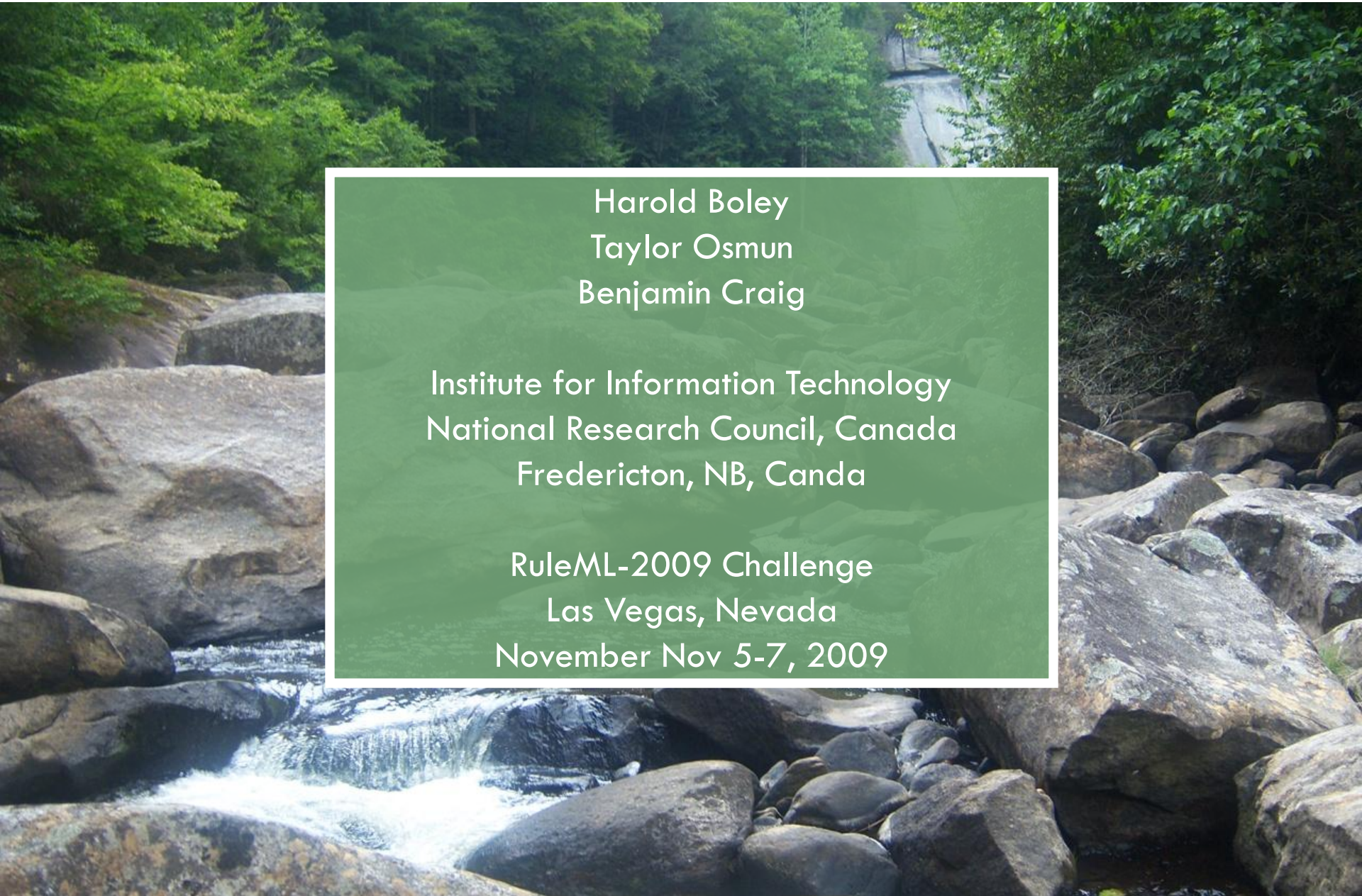
Wellness-Rules: A Web 3.0 Case Study in RuleML-Based Prolog-N3 Profile Interoperation



Harold Boley
Taylor Osmun
Benjamin Craig

Institute for Information Technology
National Research Council, Canada
Fredericton, NB, Canda

RuleML-2009 Challenge
Las Vegas, Nevada
November Nov 5-7, 2009

Outline

- WellnessRules Overview
- Profile Interoperation (POSL ⇔ N3)
- Relational (POSL) and Graph (N3) Language & Interoperation Overviews
- Global and Local Knowledge Bases
- POSL ⇔ N3 Transformation
- Taxonomy

WellnessRules Overview

- WellnessRules supports an online-interactive wellness community.
  This rule-supported community has the ability to:
  - **Create profiles** about themselves containing their preferences for activities and nutrition, their event days, and their fitness levels
  - **Compare and collaborate with others** in the community to track progress and schedule group wellness events
- Rules about wellness opportunities are
  - created by participants in rule languages such as Prolog and N3
  - interoperated within a wellness community using RuleML/XML

