

checkFamilyMember Rule 2. This rule, shown in Fig. 4, executes when the client is not submitting an inquiry for him/herself but for one of his/her sons. On the "Input Form", this is achieved by changing, for the label "Patient:", the default value "Self" to the value "Son". The value of the label "Patient Name:" is optional. This means if the client enters a value, CostShEs checks the specified son; otherwise all eligible sons are returned. The rule's body conjointly queries the fact of Subsection 3.2 and two relational built-ins as follows:

1. The familyMember call retrieves the male child (son) of an insured client.
2. The ternary relational subtract built-in binds a variable as the first argument to the result for the second and third arguments.
3. The binary relational lessThanOrEqual built-in is used here to compare two integer numbers.

This and other rules, as part of the entire open knowledge base, are linked in Section 5.



```
<And>
  <Atom>
    <Rel>familyMember</Rel>
    <Var>clientMembershipID</Var>
    <Ind>child</Ind>
    <Ind>male</Ind>
    <Var>name</Var>
    <Var type="Integer">DoB</Var>
    <Var>status</Var>
  </Atom>
  <Atom>
    <Rel>subtract</Rel>
    <Var type="Integer">Age</Var>
    <Ind type="Integer">2017</Ind>
    <Var type="Integer">DoB</Var>
  </Atom>
  <Atom>
    <Rel>lessThanOrEqual</Rel>
    <Var type="Integer">Age</Var>
    <Ind type="Integer">25</Ind>
  </Atom>
</And>
```

```
<Atom>
  <Rel>checkFamilyMember</Rel>
  <Var>clientMembershipID</Var>
  <slot><Ind>insured</Ind><Var type="Son"/></slot>
  <Var>name</Var>
</Atom>
```

Fig. 4. The head and body of checkFamilyMember Rule 2.

paymentCalculator This rule is chained to from the costSharingEstimator Rule 1. Its rules have the head:

```
<Atom>
  <Rel>paymentCalculator</Rel>
  <Var type="Integer">Price</Var>
  <Var type="Integer">Compensation</Var>
  <Var type="Integer">Payment</Var>
</Atom>
```

This head contains three variables: ?Price (the cost of the health service), ?Compensation (the money paid by the insurer), and ?Payment (the money paid by the client). The body of the paymentCalculator rule calculates the cost-sharing ?Payment by comparing ?Price with ?Compensation as shown in Fig. 2.

4 Querying the CostShEs Knowledge Base

This section discusses the exploration of the CostShEs knowledge base of Subsections 3.2 and 3.3 through the CostShEs querying screen of Fig. 5.

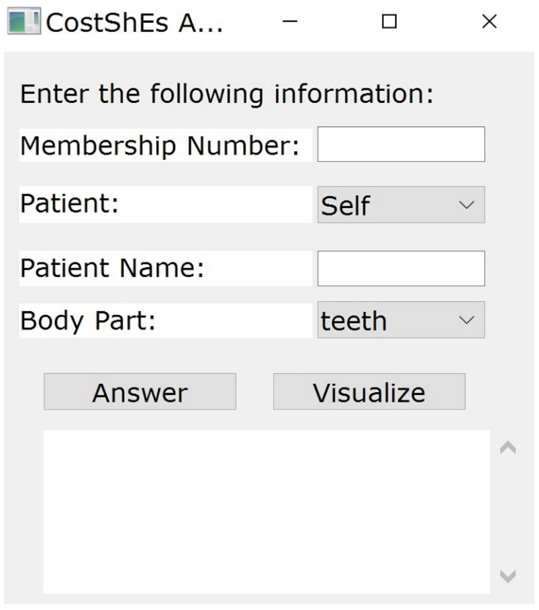


Fig. 5. Main screen of CostShEs 1.0.

A CostShEs 1.0 knowledge base is run using queries to OO jDREW 1.0[10]. The answers to the user's inquiry can be visualized with Grailog 1.0 [Bol13]. Grailog embodies a systematics for visualizing logical knowledge representations by generalized graphs. In Grailog, an atom is visualized by a directed hyperarc arrow starting from a labelnode for its predicate/relation, cutting through nodes for its non-last arguments (e.g., constants and variables), and pointing to a node for its last argument. CostShEs 1.0, besides variable values, employs ground-instantiated atoms as answers, visualized as Grailog hyperarcs. The main benefit of Grailog is that its resulting visual representations of knowledge sources are much easier to read – hence validate – for domain experts than the original symbolic representation. Moreover, the transformation rules for the graphical-to-symbolic mapping are easy to learn and remember. CostShEs can use Saxon[11] to feed RuleML/XML answer atoms into the XSLT-based translator GrailogKSViz 2.0 [Bid16] to produce Scalable Vector Graphics (SVG/XML) for rendering those answers as Grailog diagrams in modern browsers.

