The Grailog User Interface for Knowledge Bases of Ontologies & Rules

(Long version: http://www.cs.unb.ca/~boley/talks/RuleMLGrailog.pdf)

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**
- Combinable 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 **i**nscribed **log**ic 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)	mapping →	Logic
	deepname		*deepname*
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) —mapping→ Logic (vertical bar marks shallowness)

shallowname → |*shallowname*

Examples: Graph → Logic

WB → |WB

GE → |GE

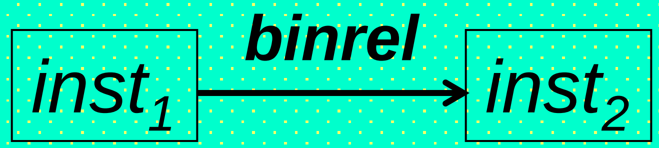
US$ 3B → |US$ 3B

Parameters: Individual Variables

General:	Graph (*hatched* node)	Logic (*italics* font, POSL uses "?" prefix)
	variable	*variable*
Examples:	Graph	Logic
	X	*X*
	Y	*Y*
	A	*A*

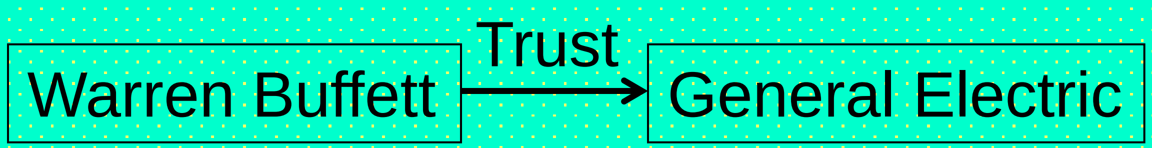
Predicates: Binary Relations (1)

General: Graph (*labeled* arc) Logic



binrel($inst_1$, $inst_2$)

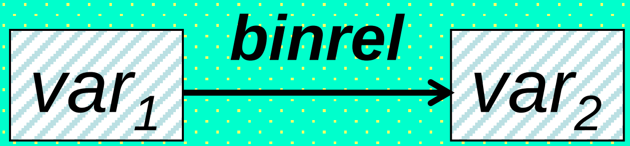
Example: Graph Logic



Trust(Warren Buffett,
General Electric
)

Predicates: Binary Relations (2)

General: Graph (*labeled* arc) Logic



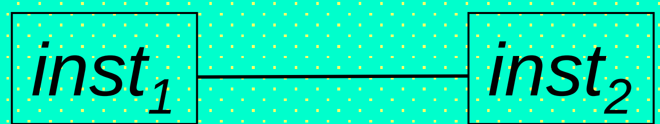
binrel(var_1, var_2)

Example: Graph Logic

Trust(*X*,*Y*)

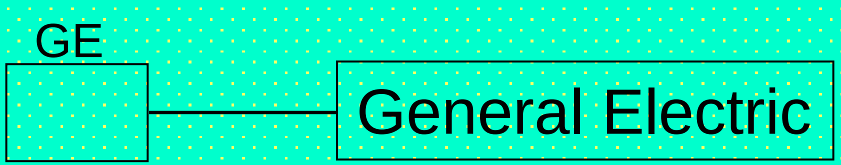
Ground Equality: Identifying Pairs of Constants

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



$inst_1 = inst_2$

Example: Graph — Logic



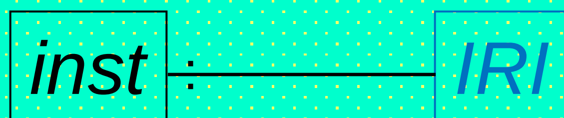
|GE = General Electric

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)

Logic (with oriented equality, webized)



inst := *IRI*

Example: Graph

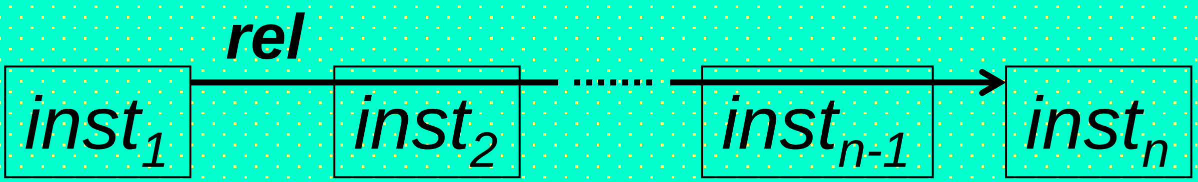
Logic

|GenEl := http://www.ge.com/

Definitional equality can also be used for the prefix part of the CURIE notation

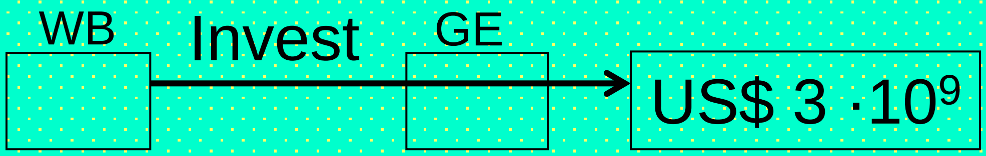
Predicates: n-ary Relations (n>1)

General: Graph (*hyper*arc) Logic



$rel(inst_1, inst_2, \ldots, inst_{n-1}, inst_n)$

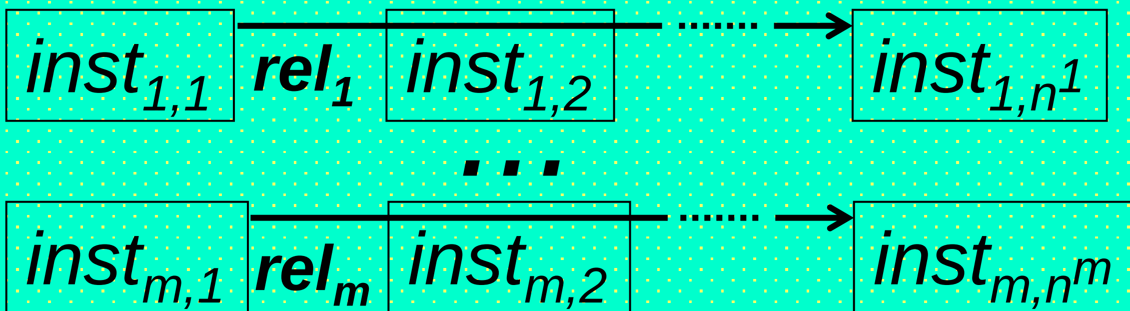
Example: (n=3) Graph Logic



Invest(|WB, |GE, US$ $3 \cdot 10^9$)

