The Social Semantic Subweb of Virtual Patient Support Groups

Harold Boley[1], Omair Shafiq[2], Derek Smith[3], and Taylor Osmun[3]

[1] Institute for Information Technology, National Research Council Canada
Fredericton, NB, Canada, harold.boley AT nrc.gc.ca
[2] Department of Computer Science, University of Calgary
Calgary, AB, Canada, moshafiq AT ucalgary.ca
[3] Faculty of Computer Science, University of New Brunswick
Fredericton, NB, Canada, {i14dy, w91pq} AT unb.ca

Abstract. Patients increasingly interact in support groups, which provide shared information and experiences about diseases, treatments, etc. Much of this interaction is mediated by the Social Web, allowing worldwide reach but lacking in semantic precision. We present an online prototype, PatientSupporter, to create a Social Semantic Subweb that will facilitate high-precision networking between patients based on ontologies and rules. PatientSupporter is an instantiation of Rule Responder that permits each patient to query other patients' profiles for finding or initiating a matching group. Rule Responder's External Agent (EA) is a Web-based patient-organization interface that passes queries to the Organizational Agent (OA). The OA represents the shared knowledge of the virtual patient organization, delegates queries to relevant Personal Agents (PAs) via a responsibility ontology, and passes checked PA answers back to the EA. Each PA represents the medical subarea of primary interest to an associated patient group. The PA assists its patients by advertising their interest profiles, employing rules about diseases and treatments as well as interaction constraints such as time, location, age range, gender, and number of participants. Profiles can be distributed across different rule engines using different rule languages (e.g., Prolog and N3), where rules, queries, and answers are interchanged via translation to and from RuleML/XML. We discuss the implementation of PatientSupporter in a use case of sports injuries structured by a part-whole ontology of affected body parts.

1 Introduction

Social Web (Web 2.0) techniques have been explored in recent years for applications in healthcare [14, 13]. Web 2.0 portals such as PatientsLikeMe[1] and (part of) samestory[2] have been developed to help patients to network with other, geographically distributed patients having similar ailments to discuss and exchange

[1] http://www.patientslikeme.com/
[2] http://www.same-story.com/sante-maladies/

information and experiences. Successful 'Patient 2.0' portals typically have good recall when searching for other patients but lack in precision.

Semantic techniques increase this networking precision, leading to the following Social Semantic Web (Web 3.0) approach to patient portals. We introduce ontologies and rules for organizing patients – here, with sports injuries – into virtual support groups around classes of an ontology of injuries – here, a commonsense partonomy for localizing sports injuries. Of course, this can only *complement* the diagnosis and therapy of diseases by medical experts – it reflects new patients' use of commonsense knowledge, rather than expert knowledge, to *find* similar patients as well as relevant literature and medical professionals.

Our initial online prototype, PatientSupporter[3], is designed to start the Social Semantic Subweb for patients by demonstrating how patients with a sports injury could be helped to find or initiate a virtual support group about that injury. Patients in an online PatientSupporter virtual organization create their semantic profile referring to classes in a disease ontology – here a partonomy of body parts affected by sports injuries. This body partonomy allows patients to base the description of their injuries on a `subPartOf` hierarchy leading to affected body parts, which is implemented as a corresponding `subClassOf` taxonomy of injury classes for those body parts. Profiles contain rules about body-part diseases and treatments as well as interaction constraints such as time, location, age range, gender, and number of participants. A patient can pose queries against the semantic profiles of other patients in his or her virtual organization to find or initiate a matching group. PatientSupporter is built upon Rule Responder [19, 9], which has also been used, e.g., in the related Social Semantic Web instantiation WellnessRules [8] and in SymposiumPlanner [12].

PatientSupporter allows patients to have their profiles expressed in either Pure Prolog [20] (Horn logic rules) or N3 [2] (graph production rules). Providing these quite different rule language paradigms permits virtual organizations or individual patients to base their PatientSupporter use on the paradigm that best suits them. Rule Responder handles the interoperation between different rule languages of patients through translators to and from RuleML/XML as the interchange format [10].

As an example, let us consider a patient with nickname Paul, who has injured part of his left leg during a rugby game. He has questions about his lesion and precautions for recovery, which others with similar lesions may be able to answer or help with. Since he lives in a small town where he knows no one else with such an injury he looks for online support.[4] Using PatientSupporter, Paul poses a query through the External Agent (EA), focusing on leg lesions. The EA submits the query to the Organizational Agent (OA), which delegates it to the Personal Agent (PA) of the leg-injury group, and checks the answers from the profiles (local knowledge bases) of its participating patients. When the OA returns many answers to the EA, Paul discovers that his query was too broad.

[3] http://ruleml.org/PatientSupporter/

[4] Other reasons for seeking support in a virtual rather than real group may include increased anonymity (via nicknames) and avoiding contagiousness (e.g., flu).

Paul, who actually hurt his left knee, thus proceeds downward the partonomy by querying PatientSupporter just for patients with knee lesions. Since no knee-injury PA exists, the OA again delegates it to the more general leg-injury PA. But within this PA's group only the knee-injury participants are eligible, hence the answers returned are less in number and more relevant. Paul picks some of the returned nicknames of patients and queries them with his interaction constraints, proposing a knee-injury discussion in a Skype-based conference call on the upcoming Saturday or Sunday any time between 10AM and 6PM EST. Paul's query returns the Skype IDs of patients who want to reveal theirs and are interested in the discussion, with the time narrowed down to Sunday between 3PM and 6PM EST. Hence, Paul invites them for a first call Sunday, 4PM to 5PM, effectively initiating a knee-injury subgroup of the leg-injury group.

It should be noted that Paul by using the PatientSupporter Social Semantic Web portal is able to initiate the virtual subgroup about his sports injury on a global scale. He also benefits from PatientSupporter's interoperation facility in the background – to transform patient profiles between Pure Prolog and N3 through RuleML/XML. The system employs a partonomy of sports-injury-affected body parts, which makes it easy for Paul to navigate hierarchically up or down, increasing recall or precision, respectively. Paul's queries invoke other patients' interaction rules, allowing him to narrow down his search in a step-wise fashion. All of this saves him from browsing through a large set of irrelevant patient profiles and permits him to efficiently converge on a first Skype call.