Sample Query 1. Figs. 6 and 7 show the RuleML answers and their visualization for a query submitted by the client with membership id= 11122. It is for finding all possible hospitals and payments for treating teeth for the insured client. The answers show exactly the partner hospitals that provide the treatments along with the payments.

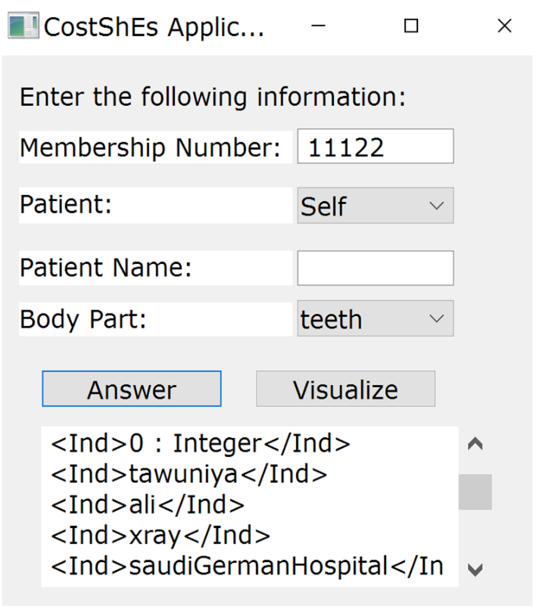
Sample Query 2. Figs. 8 and 9 show the RuleML answers and their visualization for a query submitted by a non-insured client with membership id= 89001. It is for finding all possible hospitals and payments for treating him/her. In this case, the client can receive the treatment in any hospital, and he/she has to pay the full cost of the service.

Sample Query 3. Figs. 10 and 11 show the RuleML answers and their visualization for a query submitted by an insured client with membership id= 11122 to check which of his/her daughters (Sara and Nourh) is covered by his/her insurance. For this query, CostShEs first retrieves the client's daughters who are registered in his/her web page as female children. Then, it applies an additional constraint to only retain the unmarried daughters. Here, the answers show that the only unmarried daughter (Sara) has the client's insurance coverage.

5 Conclusions

Research in health-insurance knowledge representation, querying, and visualization has been conducted via the Cost-Sharing Estimator prototype CostShEs. The CostShEs knowledge base supports calculating the cost of a healthcare service for a client with/without insurance. The CostShEs 1.0 knowledge is represented in the RuleML/XML serialization syntax. This syntax has been used for

[10] http://jdrew.org/oojdrew/1.0/oojdrew-1.0-SNAPSHOT-jar-with-dependencies.jar
[11] http://www.saxonica.com



Enter the following information:

Membership Number: 11122

Patient: Self

Patient Name:

Body Part: teeth

```
<Ind>0 : Integer</Ind>
<Ind>tawuniya</Ind>
<Ind>ali</Ind>
<Ind>xray</Ind>
<Ind>saudiGermanHospital</In
```

Fig. 6. Query 1: RuleML answers.

