WellnessRules:
N3 Implementation Through the Euler Engine

Taylor Osmun

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

Outline

- WellnessRules Overview
- WellnessRules + Rule Responder
- PA & OA components of WellnessRules
- WellnessRules Ontology in N3
- Sample WellnessRules usage through N3 & Euler
  - Sample Query
  - Sample Result
  - MyActivity Rule

WellnessRules Outline

- WellnessRules goal is to create an online-interactive wellness community. This community would have the ability to:
  - Create profiles about themselves containing their preferences for activities and nutrition, their event days, and their fitness levels.
  - Collaborate with others in the community to schedule group wellness events.
  - Track other participant's progress and relate it to their own.
- Rules about wellness opportunities are created by participants in rule languages such as Prolog and N3, and translated within a wellness community using RuleML/XML.

WellnessRules + Rule Responder

- Rule Responder is an intelligent multi-agent infrastructure for collaborative teams and virtual communities.

- Each Rule Responder instantiation uses three different kinds of agents:
  - Organizational Agent (OA)
  - Personal Agents (PAs)
  - External Agents (EAs)

- WellnessRules uses the OA, PAs, and EAs to create an online-interactive wellness community.

OA: Global Component

- Contains all global knowledge in the WellnessRules knowledge base.
- Knowledge Areas:
  - Season
    - Defines timeframe of the seasons.
  - Forecast
    - Describes the weather forecast within timeframes.
  - Meetup
    - Contains activity meet up locations for maps.

PA: Profile Component

- Contains local knowledge which is unique 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
    - Define user's individual activity preferences

Ontology

- The WellnessRules ontology is broken into two topics, Activity, and Nutrition.
- Each of these contain multiple sub-topics (i.e. Running).
- Our N3 representation uses rdf:type and rdfs:subClassOf

```
@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.
:Running        rdf:type            rdfs:Class;
                rdfs:subClassOf     :Activity.
:Swimming       rdf:type            rdfs:Class;
                rdfs:subClassOf     :Activity.
:Skating        rdf:type            rdfs:Class;
                rdfs:subClassOf     :Activity.
:Yoga           rdf:type            rdfs:Class;
                rdfs:subClassOf     :Activity.
:Hiking         rdf:type            rdfs:Class;
                rdfs:subClassOf     :Activity.
:Baseball       rdf:type            rdfs:Class;
                rdfs:subClassOf     :Activity.
```

Sample WellnessRules Usage

- The following slides contain:
  - Sample Query
  - Sample Result
  - MyActivity Rule
- The prefix ':' represents the WellnessRules knowledge base:

```
@prefix : <wellnessRules#>.
```

- There are 3 things to look for:
  - Query Constants – *User's* preferences, 'passed in' to the rule and conclusion.
  - Variables – The variables that are 'transported' from premise to conclusion.
  - Profile Constraints – *Profile's* preferences, used in the MyActivity rule.

Sample Query

- Asks the WellnessRules system if the user ‘p0001’ is interested in going for an indoor run during the given times.
- Query Constants:
  - :MyActivity
  - :p0001
  - :Running
  - :in
  - “2009-06-15T10:15:00”
  - “2009-06-15T11:15:00”

```
### Query

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

 _:myActivity
      rdf:type              :MyActivity;
      :profileID            :p0001;
      :activity             :Running;
      :inOut                :in;
      :minRSVP              ?MinRSVP;
      :maxRSVP              ?MaxRSVP;
      :startTime            "2009-06-15T10:15:00";
      :endTime              "2009-06-15T11:15:00";
      :location             ?Place;
      :duration             ?Duration;
      :fitnessLevel         ?FitnessLevel.
```

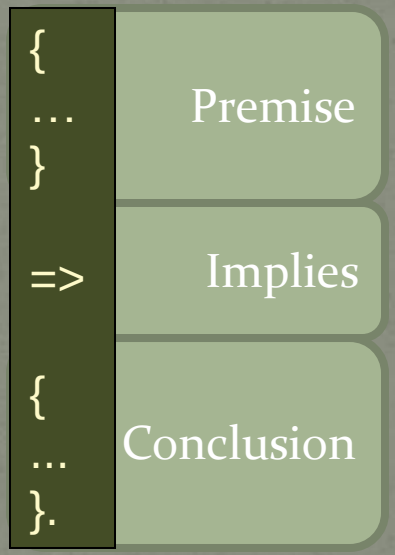
Sample Result

- p0001 is interested in running indoors within this timeframe.