The rest of the paper is organized as follows. Section 2 presents the design goals of the PatientSupporter instantiation of Rule Responder. Section 3 discusses the global knowledge base used by the OA. Section 4 describes the use of local knowledge bases to represent the profiles of individual patients underneath the PAs. Section 5 expands upon the RuleML-based interoperation of Pure Prolog and N3 rules. Section 6 explains and demonstrates the use of RuleML-based querying for patients in a distributed setting. Section 7 concludes the paper and discusses future work.

2 Instantiating Rule Responder to PatientSupporter

PatientSupporter is based on Rule Responder, where this reference architecture with each of its main agent types (i.e., EA, OA, and PAs) is instantiated as described in the following. It performs virtual support group matchmaking by querying patients organized in a disease ontology – here, a body partonomy. The following design goals have been pursued while developing PatientSupporter:

1. Identify a language of appropriate expressiveness to model patient profiles from the family of RuleML languages [3], Pure Prolog [20], N3 [2], etc.
2. Identify a language for light-weight ontologies of sports injuries such as `subPartOf` partonomies mapped to `subClassOf` taxonomies in RDFS or OWL 2. The ontology language is to be combined with the rule language.

3. Allow eliciting rule and ontology definitions in human-oriented syntaxes, while translating the resulting knowledge to and from RuleML/XML, RIF/XML [7], or XCL2 / CL RuleML[5] for interchange.
4. Allow different rule engines (e.g., OO jDREW[6] [1], Prova[7] [15], and Euler[8]) to execute global and local rulebases.
5. Allow rules as well as queries and their answers to be transmitted over an Enterprise Service Bus (ESB) – e.g., Mule[9] or Apache ServiceMix.
6. Investigate the appropriateness of languages, engines, and GUIs for rules as well as ontologies, to express, process, and transform the knowledge required in patient profiles.
7. Elicit exemplary patient profiles and abstract them to generally usable profile templates for increased usability and reusability.
8. Guide students – e.g., of Computer Science, Medicine, or Kinesiology – when forming and evolving virtual sports-injury support groups with PatientSupporter.
9. Evaluate the effectiveness and usefulness of the distributed PatientSupporter architecture, based on its ESB-interconnected engines using different languages for the dynamic formation of virtual patient support groups.
10. Adapt PatientSupporter from the sports-injury use case to other medical domains such as weight control, food allergies, oral health, or (seasonal) flu.

The current implementation of PatientSupporter has focused on goals 1.-7. It models patient profiles in POSL [5] and N3 [2]. The profiles are interoperated through RuleML/XML as the intermediate format. Enquiring users are aided by an English-XML-bridging menu-based form.[10] Knowlege about patients and their injuries is organized using rules combined with light-weight ontologies in sorted (typed) Horn logic or N3. The `subPartOf` partonomy is mapped to `subClassOf` in RDFS.[11] Human-oriented syntaxes (of POSL and N3) have been used while modeling the patient profiles. The overall communication and coordination of the rule engines (e.g., OO jDREW, Prova, and Euler) has been organized through Mule, an open-source ESB. The use of an ESB allows architectural flexibility by decoupling the functional components of Rule Responder from the communication components [11].

Rule Responder's instantiation to PatientSupporter in the sports-injury domain allows to create virtual patient organizations consisting of virtual support groups that are defined through sports injuries structured by a partonomy of affected body parts (further explained in Section 3). Specifically, the OA becomes an assistant to the entire virtual patient organization. Each PA becomes an assistant to a group of patients having the same class of injuries from the partonomy,

[5] http://wiki.ruleml.org/index.php/Relax_NG
[6] http://www.jdrew.org/oojdrew/
[7] http://www.prova.ws/
[8] http://eulersharp.sourceforge.net/
[9] http://www.mulesoft.org/
[10] http://ruleml.org/PatientSupporter/RuleResponder.html
[11] http://ruleml.org/PatientSupporter/files/PS-Taxonomy.rdf

and helps them as *profile users* to get organized as a support group. The EA is utilized by patients as *enquiry users* to (register with its virtual organization and) query the profiles of the virtual organization's other patients.

Rule Responder employs the following sequence of steps: An enquiry user interacts with the EA to author and submit queries to the OA. The OA assigns (maps and delegates) each query topic to the PA most knowledgeable about it. Each PA poses the query to its local rulebase, and returns the derived answer(s) to the OA. The OA checks the answer(s) before giving them back to the EA, hence to the enquiry user.

By default, the OA does not reveal the identity of the nicknamed patient(s) behind the answering PA(s). Keeping the personal information hidden in this way, the OA acts as a mediator that helps protect the privacy of profiles of patients in the virtual organization. For participating in PatientSupporter-scheduled online discussions via Skype, MSN, etc., or via a (smart)phone, patients might also use dedicated (Skype, MSN, etc.) IDs or phones. However, if support group participants do not want to reveal even their voice, they have to resort to typing via chat or SMS. On the other hand, after a few discussions some within the same virtual support group may decide to reveal their everyday identities to selected or all participants.

Rule Responder's earlier instantiations include SymposiumPlanner [12] and WellnessRules [8], where WellnessRules extended Rule Responder with multiple participant profiles underneath each PA. PatientSupporter further extends the functionality of Rule Responder by making Social Semantic use of partonomies, mapped to taxonomies: Patient injuries are classified in the hierarchical part-whole manner of affected body parts. For example, injuries related to Foot are subordinated to those of Leg. Similarly, Heel and Toe injuries are subordinated to those of Foot. Employing partonomies as light-weight ontologies in this way allows the 'refinement' of a virtual support group (e.g., about Leg injuries) into subgroups (e.g., about Foot injuries and further about Heel and Toe injuries).