Implicit Conjunction of Formula Graphs: Co-Occurrence on Top-Level

General: Graph (m hyperarcs)



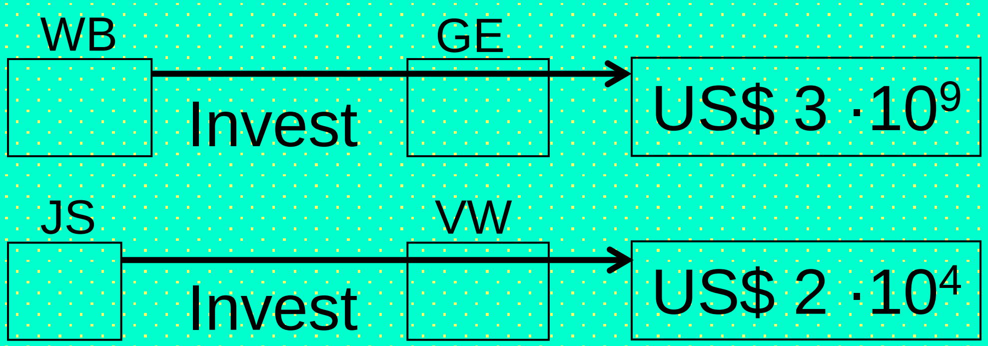
Logic

$rel_1(inst_{1,1}, inst_{1,2}, \ldots, inst_{1,n^1}) \wedge$

$\ldots \wedge$

$rel_m(inst_{m,1}, inst_{m,2}, \ldots, inst_{m,n^m})$

Example: Graph (2 hyperarcs)



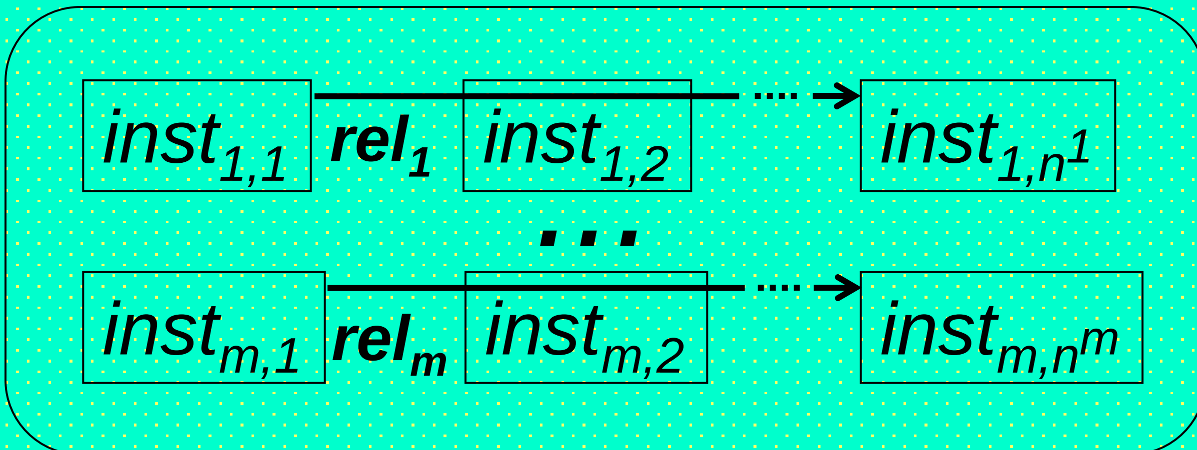
Logic

Invest(|WB, |GE, US\$ $3 \cdot 10^9$) $\wedge$

Invest(|JS, |VW, US\$ $2 \cdot 10^4$)

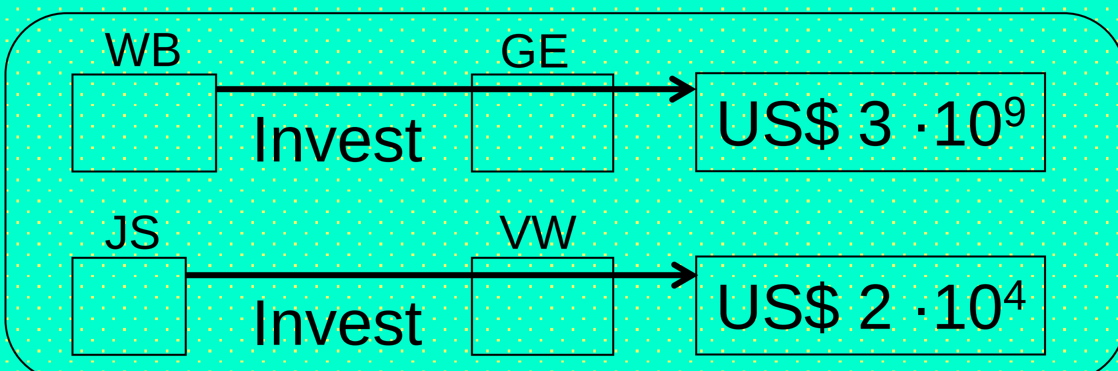
Explicit Conjunction of Formula Graphs: Co-Occurrence in Complex Node

General: Graph (*m* hyperarcs) Logic



$$(rel_1(inst_{1,1}, inst_{1,2}, \ldots, inst_{1,n^1}) \wedge \ldots \wedge rel_m(inst_{m,1}, inst_{m,2}, \ldots, inst_{m,n^m}))$$