- Variables:
  - 1
  - 2
  - :joesGym
  - “P10M”
  - 5

Sample MyActivity Rule – Overview

- A rule consists of a subgraph {… } of premises, an ‘implies’ arrow => and a subgraph {… } for the conclusion.
- We will develop a rule, showing its premises in three parts, followed by its conclusion.



Sample MyActivity Rule – Premise Part 1

- Using global and local facts, the season and temperature are retrieved.

- Profile Constraints:
  - Has a possible event
  - Season = Summer
  - Temperature >= 30

```
{?calendar
      rdf:type              :Calendar;
      :profileID            :pooo1;
      :calendarID           ?CalendarID.
?event
      rdf:type              :Event;
      :calendarID           ?CalendarID;
      :aspect               :Running;
      :tense                :possible;
      :startTime            ?StartTime;
      :endTime              ?EndTime.
?season
      rdf:type              :Season;
      :startTime            ?StartTime;
      :value                :summer.
?forecast
      rdf:type              :Forecast;
      :startTime            ?StartTime;
      :aspect               :temperature;
      :value                ?Temp.
?Temp math:notLessThan 30.
... }
```

Users may be using another participant's calendar

Sample MyActivity Rule – Premise Part 2

- Using global and local facts, the min/max RSVP, and location of the event is determined.

- Profile Constraints:
  - Has a possible event
  - Season = Summer
  - Temperature >= 30

```
{ ...
 ?participation
        rdf:type            :Participation;
        :profileID          :pooo1;
        :activity           :run;
        :inOut              :in;
        :min                ?MinRSVP;
        :max                ?MaxRSVP.
?map
        rdf:type            :Map;
        :profileID          :pooo1;
        :mapID              ?MapID.
 ?meetup
        rdf:type            :Meetup;
        :mapID              ?MapID;
        :activity           :run;
        :inOut              :in;
        :location           ?Place.
... }
```

Sample MyActivity Rule – Premise Part 3

- Using local facts, the level of the activity, and the user's preferred level are checked.

- Profile Constraints:
  - Has a possible event
  - Season = Summer
  - Temperature >= 30
  - Expected Fitness >= Required Fitness

```
{ ...
?level
      rdf:type                :Level;
      :profileID              :pooo1;
      :activity               :run;
      :inOut                  :in;
      :location               ?Place;
      :duration               ?Duration;
      :fitnessLevel           ?FitnessLevel.

 ?fitness
      rdf:type                :Fitness;
      :profileID              :pooo1;
      :startTime              ?StartTime;
      :expectedFitness        ?ExpectedFitness.

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

Sample MyActivity Rule –Conclusion

- The three key components, Query Constants, Variables, and Profile Constraints, along with other facts in the knowledge base, will be used to fill this premise. This will generate the previously seen result.

- There can be many answers to a single query.

Premise:

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

Result:

```
:MyActivity;
:p0001;
:Running;
:in;
1;
2;
"2009-06-15T10:15:00";
"2009-06-15T11:15:00";
:joesGym;
"P10M";
5.
```

Recap

Profile Constraints:

- Has a possible event

```
?event
    ...
    :tense      :possible;
    ...
```

- Season = Summer

```
?season
    ...
    :value      :summer.
```

- Temperature >= 30

```
?Temp math:notLessThan 30.
```

- Expected Fitness >= Required Fitness

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

Query Constants:

- :MyActivity
- :p0001
- :Running
- :in
- “2009-06-10T10:15:00”
- “2009-06-10T11:15:00”

```
...
rdf:type        :MyActivity;
:profileID      :p0001;
:activity       :Running;
:inOut          :in;
:startTime      "2009-06-15T10:15:00";
:endTime        "2009-06-15T11:15:00";
...
```

Variables:

- 1
- 2
- :joesGym
- “P10M”
- 5