PatientSupporter in extension to WellnessRules allows users to be supported by PAs as follows: Each patient (as a profile user) publishes a profile employed by the responsible PAs to respond to queries (from enquiry users) about his or her preferences, constraints, etc. This dynamic profile association is implemented via the Profile Responsibility Matrix (PRM), and is not possible in SymposiumPlanner, where only chairs (as profile users) are supported by PAs.

The main agent types of PatientSupporter are described in the following subsections. Figure 1 depicts the interaction between the EA, OA, and PAs.

2.1 External Agent

The External Agent (EA) is the point-of-contact that allows a patient to query the Organizational Agent (OA) of a virtual patient organization. It is based on a Web interface that allows him or her as an enquiry user to compose queries employing a menu-based form, which uses JavaScript to generate both an English description and RuleML/XML, thus making it easy to query other patients' profiles. A sports-injury patient primarily selects the injury class from the parton-

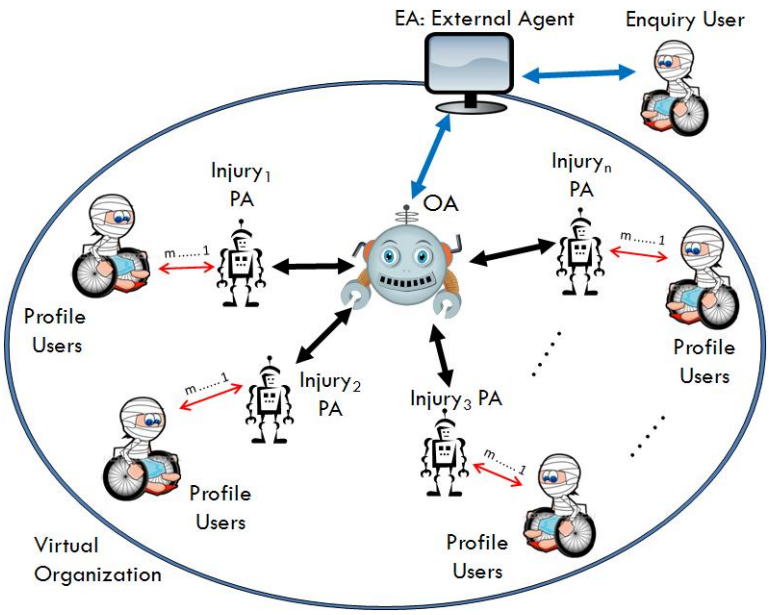

Fig. 1. Overall architecture of PatientSupporter

omy. He or she can then fill in property values about diseases and treatments as well as interaction constraints. The finished RuleML/XML query is submitted to the OA. Finally, the EA presents the OA's answer(s) to the patient.

2.2 Organizational Agent

The Organizational Agent (OA) is at the center of PatientSupporter, representing a virtual patient organization as a whole. The knowledge base of the OA is *global* across the virtual organization, and is written and run in the language and engine Prova. The OA also employs two matrices, on the basis of which incoming queries are mapped and delegated: The *Group Responsibility Matrix*, written as an OWL-Lite ontology, defines which group headed by a PA is best for which kind of query. The *Profile Responsibility Matrix*, written as an XML document, defines which patient profiles exist in a PA's group, and in which formats (here, POSL or N3).

2.3 Personal Agent

The Personal Agents (PAs) contain disease-oriented groups of patient profiles, where diseases are restricted to sports injuries. Each PA heads the group of patient profiles listed in the Profile Responsibility Matrix (cf. Section 2.2). The knowledge base of facts and rules of each profile under a PA is *local* to that profile, and is either written in POSL and run in OO jDREW or is written in N3 and run in Euler.

3 Global Knowledge Base for Virtual Organization

The ontologies and a subset of the rules are globally shared via the OA to benefit all the PAs. Another subset of rules is distributed amongst the PAs, where it is kept local (cf. Section 4). The shared ontology and the shared subset of rules are referred to as the *global knowledge base*, which is complemented by a shared *signature document*.

Global knowledge in PatientSupporter is modeled as a combination of ontologies and rules, where rule arguments are defined by signatures. The ontologies include a light-weight ontology realizing the Group Responsibility Matrix (cf. Section 2.2) and a body partonomy. The global rules include general constraints and preferences of the virtual organization. PatientSupporter makes use of the standard rule format RuleML/XML and the Rule Responder framework to transform to and from other rule languages.

The body partonomy was elicited as a commonsense ontology to reflect the patient-centric perspective of support groups. It is drawing, among others, on the Digital Anatomist Foundational Model of Anatomy (FMA) [18], the online Sports Injury Clinic [21], and the knowledge of a medically trained NRC-IIT colleague. It is referred to as a partonomy because it represents the logical hierarchy of body parts. However, it is realized as a taxonomy of injuries affecting the body parts. Thus, *A* `subPartOf` *B* implies *A Injury* `subClassOf` *B Injury*. Note that we do not express *what* (possibly still undiagnosed) injury it is, but only the (unlateralized etc.) body part *where* it is. This representation is proposed as an appropriate level of abstraction for finding patients with sports injuries, but could be refined for other purposes (cf. Section 7).

Under the root `Body`, PatientSupporter uses the partonomy classes `Head`, `Neck`, `Shoulder`, `Arm`, `Torso`, `Back`, and `Leg`. All of `Thigh`, `Lower Leg`, `Knee`, and `Foot` are regarded as direct parts of `Leg`. `Toe` and `Heel` are likewise part of `Foot`. The complete partonomy is shown in Figure 2. Its implementation as a `subClassOf` taxonomy in RDFS is available online.[11]

The rule component in PatientSupporter employs POSL with Horn logic plus Negation as failure (Naf) and N3 with scoped Naf. The use of Naf Hornlog POSL has been restricted to atoms with positional arguments,[12] leaving F-logic-like frames with property-value slots to N3. This demonstrates the range of our approach through complementary rule styles.[13]

The Naf Hornlog POSL sublanguage uses (positional) n-ary relations (or, predicates) as its central modeling paradigm. N3 instead uses (unordered) sets of binary slots (or, properties) centered around object identifiers (OIDs, called 'subjects' in RDF and N3).