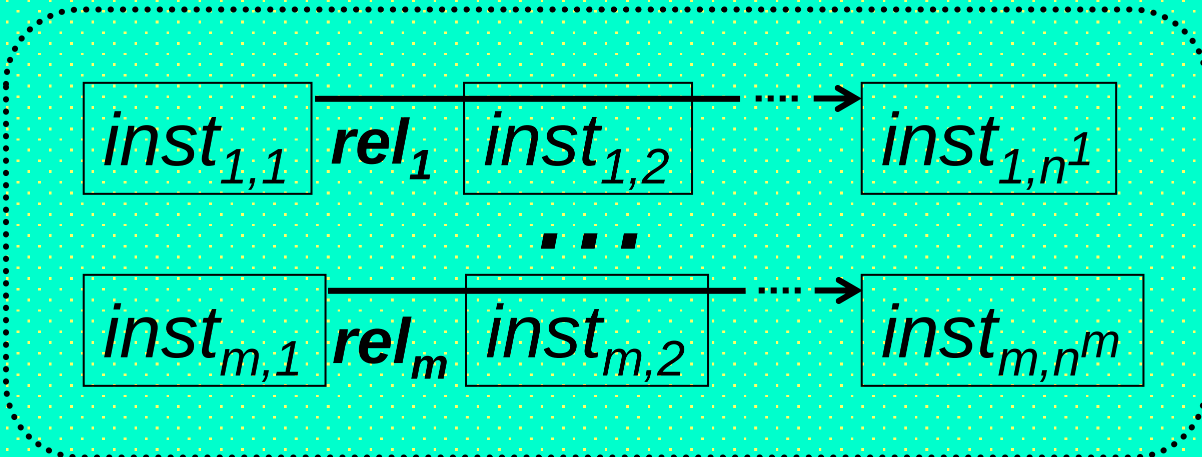
Example: Graph (2 hyperarcs) Logic



(Invest(|WB, |GE, US$ $3 \cdot 10^9$) $\wedge$ Invest(|JS, |VW, US$ $2 \cdot 10^4$))

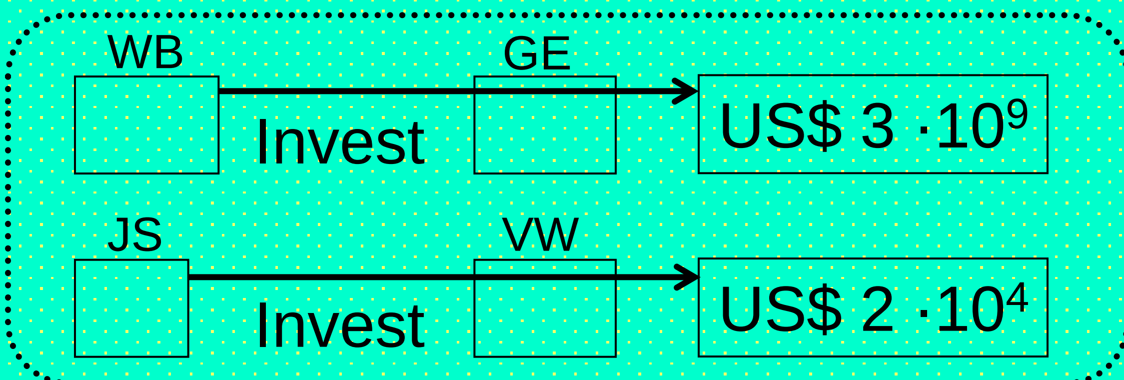
Disjunction of Formula Graphs: Co-Occurrence in Disjunctive Node

General: Graph (*dotted*) Logic



$(rel_1(inst_{1,1}, inst_{1,2}, \ldots, inst_{1,n^1}) \vee \ldots \vee rel_m(inst_{m,1}, inst_{m,2}, \ldots, inst_{m,n^m}))$

Example: Graph Logic

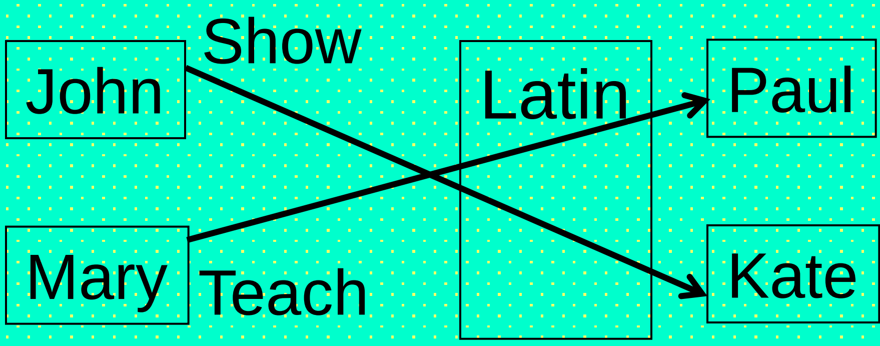


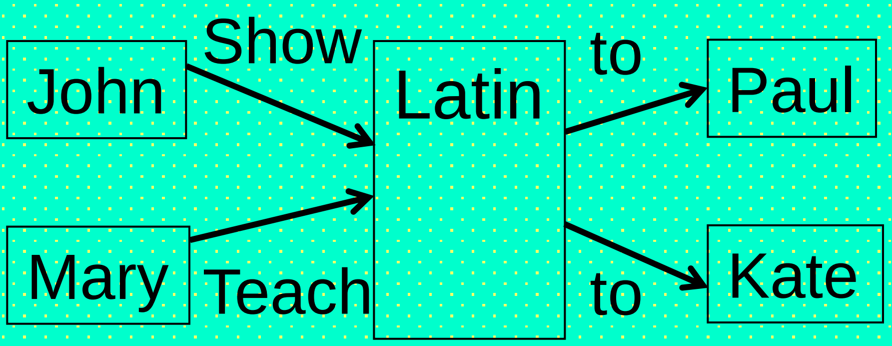
(Invest(|WB, |GE, US$ $3 \cdot 10^9$) $\vee$ Invest(|JS, |VW, US$ $2 \cdot 10^4$))

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)