```
...
:minRSVP        1;
:maxRSVP        2;
:location       :joesGym;
:duration       "P10M";
:fitnessLevel   5.
...
```

Wrap Up

- WellnessRules Overview and Rule Responder
- Local and Global components of WellnessRules
- WellnessRules Ontology in N3
- Sample WellnessRules usage through N3 & Euler
  - Query Constants
  - Variables
  - Profile Constraints
- Coming up:
  - Euler Eye Installation, Demo, and Deep Taxonomy Benchmark

Euler Eye Installation, Demo, and Deep Taxonomy Benchmark

Taylor Osmun

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

Euler Eye Benchmarking Process

- Three test cases:
  1. Linear Relationship
  2. Single Additional Option
  3. Two Additional Options

- Timed via Java JRE 1.6.0_13, using Euler Eye 5.1.3
- Timings are taken for increasing number of triples. 100, 1 000, 10 000, and 20 000.
- Final values are plotted in MATLAB and equation is estimated.

Linear Relationship - N3 Data

1. Uses a single fact:

```
...
:Test rdf:type :A1
```

2. Query is issued so that it must traverse all possible answers:

```
...
_:Subject rdf:type :A2.
```

3. Using this format for relations. Each rule counts as a triple.

```
...
{?X rdf:type :A1} => {?X rdf:type :B1}.
{?X rdf:type :B1} => {?X rdf:type :C1}.
{?X rdf:type :C1} => {?X rdf:type :D1}.
{?X rdf:type :D1} => {?X rdf:type :E1}.
...
{?X rdf:type :Y1} => {?X rdf:type :A2}.
```

Z is skipped so as to return to the beginning of the alphabet, within 24 characters.

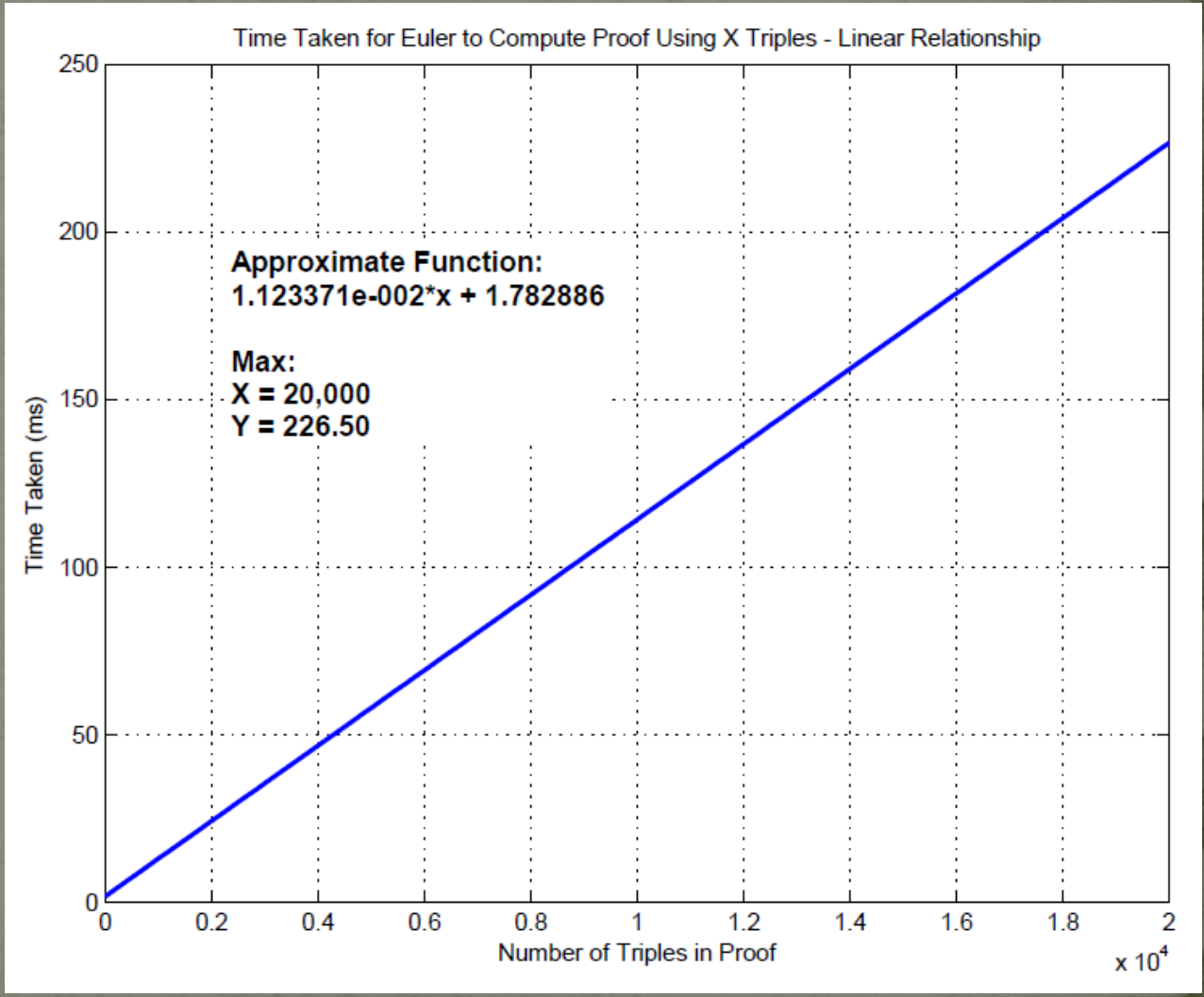
4. Produces the result:

```
...
:Test a :A2.
```

rdf:type

Linear Relationship - Results

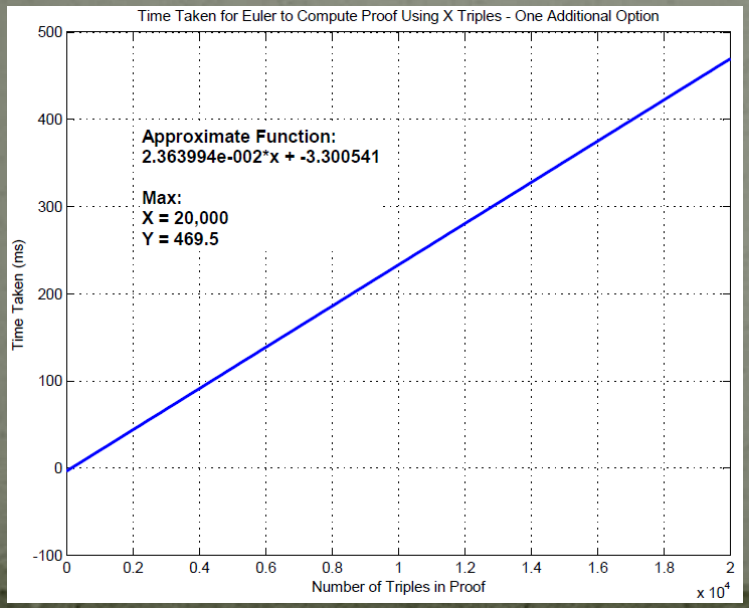
- Note the linear growth of the time taken, as more triples (linear rules) are added.



Additional Options – N3 Data & Results

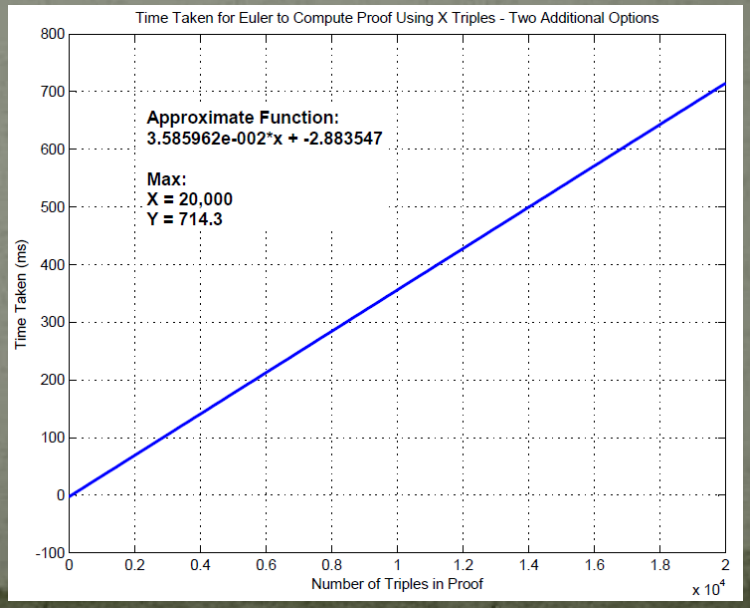
- Single additional option:

```
...
{?X rdf:type :A1} => {?X rdf:type :B1}.
{?X rdf:type :A1} => {?X rdf:type :Node1}.
{?X rdf:type :B1} => {?X rdf:type :C1}.
{?X rdf:type :B1} => {?X rdf:type :Node2}.
...
```



- Two additional options:

```
...
{?X rdf:type :A1} => {?X rdf:type :B1}.
{?X rdf:type :A1} => {?X rdf:type :Node1}.
{?X rdf:type :A1} => {?X rdf:type :Node2}.
{?X rdf:type :B1} => {?X rdf:type :C1}.
{?X rdf:type :B1} => {?X rdf:type :Node3}.
{?X rdf:type :B1} => {?X rdf:type :Node4}.
...
```

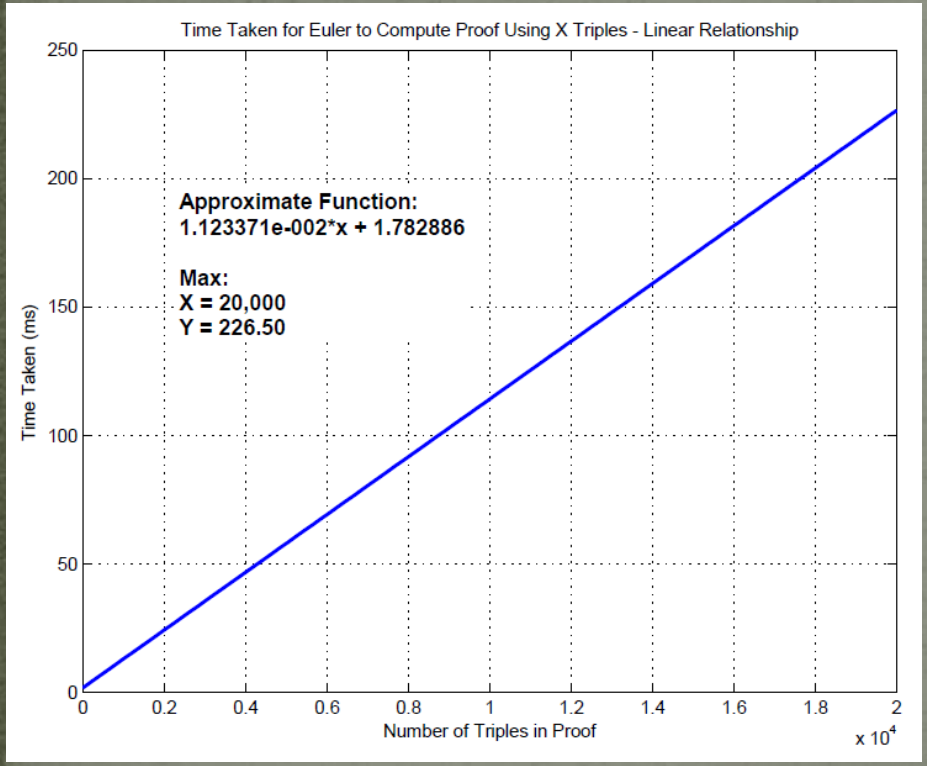


Euler Eye – All Results

- Regardless of the number of additional options, the growth of the function will still be linear.
