The Grailog User Interface for Knowledge Bases of Ontologies & Rules


Harold Boley
NRC-ICT Fredericton
Faculty of Computer Science, University of New Brunswick
Canada

OMG Technical Meeting, Ontology PSIG,
Cambridge, MA, 21 June 2012

Thanks for feedback on various versions and parts of this presentation:
Grailog: Knowledge Representation with Extended Graphs for Extended Logics
SAP Enterprise Semantics Forum, 24 April 2012
Grailog: Towards a Knowledge Visualization Standard
BMIR Research Colloquium, Stanford, CA, 4 April 2012
PARC Research Talk, Palo Alto, CA, 29 March 2012
RuleML/Grailog: The Rule Metalogic Visualized with Generalized Graphs
PhiloWeb 2011, Thessaloniki, Greece, 5 October 2011
Abstract

Following the AI tradition of simple semantic networks and the Semantic Web use of RDF triple stores, a knowledge base could in principle be specified and accessed as a single directed labeled graph. However, such a graph cannot straightforwardly represent nested structures, non-binary relationships, and relation descriptions. These advanced features require encoded constructs with auxiliary nodes and relationships, which also need to be kept separate from straightforward constructs. Therefore, various extensions of directed labeled graphs have been proposed for knowledge representation, including graph partitionings (possibly interfaced as complex nodes), n-ary relationships as directed labeled hyperarcs, and (hyper)arc labels used as nodes of other (hyper)arcs. Meanwhile, a lot of AI / Semantic Web research and development on ontologies & rules has gone into extended logics for knowledge representation such as object (frame) logics, description logics, general modal logics, and higher-order logics. The talk demonstrates how knowledge representation with graphs and logics can be reconciled. It proceeds from simple to extended graphs for logics needed in AI and the Semantic Web. Along with its visual introduction, each graph construct is mapped to its corresponding symbolic logic construct. This has led to the development of the Grailog user interface for knowledge bases as part of the Web-rule industry standard RuleML (http://ruleml.org/#Grailog)
Remove Barrier to Entry for Logic: Graph Visualization of Knowledge

• From 1-dimensional *symbol-logic* knowledge specification to 2-dimensional *graph-logic visualization* in a convenient 2D syntax
  – Supports human in the loop in knowledge elicitation, validation, and processing

• Comбинable with graph transformations for efficient *implementation* of specifications & for visualizing model-theoretic *semantics*
  – Deep names, as graph nodes, mapped directly/injectively to elements of semantic interpretation
Grailog

**Graph inscribed logic** invokes imagery for logic

Proposed cognitively motivated graph standard for visual-logic knowledge:

Easy to learn and draw, read and remember, e.g. for eScience, eLearning, and eBusiness

Generalized-graph framework as one uniform user interface to major (Semantic Web) logics:

Pick & choose subset for each knowledge base, map to/fro RuleML sublanguage and UML+OCL, and access via API4KB protocol
Grailog and API4KB

• Both strive for broad coverage of main data & knowledge representation paradigms:
  – RDF (directed-labeled-graph) and Relational (Datalog-fact-like) data
  – Ontology (description-logic) and Rule (Horn- and general-logic) knowledge

• An API can be (initially) designed and tested with a human in the loop much like a GUI
Generalized Graphs for the Representation and Mapping of Logic Languages

• We have used generalized graphs for representing various logic languages, where basically:
  – Graph nodes (vertices) represent individuals, classes, etc.
  – Graph arcs (edges) represent relations

• Next slides:
  What are the principles of this representation and what graph generalizations are required?