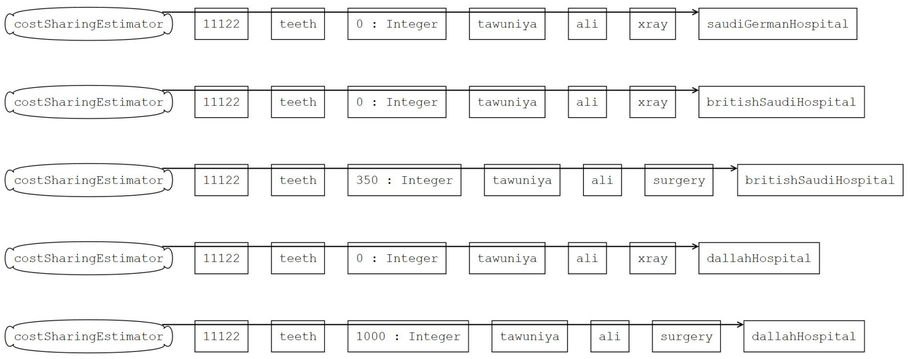
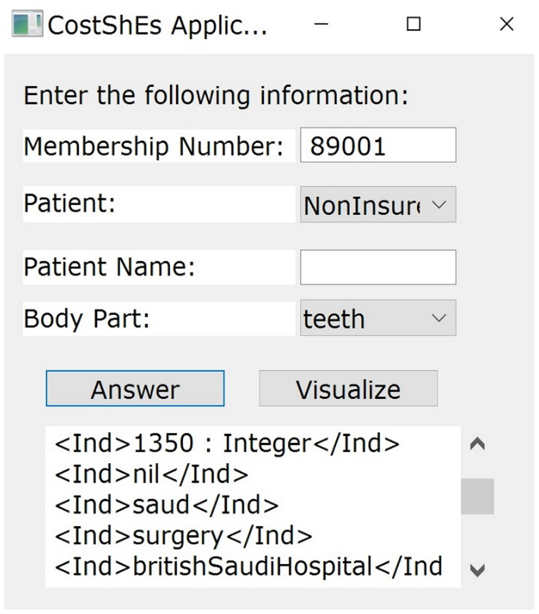


Fig. 7. Query 1: Grailog visualization of answers.


CostShEs Applic...

Enter the following information:

Membership Number: 89001

Patient: NonInsur

Patient Name:

Body Part: teeth

```
<Ind>1350 : Integer</Ind>
<Ind>nil</Ind>
<Ind>saud</Ind>
<Ind>surgery</Ind>
<Ind>britishSaudiHospital</Ind
```

Fig. 8. Query 2: RuleML answers.

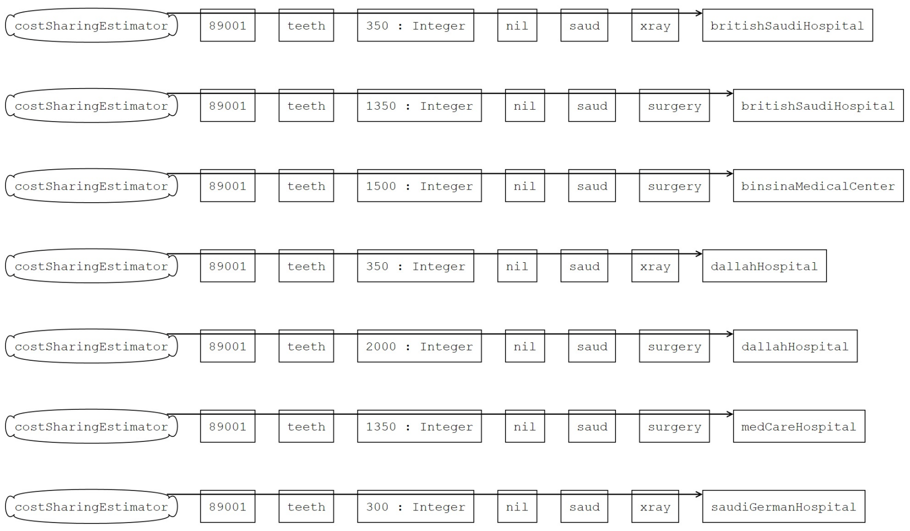
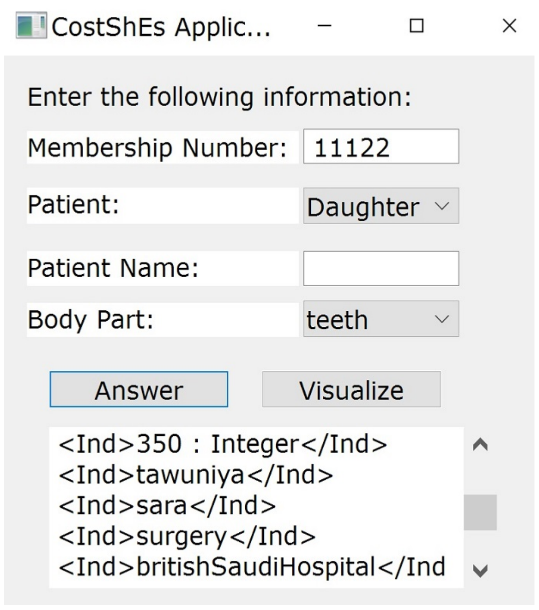