[12] The POSL syntax thus corresponds to pure-Prolog syntax except that POSL variables are prefixed by a question mark while Prolog variables are upper-cased.

[13] To didactically exemplify the positional and slotted styles as well as POSL-N3 interoperation, the online PatientSupporter prototype redundantly keeps rulebases both as `.posl` and as `.n3` documents.

The following POSL example indicates the *positional signature* of the 16-ary predicate `myDiscussion`:

```
myDiscussion(?ProfileID,?Injury,?MinAge,?MaxAge,?MinRSVP,?MaxRSVP,?Category,?Treatment,
             ?HealingStage,?StartTime,?EndTime,?Duration,?Channel,?Contact,?Gender,?TimeZone).
```

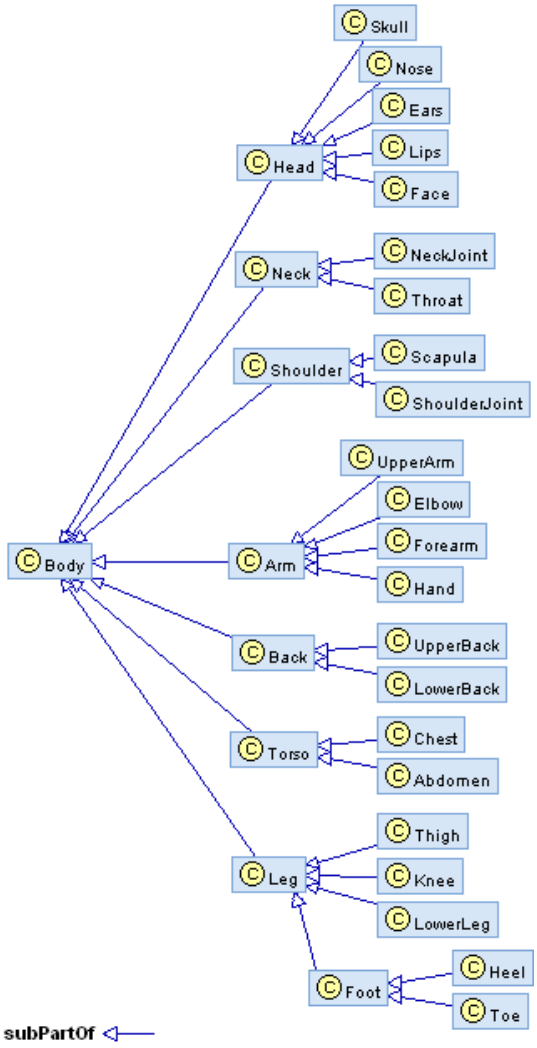

Fig. 2. Patient-centric body partonomy for localizing sports injuries

In N3 this becomes a *slotted signature* with subject `_:myDiscussion`, an `rdf:type` of `:MyDiscussion`, and the 16 arguments as the remaining slots:

```
_:myDiscussion
    rdf:type       :MyDiscussion;
    :profileID     ?ProfileID
    :injury        ?Injury;
    :minAge        ?MinAge;
    :maxAge        ?MaxAge;
    :minRSVP       ?MinRSVP;
    :maxRSVP       ?MaxRSVP;
    :category      ?Category;
    :treatment     ?Treatment;
    :healingStage  ?HealingStage;
    :startTime     ?StartTime;
    :endTime       ?EndTime;
    :duration      ?Duration;
    :channel       ?Channel;
    :contact       ?Contact;
    :gender        ?Gender;
    :timeZone      ?TimeZone.
```

The complete signatures are being maintained in a global document.[14]

While rules (including underlying facts) according to a positional signature are usually more concise, positional arguments must be specified in their fixed order (with missing or inapplicable arguments represented by 'null values'). Conversely, while a slotted signature usually makes rules more verbose, slotted arguments can be specified in any order (with missing or inapplicable arguments just becoming omitted). The Datalog special case of the positional paradigm (i.e., Hornlog without complex arguments) corresponds to the relational model in that facts with the same predicate correspond to relational tables, and rules to relational views. Conversely, the slotted paradigm is a special case of object-oriented models where objects ('subjects') are declaratively described by slots and can inherit slot values, but slot values are not procedurally updated. Depending on their previous experience with these paradigms (e.g., with relational databases or RDF metadata), whose characteristics transpire even in GUIs, virtual organizations or individual patients can take advantage of their favorite one. For a synthesis of the two paradigms see [6].

Shared rules defining PatientSupporter predicates have been collected for the rulebase of the OA. They, together with the signatures and ontologies, formalize the global knowledge of the PatientSupporter system.

An example of a POSL ('backward') rule defines `participation` in a virtual support group as follows:

```
participation(?ProfileID,?Injury,?MinRSVP,?MaxRSVP) :-
     groupSize(?ProfileID,?Injury,?Min,?Max),
     greaterThanOrEqual(?MinRSVP,?Min),
     lessThanOrEqual(?MaxRSVP,?Max).
```

The first argument of the conclusion predicate `participation` is the patient (`?ProfileID`) the rule is instantiated for, followed by a lesion (`?Injury`) argument, followed by the minimal (`?MinRSVP`) and maximal (`?MaxRSVP`) number of

[14] http://ruleml.org/PatientSupporter/Signatures/

participants the querier wants to have in a group for the lesion. The rule succeeds for its four positional arguments if `?ProfileID`'s desired group size (`groupSize`) is between `?Min` and `?Max`, `?MinRSVP` $\geq$ `?Min`, and `?MaxRSVP` $\leq$ `?Max`.