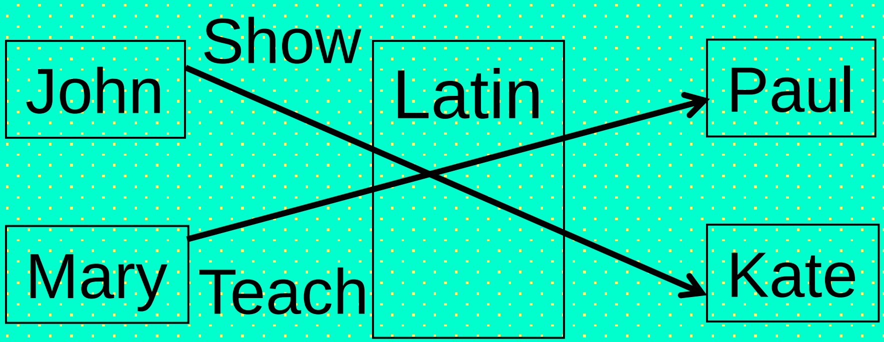


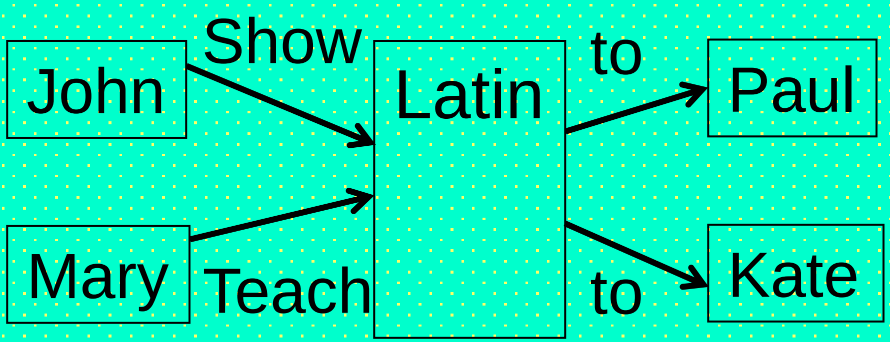


From Hyperarc Crossings to Node Copies as a Normalization Sequence (1*)

Hypergraph (2 hyperarcs, crossing inside a node)

DLG (4 arcs, do not specify to whom Latin is shown or taught)

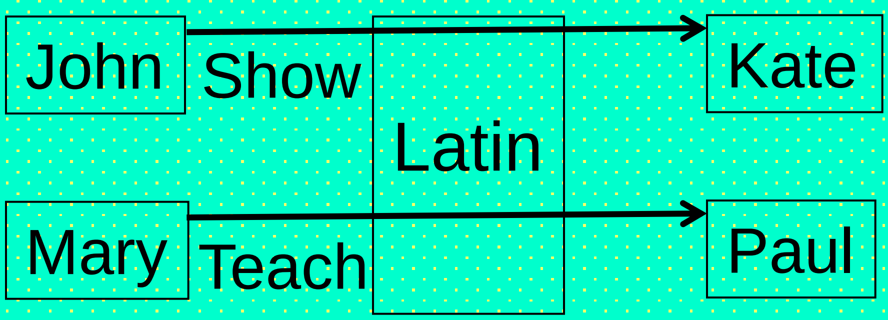


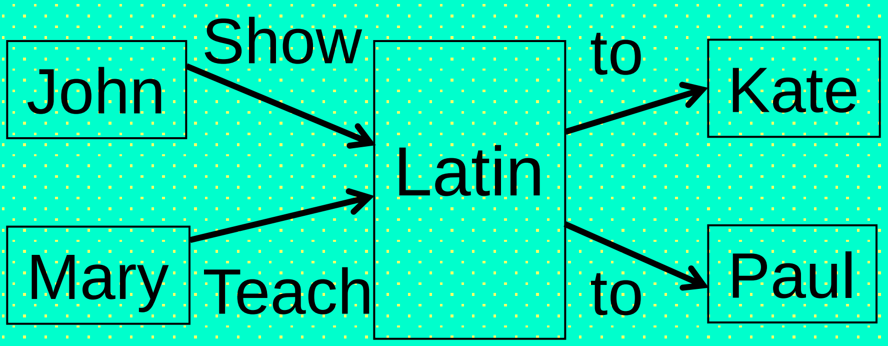


From Hyperarc Crossings to Node Copies as a Normalization Sequence (1**)

Hypergraph (2 hyperarcs, parallel-cutting a node)

DLG (4 arcs, do not specify to whom Latin is shown or taught)

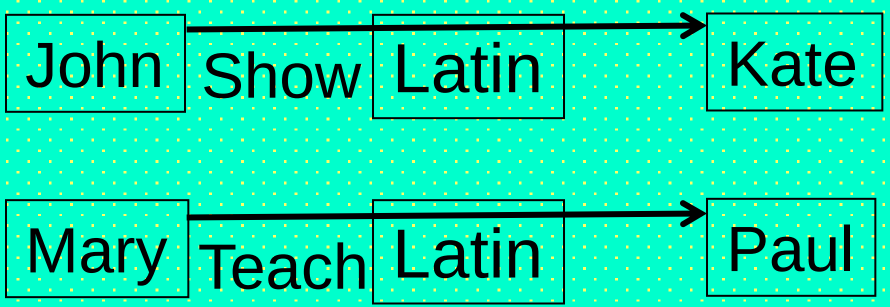




From Hyperarc Crossings to Node Copies as a Normalization Sequence (1***)

Hypergraph (2 hyperarcs, employing a node copy)

Logic (2 relations, employing a symbol copy)



Show(John, Latin, Kate)

$\wedge$

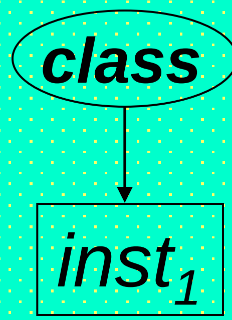
Teach(Mary, Latin, Paul)

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)

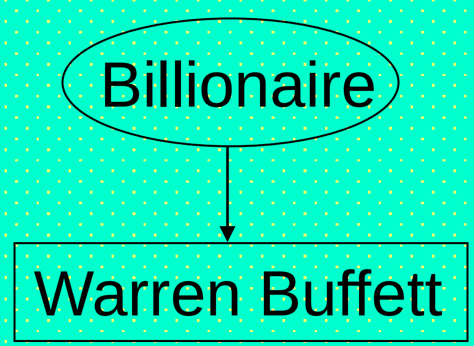
General: Graph (class applied to instance node) — Logic