costSharingEstimator 89001 teeth 350 : Integer nil saud xray britishSaudiHospital
costSharingEstimator 89001 teeth 1350 : Integer nil saud surgery britishSaudiHospital
costSharingEstimator 89001 teeth 1500 : Integer nil saud surgery binsinaMedicalCenter
costSharingEstimator 89001 teeth 350 : Integer nil saud xray dallahHospital
costSharingEstimator 89001 teeth 2000 : Integer nil saud surgery dallahHospital
costSharingEstimator 89001 teeth 1350 : Integer nil saud surgery medCareHospital
costSharingEstimator 89001 teeth 300 : Integer nil saud xray saudiGermanHospital

Fig. 9. Query 2: Grailog visualization of answers.



Enter the following information:

Membership Number:	11122
Patient:	Daughter
Patient Name:	
Body Part:	teeth

```
<Ind>350 : Integer</Ind>
<Ind>tawuniya</Ind>
<Ind>sara</Ind>
<Ind>surgery</Ind>
<Ind>britishSaudiHospital</Ind
```

Fig. 10. Query 3: RuleML answers.

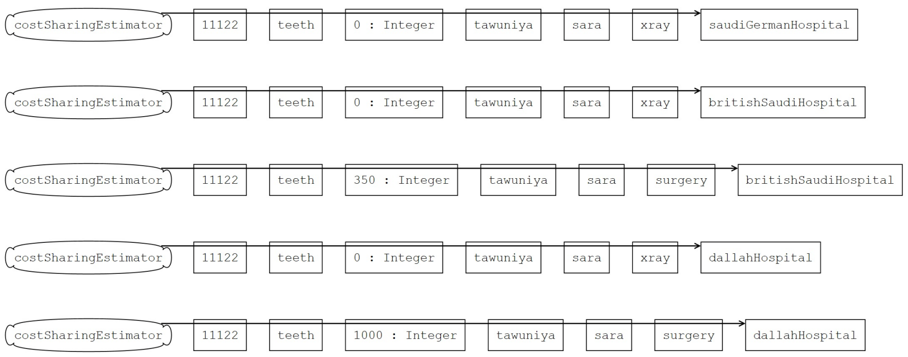

costSharingEstimator 11122 teeth 0 : Integer tawuniya sara xray saudiGermanHospital
costSharingEstimator 11122 teeth 0 : Integer tawuniya sara xray britishSaudiHospital
costSharingEstimator 11122 teeth 350 : Integer tawuniya sara surgery britishSaudiHospital
costSharingEstimator 11122 teeth 0 : Integer tawuniya sara xray dallahHospital
costSharingEstimator 11122 teeth 1000 : Integer tawuniya sara surgery dallahHospital

Fig. 11. Query 3: Grailog visualization of answers.

OO jDREW querying. The XML serialization also makes it directly amenable to the XSLT-based GrailogKSViz 2.0 translator for generating SVG/XML rendered as Grailog visualization. Feedback through the Grailog visualization has supported improving the CostShEs knowledge content.

CostShEs 1.0 underwent an initial assessment by a medical insurance agent at Aim Gulf Insurance Brokers. The agent focussed on four main aspects of CostShEs 1.0, giving the following feedback:

1. The CostShEs 1.0 research prototype is not meant for actual advice-giving, because it does not contain the complete information for advising a client. For example, the current subcategories of services and each service's price range should be much refined.
2. The knowledge base constitutes a good reference source about clients and is well organized. However, the agent had difficulty in interpreting the positions of relations with many arguments such as costSharingEstimator, whose output variables have no slot names.
3. The form-based UI is easy to use, and it simplifies knowledge base access.
4. The output is not precise because: (1) the class of the policy, e.g. the insurance offer, is not shown; (2) the policy approval limit, as well as the effective and the expiry dates of the policy, are missing. The Grailog visualization is much easier to read than the plain RuleML answers.

The CostShEs 1.0 **system**, implemented in Java, and **knowledge base** are open source, and available online[12].