The corresponding N3 ('forward') rule for deriving `_:participation` facts is as follows, where the `?rsvpQuery` premise does not correspond to a premise of the POSL rule but is needed to bind the 'input' arguments of its conclusion:

```
{
 ?rsvpQuery
     rdf:type    :RSVPQuery;
     :profileID  ?ProfileID;
     :injury     ?Injury;
     :minRSVP    ?MinRSVP;
     :maxRSVP    ?MaxRSVP.

 ?groupSize
     rdf:type    :GroupSize;
     :profileID  ?ProfileID;
     :injury     ?Injury;
     :min        ?Min;
     :max        ?Max.

 ?MinRSVP math:notLessThan ?Min.

 ?MaxRSVP math:notGreaterThan ?Max.
}
=>
{
 _:participation
     rdf:type    :Participation;
     :profileID  ?ProfileID;
     :injury     ?Injury;
     :minRSVP    ?MinRSVP;
     :maxRSVP    ?MaxRSVP.
}.
```

The global OA knowledge base is being maintained in both language paradigms,[13] i.e. POSL[15] and N3[16].

4 Locally Distributed Knowledge Bases for Patients

Locally distributed knowledge bases are grouped as profiles underneath the PAs. Each PA group has its own kind of knowledge base, according to the medical subarea associated with the body partonomy (cf. Section 3). For example, the profiles created by patients with `Leg` injuries are kept with the `Leg` PA.

The local knowledge bases have information about patients to model their profiles using the following vocabulary of properties: A unique identifier of a profile, `ProfileID`; the kind of injury of the patient, `Injury`; the age of the patient, `Age`; the time zone of the the patient, `TimeZone`; the treatment required, `Treatment`; the stage of healing of the injury, `HealingStage`; and the category information, `Category`. The properties `Treatment`, `HealingStage`, and `Category` have the following allowed value ranges: The `Treatment` property currently has one of the values `Bandage`, `MajorOperation`, `MediumOperation`,

[15] http://ruleml.org/PatientSupporter/resources/OA/PS-Global.posl
[16] http://ruleml.org/PatientSupporter/resources/OA/PS-Global.n3

MinorOperation, MajorMedication, MediumMedication, MinorMedication, or ChangeOfLifeStyle; the HealingStage property has values Fresh, Medium, Convalescent, or Healed; and the Category property has values In or Out patient.

For example, this is a local myDiscussion fact about p0001 according to the positional signature of Section 3 (in POSL we use ?:Leg as an anonymous variable of type Leg, assuming p0001 has one leg injury):

```
myDiscussion(p0001,?:Leg,20:integer,50:integer,5:integer,10:integer,Out,Bandage,
             Medium,
             dateTime[2011:integer,6:integer,1:integer,10:integer,15:integer],
             dateTime[2011:integer,6:integer,1:integer,11:integer,20:integer],
             dateTime[0:integer,0:integer,0:integer,0:integer,30:integer],
             Skype,John27,Male,-400).
```

Similarly, given below is its counterpart according to the slotted signature (in N3 we use :Leg as a constant, again standing for one leg injury):

```
:myDiscussion_1
    rdf:type       :MyDiscussion;
    :profileID     :p0001;
    :injury        :Leg;
    :minAge        :20;
    :maxAge        :50;
    :minRSVP       :5;
    :maxRSVP       :10;
    :category      :Out;
    :treatment     :Bandage;
    :healingStage  :Medium;
    :startTime     [:year 2011; :month 6; :day 1; :hour 10; :minute 15];
    :endTime       [:year 2011; :month 6; :day 1; :hour 11; :minute 20];
    :duration      [:year 0; :month 0; :day 0; :hour 0; :minute 30];
    :channel       :skype,
    :contact       :John27,
    :gender        :Male,
    :timeZone      :-400.
```

Both express interest in a myDiscussion about Leg injuries, with Medium stage of healing, Bandage level for treatment, and with category of Out patient. It is proposed for June 1st, 2011, between 10:15 AM and 11:20 AM (GMT -4:00 Atlantic Time) for a duration of 30 minutes. It should have the form of a Skype call for 5 to 10 people. The Skype user name of the person advertising this time is John27.

This fact (in POSL and N3) can be generated by a local rule (again, in both paradigms) which uses another rule and facts to satisfy its premises. Given below is an example of a positional POSL rule from the PA knowledge base of patient p0001, defining the main predicate myDiscussion about Leg injuries, specifying his desired support-group discussion:

```
myDiscussion(p0001,?:Leg,?MinAge,?MaxAge,?MinRSVP,?MaxRSVP,?Category,?Treatment,?HealingStage,
                    dateTime[?StartYear,?StartMonth,?StartDay,?StartHour,?StartMinute],
                    dateTime[?EndYear,?EndMonth,?EndDay,?EndHour,?EndMinute],
                    dateTime[?DurYear,?DurMonth,?DurDay,?DurHour,?DurMinute],
                                                  ?Channel,?Contact,?Gender,?TimeZone) :-
  ageCheck(p0001,?MinAge,?MaxAge,?Age),
  participation(p0001,?:Leg,?MinRSVP,?MaxRSVP),
  communication(p0001,?Channel,?Contact),
  notEqual(?Channel,MSN),
```

```
  event(p0001,?:Leg,Possible,
    dateTime[?StartYear,?StartMonth,?StartDay,?StartHour,?StartMinute],
    dateTime[?EndYear,?EndMonth,?EndDay,?EndHour,?EndMinute]),
  duration(p0001,?:Leg,dateTime[?DurYear,?DurMonth,?DurDay,?DurHour,?DurMinute]),
  goodDuration(p0001,?:Leg,
    dateTime[?DurYear,?DurMonth,?DurDay,?DurHour,?DurMinute],
    dateTime[?StartYear,?StartMonth,?StartDay,?StartHour,?StartMinute],
    dateTime[?EndYear,?EndMonth,?EndDay,?EndHour,?EndMinute]),

  category(p0001,?:Leg,?Category),
  treatment(p0001,?:Leg,?Treatment),
  healingStage(p0001,?:Leg,?HealingStage),

  gender(p0001,?Gender),
  timeZone(p0001,?TimeZone).
```

The rule conclusion's `myDiscussion` predicate starts with the person's profile ID, `p0001`, followed by the kind of injury, `?:Leg`, the age limits `?MinAge` and `?MaxAge`, the group limits `?MinRSVP` and `?MaxRSVP`, the patient `?Category` (`In`/`Out`), the `?Treatment` and `?HealingStage` to be discussed, the start time, end time, and duration of the discussion, its `communication ?Channel`, as well as `p0001`'s `?Contact` name, `?Gender`, and `?TimeZone`.