↓ **hasInstance**



class(inst$_1$)

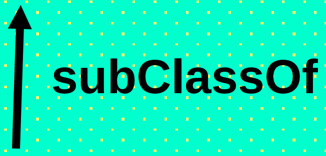
Example: Graph — Logic

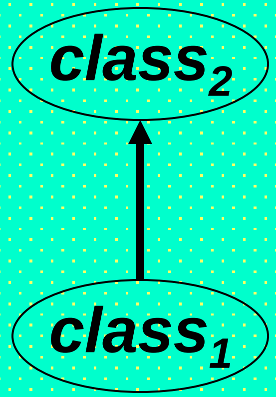


Billionaire(
Warren Buffett)

Class Hierarchies (Taxonomies): Subclass Relation

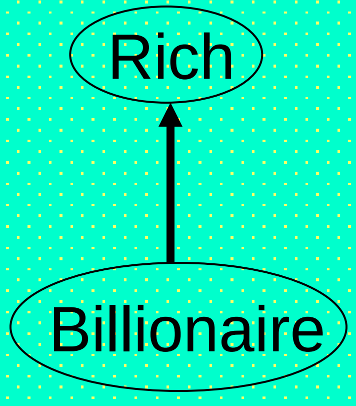
General: Graph (two nodes) (Description) Logic





$class_1 \sqsubseteq class_2$

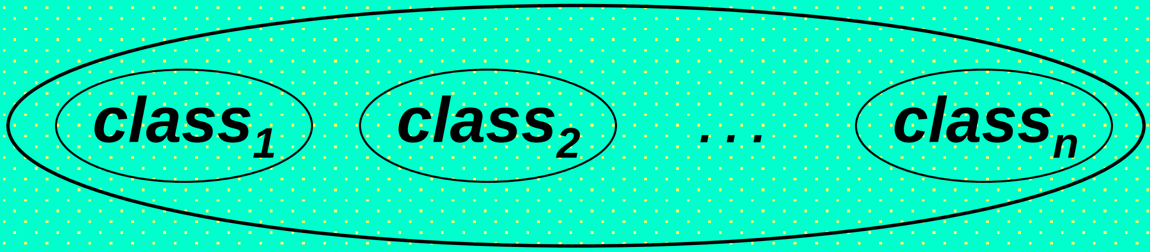
Example: Graph (Description) Logic



$\text{Billionaire} \sqsubseteq \text{Rich}$

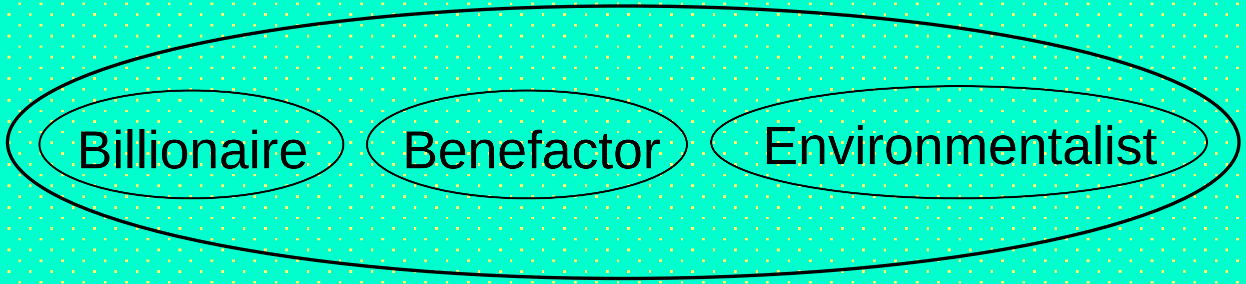
Intensional Class Constructions (Ontologies): Class Intersection

General: Graph (*solid* node, as for conjunction) — (Description) Logic



$class_1 \sqcap class_2 \sqcap \ldots \sqcap class_n$

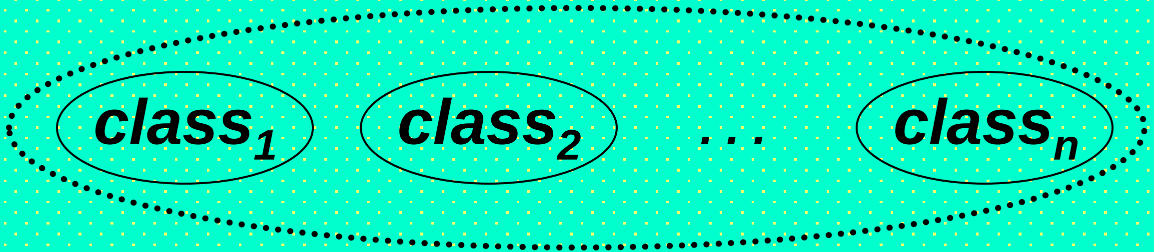
Example: Graph — (Description) Logic



Billionaire ⊓ Benefactor ⊓ Environmentalist

Intensional Class Constructions (Ontologies): Class Union

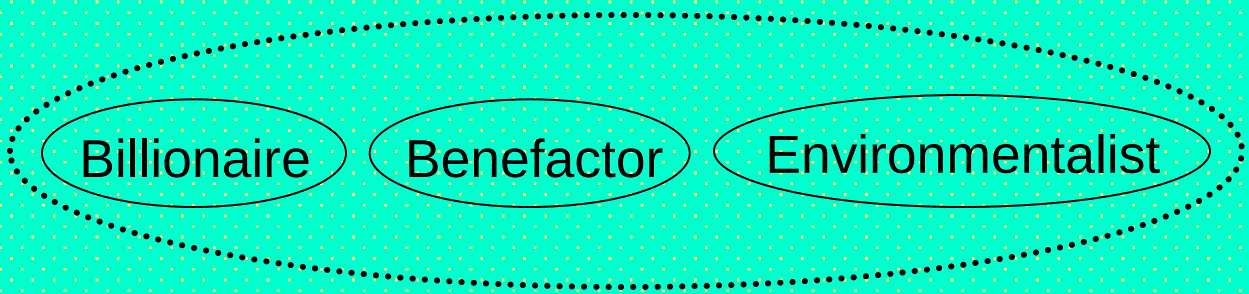
General: Graph (*dotted* node, as for disjunction)