• Later slides:
  How are these graphs mapped (invertibly) to logic, thus specifying Grailog as a ‘GUI’ for RuleML?
Graphical Elements: Names

- **Written into** boxes (nodes):
  - **Deep** (canonical, distinct) names
    - (Occurrence-)restricted
      Unique Name Assumption (rUNA)
      via Deep Name Occurrence (DNO)

- **Written onto** boxes (node labels):
  - **Shallow** (alternate, ‘aka’) names
    - (Occurrence-)restricted
      Non-unique Name Assumption (rNNA)
      via Shallow Name Occurrence (SNO)
Instances: Individual Constants with Deep Name Specification

General: Graph (node)  Logic

Examples: Graph  Logic

Warren Buffett  Warren Buffett

General Electric  General Electric

US$ 3,000,000,000  US$ 3,000,000,000
Instances: Individual Constants with Shallow Name Specification

General: Graph (node)  Logic (vertical bar marks shallowness)

Examples: Graph  Logic

WB  /WB
GE  /GE
US$ 3B  /US$ 3B
Parameters: Individual Variables

General: Graph (*hatched* node) Logic (*italics* font, POSL uses "?" prefix)

Examples: Graph

\[
\begin{array}{c}
X \\
Y \\
A
\end{array}
\]

Logic

\[
\begin{array}{c}
X \\
Y \\
A
\end{array}
\]
Predicates: Binary Relations (1)

General: Graph (labeled arc) Logic

Example: Graph Logic

Warren Buffett Trust Warren Buffett, General Electric
Predicates: Binary Relations (2)

General: Graph *(labeled arc)*

Logic

Example: Graph

Logic

```
var_1 \[binrel\] var_2
```

```
X \[Trust\] Y
```
Ground Equality: Identifying Pairs of Constants

General: Graph (unlabeled undirected arc) Logic (with equality)

Example: Graph Logic

\[
\text{inst}_1 \rightarrow \text{inst}_2 \quad \text{inst}_1 = \text{inst}_2
\]

Inspired by Charles Sanders Peirce’s line of identity, as a co-reference link
Ground Equality: Defining Symbolic Constants as IRIs

General: Graph (unlabeled undirected, colon-tailed arc)

\[ \text{inst} : \text{IRI} \]

Logic (with oriented equality, webized)

\[ \text{inst} := \text{IRI} \]

Example: Graph

GenEl

\[ \text{http://www.ge.com/} \]

Logic

\[ /\text{GenEl} := \text{http://www.ge.com/} \]

Definitional equality can also be used for the prefix part of the CURIE notation
Predicates: n-ary Relations (n>1)

General: Graph (hyperarc) Logic

Example: Graph Logic

(n=3)

Invest(WB, GE, US$ 3·10^9)
Implicit Conjunction of Formula Graphs: Co-Occurrence on Top-Level

General: Graph \((m\) hyperarcs)\n
\[
\text{\(\text{inst}_{1,1}\)} \text{\(\text{rel}_{1}\)} \text{\(\text{inst}_{1,2}\)} \ldots \text{\(\text{inst}_{1,n^1}\)}
\]

\[
\text{\(\text{inst}_{m,1}\)} \text{\(\text{rel}_{m}\)} \text{\(\text{inst}_{m,2}\)} \ldots \text{\(\text{inst}_{m,n^m}\)}
\]

Example: Graph \((2\) hyperarcs)\n
Logic

\[
\text{rel}_1(\text{\(\text{inst}_{1,1}, \text{\(\text{inst}_{1,2}, \ldots, \text{\(\text{inst}_{1,n^1})}\)} \wedge
\]

\[
\text{\ldots \text{\(\text{rel}_m(\text{\(\text{inst}_{m,1}, \text{\(\text{inst}_{m,2}, \ldots, \text{\(\text{inst}_{m,n^m})}\)} \wedge
\]

Invest(\(\text{/WB, /GE, US}\$ \text{\(3 \cdot 10^9\)}\) \wedge

Invest(\(\text{/JS, /VW, US}\$ \text{\(2 \cdot 10^4\)}\)
Explicit Conjunction of Formula Graphs: Co-Occurrence in Complex Node

General: Graph (m hyperarcs)

Logic

\[(rel_1(inst_{1,1}, inst_{1,2}, ..., inst_{1,n^1}) \land \land \land rel_m(inst_{m,1}, inst_{m,2}, ..., inst_{m,n^m}))\]

Example: Graph (2 hyperarcs)