6 Future Work

CostShEs 1.0 can be improved in various ways, e.g. by addressing the agent's feedback in Section 5. The CostShEs 1.0 knowledge base can be extended to contain more facts about different medical insurance companies and rules about further cases of decision-making. Generalizing CostShEs 1.0's use of a fixed payment per service, cost sharing could use a percentage, with a maximum limit. For enriched background knowledge, a future CostShEs version could also reuse relevant health/insurance ontologies and rules from open repositories. The resulting release should undergo a complete evaluation. A consolidated release could then support users in their selection of a suitable insurance. Additional insurance types (e.g., travel insurance) could later be formalized by similar systems. Besides visualizing relational answer atoms, GrailogKSViz 2.0 could be extended to visualize slotted answer atoms and an existing Grailog visualization for CostShEs KBs could be extended from Datalog rules to Naf Datalog rules. Concurrently, RuleML/XML and OO jDREW could be further developed. Finally, employing a RuleML$\leftrightarrow$POSL translator pair [Bol10], the knowledge base could be translated from RuleML/XML, acting as a canonical language [Bol16], to POSL, and, employing, e.g., XSLT-based translators, to (sub)languages for use by engines other than OO jDREW, including by PSOATransRun [ZB15] and Prolog engines.

[12] `https://github.com/zainabAlmugbel/CostShEs.git`

7 Acknowledgements

We thank the reviewers of the International Rule Challenge 2017 for helpful comments. Special thanks go to Mrs. Leah Bidlake for her creation of GrailogKSViz 2.0, to Mr. Ali Alzoury for his initial assessment of the CostShEs 1.0 system, and to Mr. Alawi Almogbel for his suggestions about CostShEs improvements.

References

[AV13] Nidhi Arora and Sanjay K Vij. Reckoner for health risk and insurance premium using adaptive neuro-fuzzy inference system. *Neural Computing and Applications*, 23(7-8):2121–2128, 2013.

[Bid16] Leah Bidlake. Grailog KS Viz 2.0: Graph-Logic Knowledge Visualization by XML-Based Translation. Master's thesis, University of New Brunswick, 2016.

[Bol10] Harold Boley. Integrating Positional and Slotted Knowledge on the Semantic Web. *Journal of Emerging Technologies in Web Intelligence*, 4(2):343–353, November 2010.

[Bol13] Harold Boley. Grailog 1.0: Graph-Logic Visualization of Ontologies and Rules. In Leora Morgenstern and Adrian Paschke, editors, *Proc. 7th International Web Rule Symposium: Research Based and Industry Focused (RuleML 2013), Seattle, Washington, USA*, volume 8035 of *Lecture Notes in Computer Science*, pages 52–67. Springer, July 2013.

[Bol16] Harold Boley. The RuleML Knowledge-Interoperation Hub. In José Júlio Alferes, Leopoldo E. Bertossi, Guido Governatori, Paul Fodor, and Dumitru Roman, editors, *Rule Technologies. Research, Tools, and Applications - 10th International Symposium, RuleML 2016, Stony Brook, NY, USA, July 6-9, 2016. Proceedings*, volume 9718 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2016.

[HvH08] Jim Hendler and Frank van Harmelen. The semantic web: webizing knowledge representation. *Foundations of Artificial Intelligence*, 3:821–839, 2008.

[IMA13] Ashikul Islam, Rob McKim, and Erez Allouche. Fuzzy logic system approach for quantifying risk associated with the installation of trenchless technologies. In *Pipelines 2013: Pipelines and Trenchless Construction and Renewals – A Global Perspective*, pages 1458–1467. 2013.

[PH15] Jim Prentzas and Ioannis Hatzilygeroudis. Assessment of life insurance applications: an approach integrating neuro-symbolic rule-based with case-based reasoning. *Expert Systems*, 2015.

[Ver14] Ruben Verborgh. *Serendipitous Web Applications through Semantic Hypermedia.* PhD thesis, Ghent University, Ghent, Belgium, February 2014.

[ZB15] Gen Zou and Harold Boley. PSOA2Prolog: Object-Relational Rule Interoperation and Implementation by Translation from PSOA RuleML to ISO Prolog. In *Proc. 9th International Web Rule Symposium (RuleML 2015), Berlin, Germany*, Lecture Notes in Computer Science. Springer, August 2015.