(Description) Logic

$$class_1 \sqcup class_2 \sqcup \dots \sqcup class_n$$

Example: Graph

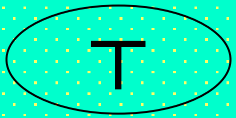


(Description) Logic

Billionaire $\sqcup$ Benefactor $\sqcup$ Environmentalist

Class Hierarchies (Taxonomy DAGs): Top and Bottom

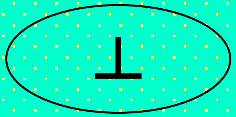
General: Top (*special* node) (Description) Logic



⊤

(owl:Thing)

General: Bottom (*special* node) (Description) Logic



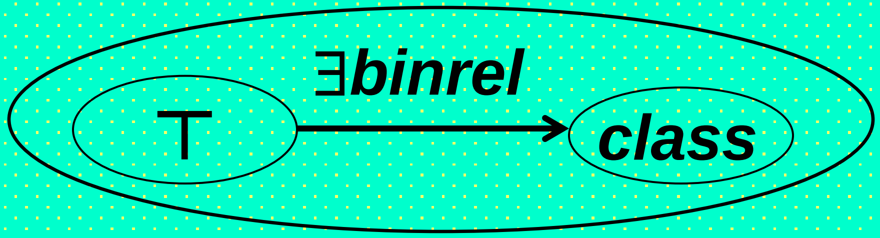
⊥

(owl:Nothing)

Intensional Class Constructions (Ontologies):

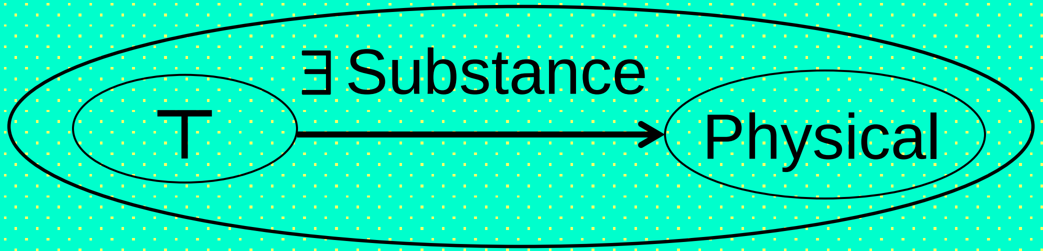
Class-Property-Restricting TBox—Existential (1*)

General: Graph (normal) (Description) Logic



∃*binrel* **.** *class*

Example: Graph (Description) Logic



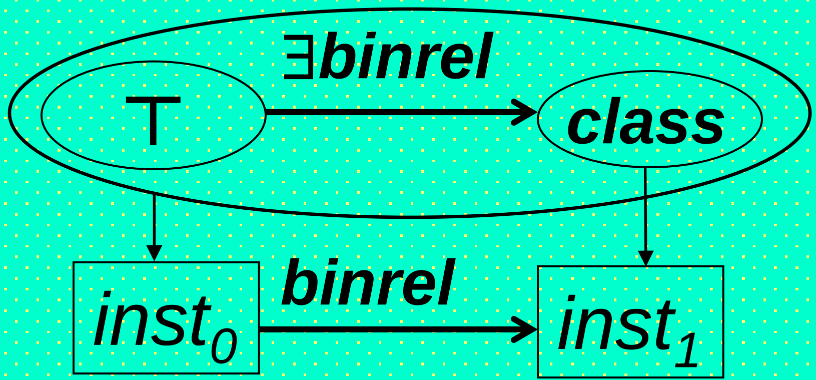
∃Substance **.** Physical

A kind of schema, where Top class is specialized to have (multi-valued) attribute/property, Substance, *with <u>at least one</u> value typed by class* Physical

Instance Assertions (Populated Ontologies):

Adding ABox to Restriction TBox—Existential (1*)

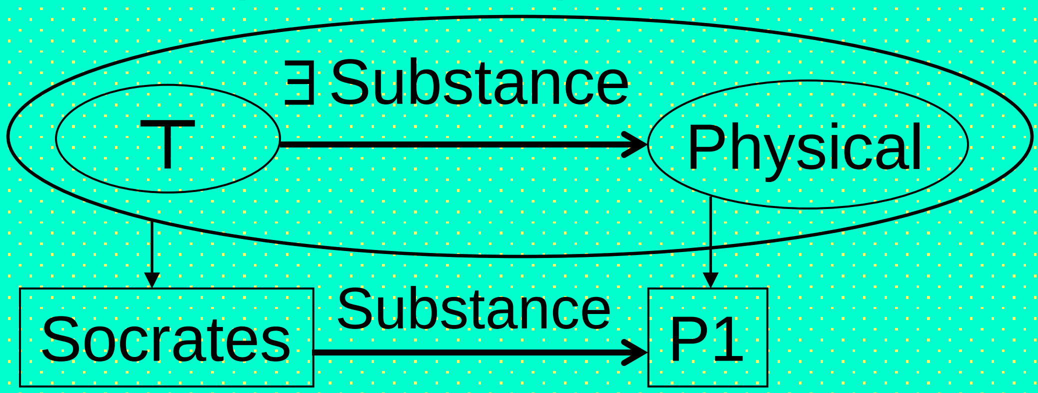
General: Graph (normal)



(rUNA-Description) Logic

$\exists binrel.class(inst_0) \wedge$
$class(inst_1) \wedge$
$binrel(inst_0, inst_1)$