The initial rule premises perform an `ageCheck`, test the `participation` constraints (cf. rule in Section 3), and filter on the `communication ?Channel`. The next premises query `p0001`'s `Possible ?:Leg`-injury `event`s and planned `duration` for this discussion, making sure it is a `goodDuration` within the event interval. The penultimate premises compute the `category`, `treatment` and `healingStage` constraints. The final premises concern the `gender` and `timeZone`.

The corresponding slotted N3 rule is given, abridged, below:

```
{
 ...

 ?event
  rdf:type       :Event;
  :profileID     :p0001;
  :injury        :Leg;
  :tense         :Possible;
  :startDateTime [:year ?StartYear; :month ?StartMonth; :day ?StartDay; :hour ?StartHour;
                                                                    :minute ?StartMinute];
  :endDateTime   [:year ?EndYear; :month ?EndMonth; :day ?EndDay; :hour ?EndHour;
                                                                      :minute ?EndMinute].

 ...
}
=>
{
 _:myDiscussion
  rdf:type       :MyDiscussion;
  :profileID     :p0001;
  :injury        :Leg;
  :minAge        ?MinAge;
  :maxAge        ?MaxAge;
  :minRSVP       ?MinRSVP;
  :maxRSVP       ?MaxRSVP;
  :category      :Out;
  :treatment     ?Treatment;
  :healingStage  ?Stage;
  :startDateTime [:year ?StartYear; :month ?StartMonth; :day ?StartDay; :hour ?StartHour;
                                                                    :minute ?StartMinute];
  :endDateTime   [:year ?EndYear; :month ?EndMonth; :day ?EndDay; :hour ?EndHour;
                                                                      :minute ?EndMinute];
```

```
  :duration      [:year ?DurYear; :month ?DurMonth; :day ?DurDay; :hour ?DurHour;
                                                                    :minute ?DurMinute];
  :channel       ?Channel;
  :contact       ?Contact;
  :gender        ?Gender;
  :timeZone      ?TimeZone.
}.
```

The POSL and N3 rules (and facts) for this and other fictitious patients are available, as templates, online.[17]

5 Interoperation between POSL and N3 Rules via RuleML/XML

The PatientSupporter use case includes a testbed for the interoperation (i.e., alignment and translation) of information in knowledge bases in the two main rule paradigms: Prolog-style (positional) relations and N3-style (slotted) frames. PatientSupporter inherits the interoperation mechanisms from Rule Responder. The interoperation methodology makes iterative use of alignment and translation: An initial alignment permits the translation of parts of a hybrid knowledge base. This then leads to more precise alignments, which in turn lead to better translations. Using this methodology, PatientSupporter can maintain relational (Pure Prolog) as well as frame (N3) versions of rules, both accessing the same, independently maintained, body partonomy.

The PAs of PatientSupporter can thus use either of these rule paradigms, while interoperation is carried out through the intermediate rule language RuleML/XML, which has sublanguages for both of them, so that the cross-paradigm translations can use the common XML syntax of RuleML. A pair of online converters[18] is used for rulebase conversion between the human-oriented POSL syntax and its XML serialization in RuleML.

For rulebase translation, the signatures of PatientSupporter relations and frames are aligned in a shared signature document,[14] discussed in Section 3, which specifies the argument positions of relations and slot names of frames. The alignment of sample relations and frames in Sections 3 and 4 then suggests the actual translations between the two rule paradigms.

Translations that are considered to be 'static' or 'at compile-time' take an entire rulebase as input and return its entire transformed version in RuleML/XML. Thus, an assumption of 'closed-arguments' of fixed signatures for relations and frames is made [8].

Positional-slotted translators for a version of RuleML are available online as an XSLT implementation.[19]

For example, POSL's `myDiscussion` relational fact of Section 4 is serialized in positional RuleML as follows, where `Ind`ividual constants are distinguished from `Data` literals:

[17] http://ruleml.org/PatientSupporter/resources/PA/
[18] http://ruleml.org/posl/converter.jnlp
[19] http://ruleml.org/ooruleml-xslt/oo2prml.html

```
<Atom>
  <Rel>myDiscussion</Rel>
  <Ind>p0001</Ind>
  <Var type="Leg"/>
  . . .
  <Data>Out</Data>
  <Ind>Bandage</Ind>
  <Data>Medium</Data>
  . . .
</Atom>
```

Extending the mappings in OO RuleML[20], N3's `myDiscussion` frame fact of Section 4 is serialized in slotted RuleML as follows, where RuleML's `Rel` represents N3's `rdf:type`:

```
<Atom>
  <oid><Ind iri=":myDiscussion_1"/></oid>
  <Rel iri=":MyDiscussion"/>
  <slot>
    <Ind iri=":profileID"/>
    <Ind>p0001</Ind>
  </slot>
  <slot>
    <Ind iri=":injury"/>
    <Ind>Leg</Ind>
  </slot>
  . . .
  <slot>
    <Ind iri=":category"/>
    <Data>Out</Data>
  </slot>
  <slot>
    <Ind iri=":treatment"/>
    <Ind>Bandage</Ind>
  </slot>
  <slot>
    <Ind iri=":healingStage"/>
    <Data>Medium</Data>
  </slot>
  . . .
</Atom>
```

While slotted-to-positional translation of atoms essentially fixes the argument order and omits the slot names, positional-to-slotted translation looks up the slot names in the shared signature document.[14] For the translation of a rule, the above translation of atoms is applied to the atom in the conclusion and to all the atoms in the premises. For a rulebase, the translation then applies to all of its rules. With the above-discussed human-oriented syntax translators, rulebases containing rules like the `myDiscussion` rule in Section 4 can thus be translated from Pure Prolog to POSL to RuleML (positional to slotted) and to N3, as well as vice versa. These translators permit rule, query, and answer interoperation, via RuleML/XML, for the Rule Responder infrastructure inherited by PatientSupporter.