- Change in magnitude:
  - Linear Relationship @ 20,000:
    - 2,265 sec
  - One Additional Option @ 20,000:
    - 4,695 sec
  - Two Additional Options @ 20,000:
    - 7,143 sec
- But again, a linear pattern is observed.
- Therefore, Euler EYE is extremely efficient with regards to overall time, as well as increasing complexity of the knowledge base.

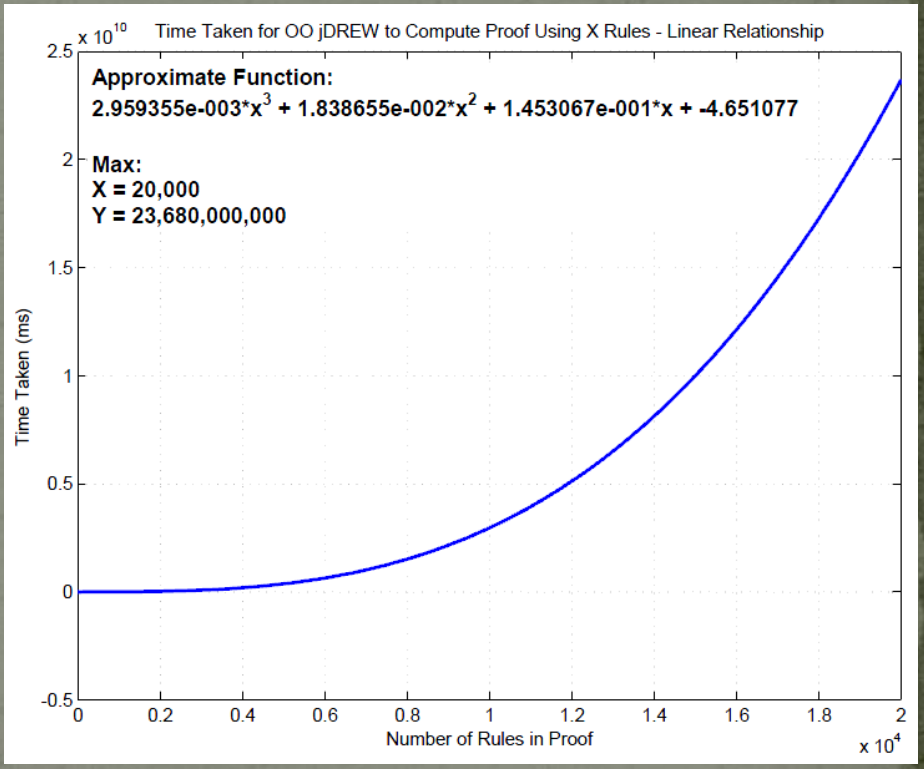
Euler Eye vs OO jDREW

- Euler Eye Results:



- Linear Relationship @ 20,000
  - 2,265 sec = 37.75 min

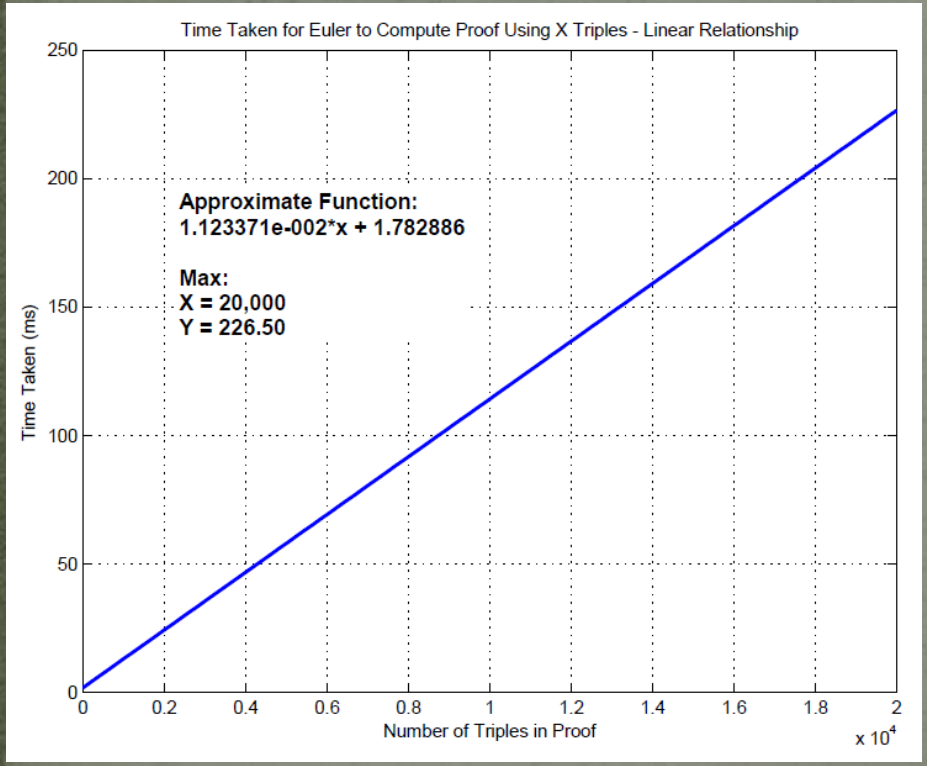
- OO jDREW Results:



- Linear Relationship @ 20,000
  - 2.3*10^14ms = 750 years

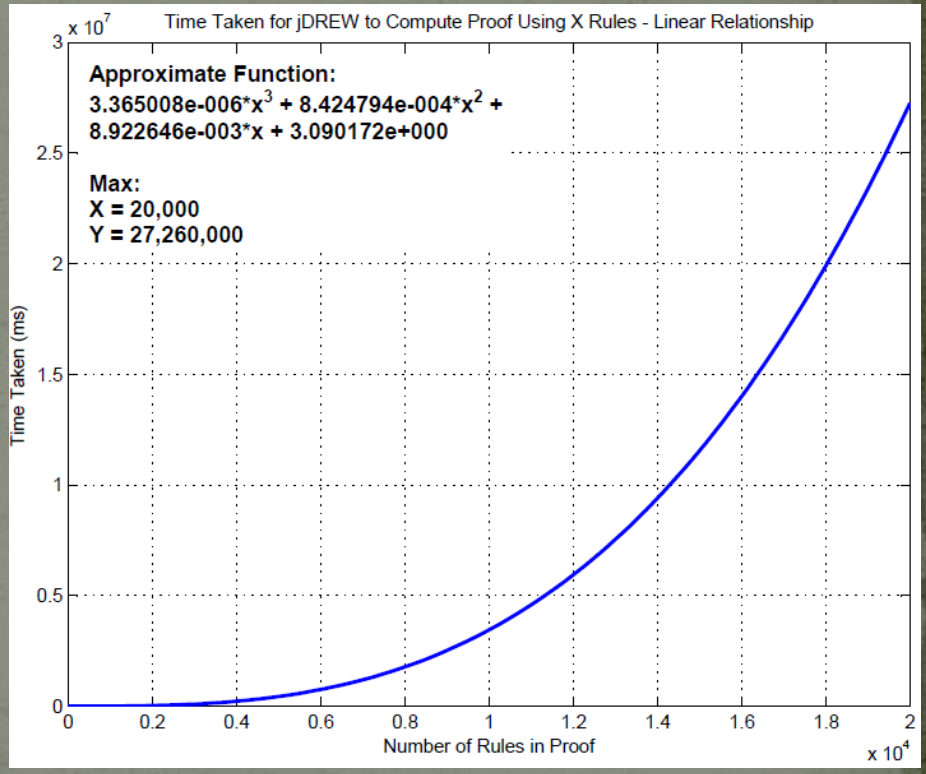
Euler EYE vs jDREW

- Euler Eye Results:



- Linear Relationship @ 20,000
  - 2,265 sec = 37.75 min

- jDREW Results:



- Linear Relationship @ 20,000
  - 2.7*10^11 = 316 days
    (OO jDREW = 750 years)

Wrap Up

- Euler EYE was set up for use in Eclipse.
- Small demo using WellnessRules was shown.
- Using the three test cases, benchmarking results were analyzed for Euler EYE
- These results were then compared to OO jDREW and jDREW, respectively.

Links

- Euler:
  - http://www.agfa.com/w3c/euler/
- Semantic Web Tutorial Using N3:
  - http://www.w3.org/2000/10/swap/doc/
- WellnessRules – Rule Responder:
  - http://ruleml.org/WellnessRules/RuleResponder/