Example: Graph



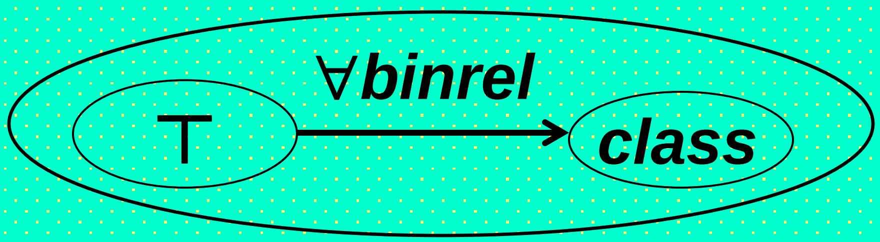
(rUNA-Description) Logic

∃Substance.Physical (Socrates) ∧
Physical(P1) ∧
Substance(Socrates, P1)

Intensional Class Constructions (Ontologies):

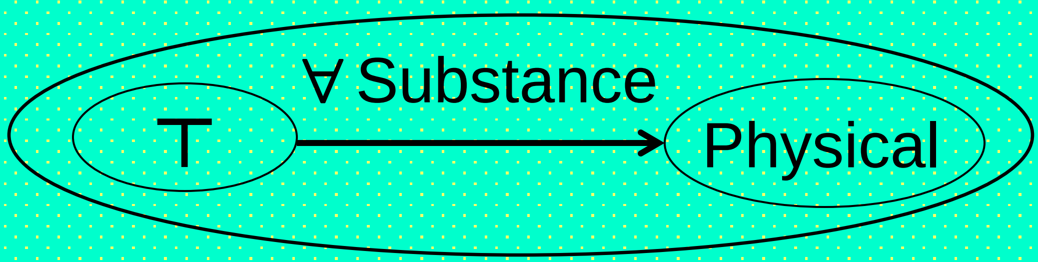
Class-Property-Restricting TBox—Universal (1*)

General: Graph (normal) (Description) Logic



$\forall$*binrel*.*class*

Example: Graph (Description) Logic

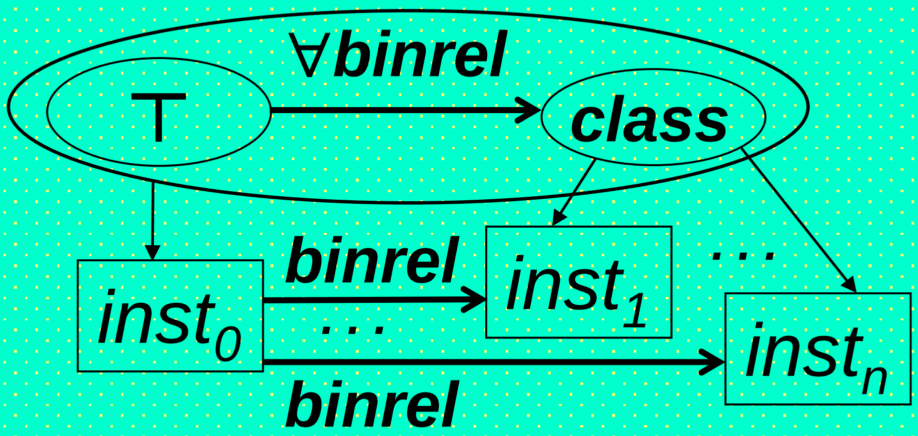


$\forall$Substance.Physical

A kind of schema, where Top class is specialized to have (multi-valued) attribute/property, Substance, *with* <u>*each*</u> *value typed by class* Physical

Instance Assertions (Populated Ontologies): Adding ABox to Restriction TBox—Universal (1*)

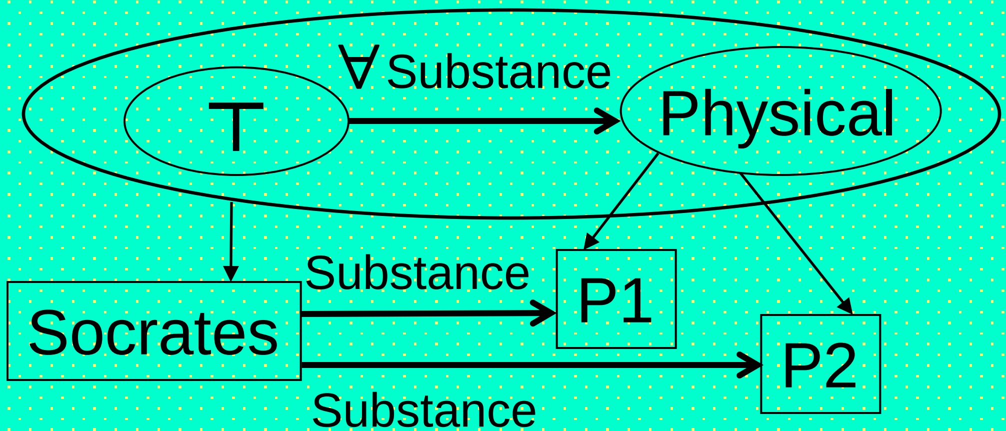
General: Graph (normal)



(rUNA-Description) Logic

$\forall binrel.class(inst_0) \wedge$
$class(inst_1) \wedge$
...
$class(inst_n) \wedge$
$binrel(inst_0, inst_1) \wedge$
...
$binrel(inst_0, inst_n)$

Example: Graph

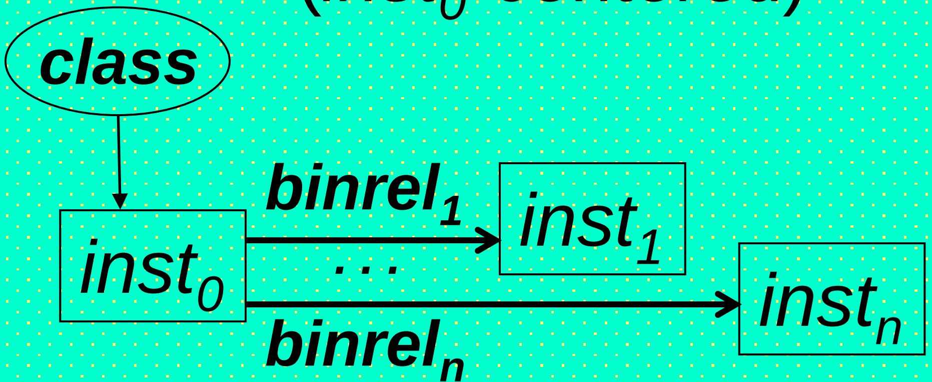


(rUNA-Description) Logic

∀Substance.Physical (Socrates) ∧
Physical(P1) ∧
Physical(P2) ∧
Substance(Socrates, P1) ∧
Substance(Socrates, P2)