The translators have been complemented by mappings between the Dlex subset of RuleML and of RIF [4].

[20] http://ruleml.org/indoo/n3ruleml.html

6 Distributed Rule Responder Querying

PatientSupporter inherits the distributed query mechanism from Rule Responder. For querying different rule engines, transformations between queries and answers from N3 and Pure Prolog through RuleML/XML are done as described for rules in Section 5. Both the global knowledge base, described in Section 3, and locally distributed knowledge bases, described in Section 4, are used in query answering.

Given below is an example of a POSL query for patient profiles, which is executed by Rule Responder's OO jDREW TD (Top-Down) engine:

```
myDiscussion(?ProfileID,?Injury:Leg,20:integer,50:integer,5:integer,10:integer,...)
```

It uses the rule from Section 4 to find any patient `(?ProfileID)` who has a `Leg` injury, age between `20:integer` and `50:integer`, and is interested in joining a discussion group with minimum `5:integer` to maximum `10:integer` people, where all the remaining arguments, indicated by '`...`', are left open as free variables.

This is the corresponding N3 query, to be executed by Rule Responder's EulerSharp EYE bottom-up engine:

```
@prefix : <patient_profiles#>.
@prefix rdf: <http://www.w3.org/1999/02/22-
            rdf-syntax-ns#>.

_:myDiscussion
  rdf:type        :MyDiscussion;
  :profileID      ?ProfileID;
  :injury         :Leg;
  :minAge         :20;
  :maxAge         :50;
  :minRSVP        :5;
  :maxRSVP        :10;
  ... .
```

After having declared two prefixes, it builds an existential ('_') node, `_:myDiscussion`, using slots for the fixed parameters and the fact-provided `?MinRSVP` (5) and `?MaxRSVP` (10) bindings to fill the variable slots again indicated by '`...`'.

Within our online test environment, the above sample query produces twelve solutions. These can be narrowed down to produce four solutions by descending the partonomy from `?Injury:Leg` to `?Injury:Foot`, and to two solutions when `?Injury:Foot` becomes `?Injury:Toe`. Using variations of such queries, patients-as-enquiry-users of PatientSupporter will be able to explore profiles of patients-as-profile-users to find or initiate a support group.

In our experiments, the overall processing times for the online-selectable[10] positional `myDiscussion` Example Queries 1-4 in Rule Responder instantiated to PatientSupporter on average were, respectively, 11s (for 12 answers), 7s (for 5 answers), 5s (for 2 answers), and 4s (for 1 answer), measured for Java JRE6 in Windows XP on an Intel Core 2 Duo 2.80GHz processor.

The implemented Rule Responder instantiation for PatientSupporter, with all source files and test queries, is available online.[10]

7 Conclusions and Future Work

PatientSupporter demonstrates the Social Semantic Subweb for patients who want to collaborate and share information with each other about sports injuries, on a global scale. It enables precise networking between patients by using ontologies combined with rules to specify and query patient profiles. Key features of PatientSupporter are: First, it permits interoperation of patient profiles between Pure Prolog (Naf Hornlog) and N3 through RuleML/XML. Second, it enables scalability of distributed knowledge on the Social Semantic Subweb via its PA modularization, starting with derivation rules and light-weight ontologies. Third, PatientSupporter uses the OO jDREW, Euler, and Prova engines, while its open Rule Responder architecture makes it easy to bring in new engines. Fourth, it makes use of a body partonomy for modeling sports injuries in a hierarchical manner (from the patients' commonsense point-of-view). Fifth, it makes use of ontologies and rules to precisely search for patient profiles, and allows enquiry users to narrow down their search in a step-wise fashion. Hence, by delivering the relevant profiles, PatientSupporter saves enquiry users from the hassle of browsing through a large set of patient profiles.

While the querying of patient rulebases by enquiry users is pretty well supported with a menu-based GUI[10], the editing of patient rulebases by profile users should be similarly supported. Especially for newcomers, the choice between positional, slotted, and combined language paradigms could be abstracted away as far as possible: A new profile user would visually select the features relevant for their profile and the underlying system would generate the profile in the language most appropriate for this selection. In future work, controlled or natural language interfaces could be developed for both querying and editing, following the ACE query interface of SymposiumPlanner-2011 [22], and information extraction methods could be explored as an alternative to 'from-scratch' profile editing.

In a separate effort, PatientSupporter's current vocabulary of properties could be refined – and rules over them could be written – to express multiple injuries to the same body part, injuries of multiple body parts, indirect symptoms, lab results, as well as specifics about diagnoses and therapies. Besides the treatment of injuries, their prevention could be represented.

An extension of PatientSupporter could include – along with the patients and their Social Semantic profiles – medical professionals and Personal Health Records (PHRs) [17]. This would assist in the formation of virtual support groups consisting of doctors and nurses as well as patients, based on the preferences and constraints of all three subgroups. For this, patients' commonsense knowledge and profiles should be mapped to medical expert knowledge and PHRs – and (partially) vice versa. For example, PatientSupporter's body partonomy for localizing global knowledge about sports injuries could (be mapped to a full-blown medical ontology such as SNOMED and) act as an index into medical knowledge about anatomy, physiology, etc. as it pertains to sports injuries. A medical professional could then provide injury-specific knowledge that is not patient-specific to the entire support group, rather than repeating it for each patient.

PatientSupporter and its use cases will thus provide new challenges and suggestions for improvements of RuleML, Rule Responder, and the involved engines. Since PatientSupporter rules are interoperated through RuleML/XML, they can also be ported from Rule Responder to other Social Semantic Web frameworks such as EMERALD [16] or be read into another Java-based system via JAXB.