Logic

\[(\text{Invest}(/\text{WB}, /\text{GE}, \text{US$} \cdot 10^9) \land \land \land \text{Invest}(/\text{JS}, /\text{VW}, \text{US$} \cdot 10^4))\]
Disjunction of Formula Graphs: Co-Occurrence in Disjunctive Node

General: Graph (dotted)

\[ \text{inst}_{1,1} \xrightarrow{\text{rel}_1} \text{inst}_{1,2} \xrightarrow{\ldots} \text{inst}_{1,n^1} \]

\[ \text{inst}_{m,1} \xrightarrow{\text{rel}_m} \text{inst}_{m,2} \xrightarrow{\ldots} \text{inst}_{m,n^m} \]

Logic

\[ (\text{rel}_1(\text{inst}_{1,1}, \text{inst}_{1,2}, \ldots, \text{inst}_{1,n^1}) \lor \ldots \lor \text{rel}_m(\text{inst}_{m,1}, \text{inst}_{m,2}, \ldots, \text{inst}_{m,n^m}) ) \]

Example: Graph

Logic

\[ (\text{Invest}(/\text{WB}, /\text{GE}, \text{US} \cdot 3 \cdot 10^9) \lor \text{Invest}(/\text{JS}, /\text{VW}, \text{US} \cdot 2 \cdot 10^4)) \]
From Hyperarc Crossings to Node Copies as a Normalization Sequence (1)

Hypergraph (2 hyperarcs, crossing outside nodes)

DLG (4 arcs, do not specify to whom Latin is shown or taught)
From Hyperarc Crossings to Node Copies as a Normalization Sequence (1*)

Hypergraph (2 hyperarcs, crossing inside a node)

DLG (4 arcs, do not specify to whom Latin is shown or taught)
From Hyperarc Crossings to Node Copies as a Normalization Sequence (1**) 

Hypergraph (2 hyperarcs, parallel-cutting a node) 

DLG (4 arcs, do not specify to whom Latin is shown or taught)
From Hyperarc Crossings to Node Copies as a Normalization Sequence (1***)

Hypergraph (2 hyperarcs, employing a node copy)

Logic (2 relations, employing a symbol copy)

![Diagram](image_url)

Both ‘Latin’ occurrences remain one node even when copied for easier layout: As a deep name, ‘Latin’ will remain unique
Predicates: Unary Relations (Classes, Concepts, Types)

General: Graph (class applied to instance node)

Example: Graph

Billionaire

Warren Buffett

Logic

class

hasInstance

class(inst₁)

Billionaire(Warren Buffett)
Class Hierarchies (Taxonomies): Subclass Relation

General: Graph (two nodes)

Example: Graph

```
class₁ \sqsubseteq class₂

Rich

Billionaire \sqsubseteq Rich
```

(Description) Logic
Intensional Class Constructions (Ontologies): Class Intersection

General: Graph (solid node, as for conjunction)

Example: Graph

Billionaire  Benefactor  Environmentalist
Intensional Class Constructions (Ontologies): Class Union

General: Graph (dotted node, as for disjunction)

Example: Graph

(Billionaire $\cup$ Benefactor $\cup$ Environmentalist)

(Description)

Logic

class$_1 \cup$
class$_2 \cup$
$\ldots \cup$
class$_n$

(Description)

Logic

Billionaire $\cup$
Benefactor $\cup$
Environmentalist
Class Hierarchies (Taxonomy DAGs): Top and Bottom

General: Top (special node)  (Description) Logic

T

General: Bottom (special node)  (Description) Logic

⊥

(owl:Nothing)
Intensional Class Constructions (Ontologies): Class-Property-Restricting TBox—Existential (1*)

General: Graph (normal) (Description) Logic

Example: Graph (Description) Logic

A kind of schema, where Top class is specialized to have (multi-valued) attribute/property, Substance, with at least one value typed by class Physical
Instance Assertions (Populated Ontologies): Adding ABox to Restriction TBox—Existential (1*)

General: Graph (normal)

Example: Graph

(rUNA-Description) Logic

Example:

(rUNA-Description) Logic
Intensional Class Constructions (Ontologies): Class-Property-Restricting TBox—Universal (1*)