Profile Interoperation (POSL ⇔ N3)

- Support for both ***logic-relational*** (e.g., POSL) and ***graph-oriented*** (e.g., N3) knowledge representations
- Users may write their profile in either language
- Support for **OO jDREW** and **Euler** engines to execute queries issued to **POSL** and **N3** knowledge bases, respectively
- *Previously seen in the demo:*
  By using a RuleML subset as interchange language and Rule Responder as interchange platform, queries are applied to all supported engines, with answers returned in RuleML

POSL

- POSL integrates **po**sitional and **sl**otted knowledge for humans (e.g.: Prolog's positional and F-logic's slotted knowledge)
- WellnessRules uses **positional** POSL for ***logic-relational*** knowledge, displayed in a Prolog-like syntax

- Positional Notations:
  - Relation names:
    - Each fact or rule has a relation name

```
season(?StartTime,summer).
```

  - Values:
    - Values can be upper or lower case, separated by a comma (,)

```
season(?StartTime,summer).
```

  - Variables:
    - Can be named (prefix "**?**"), or anonymous (stand-alone "**?**")

```
season(?StartTime,?).
```

Notation 3 – N3

- A language which is a compact and readable alternative to **RDF**'s XML syntax. Uses RDF triples (**subject, property, object**) to represent knowledge
- WellnessRules uses **N3** for ***graph-oriented*** knowledge

- Slotted Notations:
  - Subject names:
    - Each fact or rule has a subject name

':' here denotes a local knowledge base

```
:season_1
   rdf:type     :Season;
   :startTime   ?StartTime;
   :period      :summer.
```

  - Variables:
    - Can be named (prefix "**?**"), or anonymous (stand-alone "**?**")

```
:season_1
   rdf:type     :Season;
   :startTime   ?StartTime;
   :period      ?.
```

  - Values (property-object pairs):
    - Each value must have a **property** (slot name):

```
:season_1
   rdf:type     :Season;
   :startTime   ?StartTime;
   :period      :summer.
```

    - Each value must also have an **object** (slot value):

```
:season_1
   rdf:type     :Season;
   :startTime   ?StartTime;
   :period      :summer.
```

Global Knowledge Base

- Contains knowledge that is relevant to everyone in the WellnessRules community

- Knowledge Areas:
  - **Season**
    - Defines timeframe of the seasons
  - **Forecast**
    - Describes the weather forecast within timeframes
  - **Meetup**
    - Contains activity meet up locations for maps

Local Knowledge Base

- Contains local knowledge that is specific to each participant in the WellnessRules community
- Knowledge Areas:
  - **Calendar**
    - Used for event planning. Allows for sharing of calendars between profiles
  - **Map**
    - Links to Meetup locations. Allows for sharing of maps between profiles
  - **Fitness**
    - Defines expected fitness level for specific a period of time (scale of 1-10)
  - **Event**
    - Possible/Planned/Performing/Past
  - **MyActivity**
    - Derive participant's individual activity preferences

Example MyActivity Rule - POSL

```
myActivity(p0001,Running,out,?MinRSVP,?MaxRSVP,?StartTime,?EndTime,?Place,?Duration,?Level)
:-
  calendar(p0001,?Calendar),
  event(?Calendar,?:Running,possible,?StartTime,?EndTime),
  participation(p0001,run,out,?MinRSVP,?MaxRSVP),
  season(?StartTime,summer),
  forecast(?StartTime,sky,?Weather),
  notEqual(?Weather,raining),
  map(p0001,?Map),
  meetup(?Map,run,out,?Place),
  level(p0001,run,out,?Place,?Duration,?Level),
  fitness(p0001,?StartTime,?ExpectedFitness),
  greaterThanOrEqual(?ExpectedFitness,?Level),
  goodDuration(?Duration,?StartTime,?EndTime).
```

- Based on this rule the following are p0001's preferences for Running outdoors:
  - The number of participants must be within the minimum and maximum
  - The season must be summer
  - It must not be raining outside
  - p0001's fitness level is greater than or equal to the required fitness level

POSL ⇔ N3 Transformation-1

- N3 requires the use of subjects for naming relationships.
  The **subject name** uses the relation name followed by "_#" where '#' is the **iteration number**
- Each corresponding N3 rule's '**relation name**' is defined via **rdf:type** and the uppercase version of the name

POSL `season(?StartTime,?Season).` ⇔

```
:season_1
   rdf:type      :Season;
   :startTime    ?StartTime;
   :period       ?Season.
```

N3

- In positional POSL slot names are not needed. Therefore, **slot names** (properties) must be created for N3, while the **slot variables** (variable objects) use the **same variable names** as POSL