8 Acknowledgements

The authors would like to thank colleagues at NRC-IIT Fredericton, especially Benjamin Craig and Julie Maitland, for helpful discussions. We also thank Zainab Almugbel, Usman Ali Chaudhry, and Sujan Saha, as well as the organizers, reviewers, and participants of CSWS2011 for valuable feedback. NSERC is thanked for its support through a Discovery Grant for Harold Boley. The work by Omair Shafiq, Derek Smith, and Taylor Osmun was done during their stays at NRC-IIT.

References

1. M. Ball, H. Boley, D. Hirtle, J. Mei, and B. Spencer. The OO jDREW Reference Implementation of RuleML. In A. Adi, S. Stoutenburg, and S. Tabet, editors, *Rules and Rule Markup Languages for the Semantic Web, First International Conference (RuleML 2005), Galway, Ireland, November 10-12, 2005, Proceedings*, volume 3791 of *Lecture Notes in Computer Science*, pages 218–223. Springer, 2005.
2. T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, and J. Hendler. N3Logic: A Logical Framework For the World Wide Web. *Theory and Practice of Logic Programming (TPLP)*, 8(3), May 2008.
3. H. Boley. Are Your Rules Online? Four Web Rule Essentials. In A. Paschke and Y. Biletskiy, editors, *Proc. Advances in Rule Interchange and Applications, International Symposium (RuleML-2007), Orlando, Florida*, volume 4824 of *LNCS*, pages 7–24. Springer, 2007.
4. H. Boley. RIF RuleML Rosetta Ring: Round-Tripping the Dlex Subset of Datalog RuleML and RIF-Core. In G. Governatori, J. Hall, and A. Paschke, editors, *RuleML*, volume 5858 of *Lecture Notes in Computer Science*, pages 29–42. Springer, 2009.
5. H. Boley. Integrating Positional and Slotted Knowledge on the Semantic Web. *Journal of Emerging Technologies in Web Intelligence*, 4(2):343–353, Nov. 2010.
6. H. Boley. A RIF-Style Semantics for RuleML-Integrated Positional-Slotted, Object-Applicative Rules. In N. Bassiliades, G. Governatori, and A. Paschke, editors, *RuleML Europe*, volume 6826 of *Lecture Notes in Computer Science*, pages 194–211. Springer, 2011.
7. H. Boley and M. Kifer. A Guide to the Basic Logic Dialect for Rule Interchange on the Web. *IEEE Transactions on Knowledge and Data Engineering*, Forthcoming 2010.
8. H. Boley, T. M. Osmun, and B. L. Craig. Social Semantic Rule Sharing and Querying in Wellness Communities. In *4th Asian Semantic Web Conference, ASWC 2009, Shanghai, China*, 2009.

9. H. Boley and A. Paschke. Rule Responder Agents: Framework and Instantiations. In A. Elci, M. T. Kone, and M. A. Orgun, editors, *Semantic Agent Systems: Foundations and Applications*. Springer Studies in Computational Intelligence, Vol. 344, 2011.
10. H. Boley, A. Paschke, and O. Shafiq. RuleML 1.0: The Overarching Specification of Web Rules. In *Proc. 4th International Web Rule Symposium: Research Based and Industry Focused (RuleML-2010), Washington, DC, USA, October 2010*, Lecture Notes in Computer Science. Springer, 2010.
11. D. Chappell. Enterprise Service Bus. OReilly: June 2004, ISBN 0-596-00675-6.
12. B. L. Craig and H. Boley. Personal Agents in the Rule Responder Architecture. In N. Bassiliades, G. Governatori, and A. Paschke, editors, *RuleML*, volume 5321 of *Lecture Notes in Computer Science*, pages 150–165. Springer, 2008.
13. G. Eysenbach. Medicine 2.0: Social Networking, Collaboration, Participation, Apomediation, and Openness. Jul-Sep 2008.
14. B. Hughes, I. Joshi, and J. Wareham. Health 2.0 and Medicine 2.0: Tensions and Controversies in the Field. 2008.
15. A. Kozlenkov. Prova Rule Language Version 3.0 User's Guide. http://www.prova.ws/etc/Prova 3.0 User Guide.pdf, May 2010.
16. K. Kravari, T. M. Osmun, H. Boley, and N. Bassiliades. Cross-Community Interoperation between the EMERALD and Rule Responder Multi-Agent Systems. In N. Bassiliades, G. Governatori, and A. Paschke, editors, *RuleML Europe*, volume 6826 of *Lecture Notes in Computer Science*, pages 44–51. Springer, 2011.
17. L. S. Liu, P. C. Shih, and G. R. Hayes. Barriers to the Adoption and Use of Personal Health Record Systems. In *Proc. iConference, Seattle, WA, USA*, pages 363–370. ACM, February 2011.
18. J. L. V. Mejino, A. V. Agoncillo, K. L. Rickard, and C. Rosse. Representing Complexity in Part-Whole Relationships within the Foundational Model of Anatomy. http://sig.biostr.washington.edu/projects/fm/FME/.
19. A. Paschke, H. Boley, A. Kozlenkov, and B. Craig. Rule Responder: RuleML-Based Agents for Distributed Collaboration on the Pragmatic Web. In *2nd ACM Pragmatic Web Conference*. ACM, 2007.
20. L. Sterling and E. Y. Shapiro. *The Art of Prolog - Advanced Programming Techniques, 2nd Ed.* MIT Press, 1994.
21. M. Walden, H. Mills, and T. Dekkers. Sports Injury Clinic. http://www.sportsinjuryclinic.net/cybertherapist/injurylist.htm.
22. Z. Zhao, A. Paschke, C. U. Ali, and H. Boley. Principles of the SymposiumPlanner Instantiations of Rule Responder. In F. Olken, M. Palmirani, and D. Sottara, editors, *Rule-Based Modeling and Computing on the Semantic Web*. LNCS 7018, Springer, 2011.