General: Graph (normal)

Example: Graph

A kind of schema, where Top class is specialized to have (multi-valued) attribute/property, Substance, with each value typed by class Physical.
Instance Assertions (Populated Ontologies): Adding ABox to Restriction TBox—Universal (1*)

General: Graph (normal)

(rUNA-Description) Logic
\[ \forall \text{binrel.class}(\text{inst}_0) \land \text{class}(\text{inst}_1) \land \ldots \land \text{class}(\text{inst}_n) \land \text{binrel}(\text{inst}_0, \text{inst}_1) \land \ldots \land \text{binrel}(\text{inst}_0, \text{inst}_n) \]

Example: Graph

(Socrates) 
\[ \forall \text{Substance}.\text{Physical} \]
\[ \text{Physical}(\text{P1}) \land \text{Physical}(\text{P2}) \land \text{Substance}(\text{Socrates, P1}) \land \text{Substance}(\text{Socrates, P2}) \]
Object-Centered Logic: Grouping Binary Relations Around Instance

**General:** Graph \((inst_0\text{-centered})\)

\[
\begin{align*}
\text{class} & \quad \downarrow & \quad \text{inst}_0 \\
\quad & \quad \text{binrel}_1 & \quad \text{inst}_1 \\
\cdots & & \\
\quad & \quad \text{binrel}_n & \quad \text{inst}_n \\
\end{align*}
\]

**Example:** Graph \((\text{Socrates-centerend})\)

\[
\begin{align*}
\text{Philosopher} & \quad \downarrow & \quad \text{Substance}\ (\text{Socrates, P1}) \\
\text{Socrates} & & \quad \text{T1} \\
& \quad \text{Teaching} & \quad \text{Teaching}\ (\text{Socrates, T1})
\end{align*}
\]

\[
\begin{align*}
\text{(Object-Centered) Logic} & \\
\text{class}(\text{inst}_0) & \land \\
\text{binrel}_1(\text{inst}_0, \text{inst}_1) & \land \\
\cdots & \\
\text{binrel}_n(\text{inst}_0, \text{inst}_n) & \\
\end{align*}
\]
RDF-Triple (‘Subject’-Centered) Logic: Grouping Properties Around Instance

General: Graph (inst₀-centered)

(Subject-Centered) Logic

{(inst₀, rdf:type, class),
 (inst₀, property₁, inst₁),
 ...,
 (inst₀, propertyₙ, instₙ)}

Example: Graph (Socrates-centered)

(Subject-Centered) Logic

{(Socrates, rdf:type, Philosopher),
 (Socrates, Substance, P1),
 (Socrates, Teaching, T1)
Logic of Frames (‘Records’): Associating Slots with OID-Distinguished Instance

General: Graph

(PSOA-like Frame) Logic

Example: Graph

(PSOA-like Frame) Logic

instₐ #class( slot₁->inst₁;
…
slotₙ->instₙ)

inst₀ ∈ class,
slot₁ = inst₁,
…
slotₙ = instₙ

Socrates#Philosopher( Substance->P1;
Teaching->T1)
Rules: Relations Imply Relations (1)

General: Graph (ground, shorthand)

Logic

Example: Graph

Logic

\[
\begin{align*}
rel_1 (inst_{1,1}, inst_{1,2}, \ldots, inst_{1,n^1}) & \Rightarrow \\
rel_2 (inst_{2,1}, inst_{2,2}, \ldots, inst_{2,n^2}) & \Rightarrow
\end{align*}
\]
Rules: Relations Imply Relations (3)

General: Graph (inst/var terms)

Logic

$(\forall \text{var}_{i,j})$

$\text{rel}_1(\text{term}_{1,1}, \text{term}_{1,2}, \ldots, \text{term}_{1,n^1}) \Rightarrow \text{rel}_2(\text{term}_{2,1}, \text{term}_{2,2}, \ldots, \text{term}_{2,n^2})$

Example: Graph

Logic

$(\forall \ Y, A)$

Invest(/WB, Y, A) \(\Rightarrow\)