POSL `season(?StartTime,?Season).` ⇔

```
:season_1
   rdf:type      :Season;
   :startTime    ?StartTime;
   :period       ?Season.
```

N3

POSL ⇔ N3 Transformation-2

- **Rules** are represented and handled differently.
  OO jDREW (using POSL) is essentially
  a **top-down** (:-) reasoner.
  Euler (using N3) is a **bottom-up** reasoner (=>):
- Assumes previous slide
  (**Transformation-1**)

POSL

```
myActivity(p0001,Running,in,?MinRSVP,?MaxRSVP,
           ?StartTime,?EndTime,?Place,?Duration,?Level)
 :-
   ...
   forecast(?StartTime,sky,?Weather),
   notEqual(?Weather,raining),
    ...
.
```

⇔

N3

```
{
...
?forecast
      rdf:type              :Forecast;
      :startTime            ?StartTime;
      :aspect               :sky;
      :value                ?Weather.

 ?Weather log:notEqualTo :raining.
...
}
=>
{
 _:myActivity
      rdf:type              :MyActivity;
      :profileID            :p0001;
      :activity             :Running;
      :inOut                :in;
      :minRSVP              ?MinRSVP;
      :maxRSVP              ?MaxRSVP;
      :startTime            ?StartTime;
      :endTime              ?EndTime;
      :location             ?Place;
      :duration             ?Duration;
      :fitnessLevel         ?FitnessLevel.
}.
```

POSL ⇔ N3 Transformation-3

- The POSL handling of **negation as failure** (NAF) is via a built-in:

```
naf( event(?Calendar, ?:Running, past, ?StartTimeYDay, ?EndTimeYDay)),
```

POSL

N3 does not have a built-in to handle NAF. Therefore, NAF is encoded by a **e:findall** searching for an **empty list**

```
?NAF e:findall
    (?event
        {?event
            rdf:type          :Event;
            :calendarID     ?CalendarID;
            :aspect           :Running;
            :tense            :past;
            :startTime      ?StartTimeYDay;
            :endTime        ?EndTimeYDay.}
                                        () ).
```

N3

- **POSL** has **built-in** math operations. **N3** uses **package-prefixed** math operations.

POSL

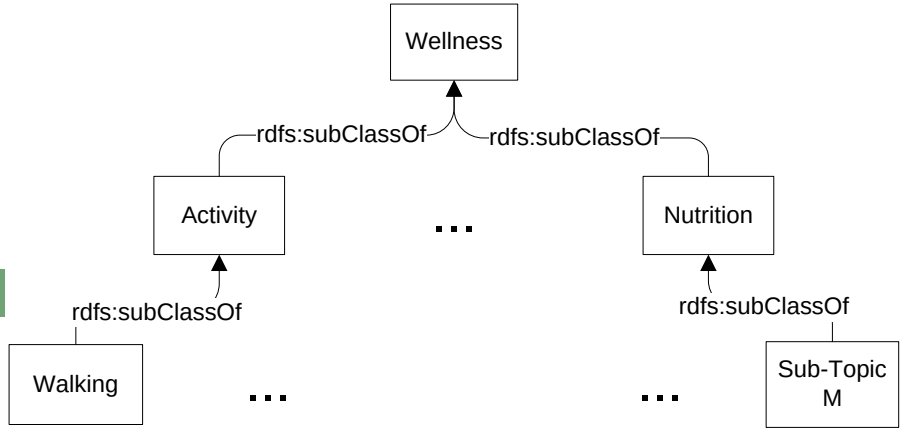
```
greaterThanOrEqual(?ExpectedFitness,?Level),
```

⇔

```
?ExpectedFitness math:notLessThan ?FitnessLevel.
```

N3

Taxonomy



- The WellnessRules taxonomy is broken into two topics: Activity and Nutrition
- Each of these contains multiple sub-topics (e.g. Walking or Running)
- Both representations use **rdf:type, rdfs:Class** and **rdfs:subClassOf**
- Taxonomy classes function as user-defined types to restrict rule variables

RDF (used by POSL)

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

   <rdf:Description rdf:ID="Wellness">
     <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
   </rdf:Description>

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

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

    …
```

⇔

N3

```
@prefix : <wellnessRules#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.

:Wellness     rdf:type            rdfs:Class.
:Activity     rdf:type            rdfs:Class;
              rdfs:subClassOf     :Wellness.
:Walking      rdf:type            rdfs:Class;
              rdfs:subClassOf     :Activity.
…
```

Wrap Up

- The WellnessRules case study:
  - Demonstrates **profile interoperation** between both ***logic-relational*** (e.g., POSL) and ***graph-oriented*** (e.g., N3) knowledge representations
  - Provides **transformation** techniques in the context of WellnessRules **between** these knowledge representation **formats**
  - *Previously seen in the demo:*
    With an exciting use case, creates an **online-interactive wellness community** through the WellnessRules Rule Responder system