Invest(/JS, Y, US$ 5 \cdot 10^3$)
Rules: Conjuncts Imply Relations (1)

General: Graph (shorthand)

Logic

\( \forall \text{var}_{i,j} \)

\( rel_1(\text{term}_{1,1}, \text{term}_{1,2}, \ldots, \text{term}_{1,n^1}) \land \\ rel_2(\text{term}_{2,1}, \text{term}_{2,2}, \ldots, \text{term}_{2,n^2}) \Rightarrow \\ rel_3(\text{term}_{3,1}, \text{term}_{3,2}, \ldots, \text{term}_{3,n^3}) \)

Example: Graph

Logic

\( \forall \ Y, A \)

\( \text{Invest}(\:/\text{WB}, \ Y, \ A) \land \\ \text{Trust}(\:/\text{JS}, \ Y) \Rightarrow \\ \text{Invest}(\:/\text{JS}, \ Y, \ \text{US}\$5 \cdot 10^3) \)
Implication-Defined Predicate Odd: RuleML/XML Serialization

RuleML/XML

<Implies closure="universal">
  <And>
    <Atom>
      <Rel>Greater</Rel>
      <Var>X</Var>
      <Data>2</Data>
    </Atom>
    <Atom>
      <Rel>Prime</Rel>
      <Var>X</Var>
    </Atom>
  </And>
  <Atom>
    <Rel>Odd</Rel>
    <Var>X</Var>
  </Atom>
</Implies>

Logic

(∀ X) 
Greater(X, 2) ∧ Prime(X) ⇒ Odd(X)

Graph (prenormal)

Graph ‘⇒’ arrow normalizes to RuleML-like closure="universal"
Equivalence-Defined Predicate Even: RuleML/XML Serialization

RuleML/XML

<Equivalent oriented="yes" closure="universal">
  <Atom>
    <Rel>Even</Rel>
    <Var>X</Var>
  </Atom>
  <Atom>
    <Rel>Divisible</Rel>
    <Var>X</Var>
    <Data>2</Data>
  </Atom>
</Equivalent>
Conclusions

• Presented new version of Grailog, including feedback
• Graphical elements for box and arrow systematics, leaving color (except for IRIs) for other purposes, e.g. highlighting subgraphs (for retrieval & inference)
• Introducing Deep vs. Shallow Name Specification
• Focus on mapping to a family of logics as in RuleML
• Use cases from philosophy to technology to business
• Processing of earlier Grailog-like DRLHs studied in Lisp, FIT, and Relfun
• Now aligned with Web-rule industry standard RuleML
Future Work (1)

- Refining/extending Grailog, based on API4KB effort
  - Comparing with other graph formalisms, e.g. Conceptual Graphs ([http://conceptualstructures.org](http://conceptualstructures.org))
  - Defining mappings to/fro UML structure diagrams + OCL, adopting UML behavior diagrams ([http://www.uml.org](http://www.uml.org))
- Implementing tools, e.g. as use case for (Functional) RuleML ([http://ruleml.org/fun](http://ruleml.org/fun)) engines
  - More mappings between graphs, logic & RuleML/XML
  - Graph indexing & querying (cf. [http://www.hypergraphdb.org](http://www.hypergraphdb.org))
  - Graph transformations (normal form, typing homomorphism, merge, ...)
  - Advanced graph-theoretical operations (e.g., path tracing)
- Benefit from, and contribute to, Protégé visualization plug-ins such as Jambalaya/OntoGraf and OWLViz for OWL ontologies and Axiomé for SWRL rules
Future Work (2)

• Proceeding from the 2-dimensional (planar) Grailog to a 3-dimensional (spatial) one
  – Exploiting advantages of crossing-free layout, spatial shortcuts, and analogical representation of 3D worlds
  – Mitigating disadvantages of occlusion and of harder spatial orientation and navigation

• Considering the 4\textsuperscript{th} (temporal) dimension of animations to visualize logical inferences, graph processing, etc.

• Submitting for standardization

• See also: http://ruleml.org/#Grailog