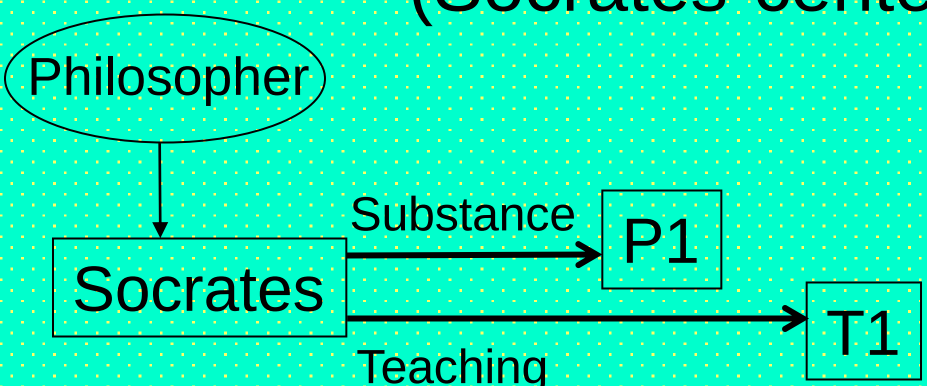
Object-Centered Logic: Grouping Binary Relations Around Instance

General: Graph ($inst_0$-centered) — (Object-Centered) Logic



$class(inst_0) \wedge$
$binrel_1(inst_0, inst_1) \wedge$
. . .
$binrel_n(inst_0, inst_n)$

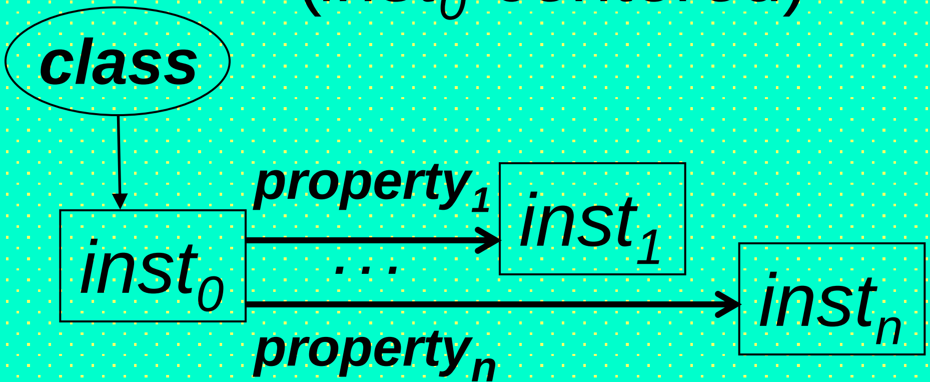
Example: Graph (Socrates-centered) — (Object-Centered) Logic



Philosopher(Socrates) $\wedge$
Substance(Socrates, P1) $\wedge$
Teaching(Socrates, T1)

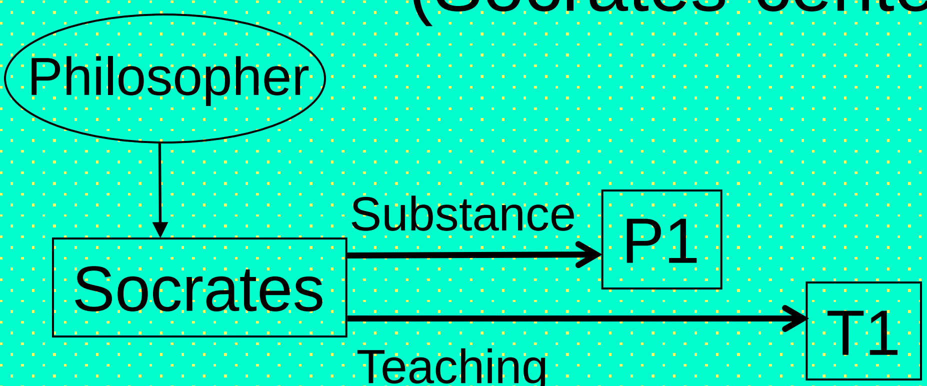
RDF-Triple ('Subject'-Centered) Logic: Grouping Properties Around Instance

General: Graph ($inst_0$-centered) | (Subject-Centered) Logic



{($inst_0$, rdf:type, *class*),
($inst_0$, $property_1$, $inst_1$),
. . .
($inst_0$, $property_n$, $inst_n$)}

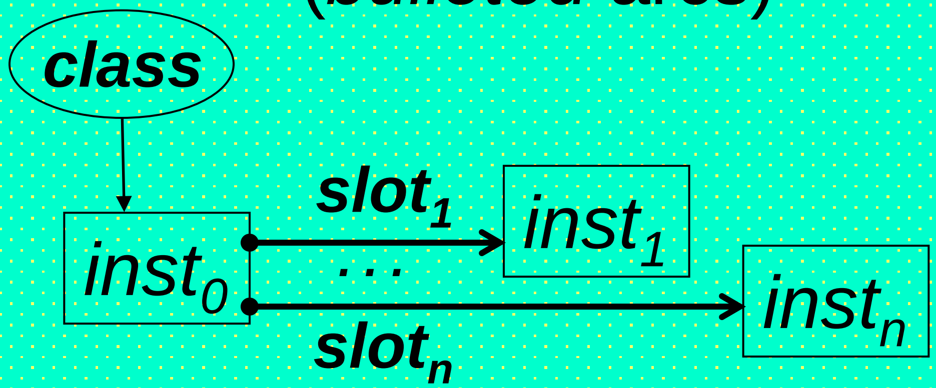
Example: Graph (Socrates-centered) | (Subject-Centered) Logic



{(Socrates, rdf:type, Philosopher),
(Socrates, Substance, P1),
(Socrates, Teaching, T1)}

Logic of Frames ('Records'): Associating Slots with OID-Distinguished Instance

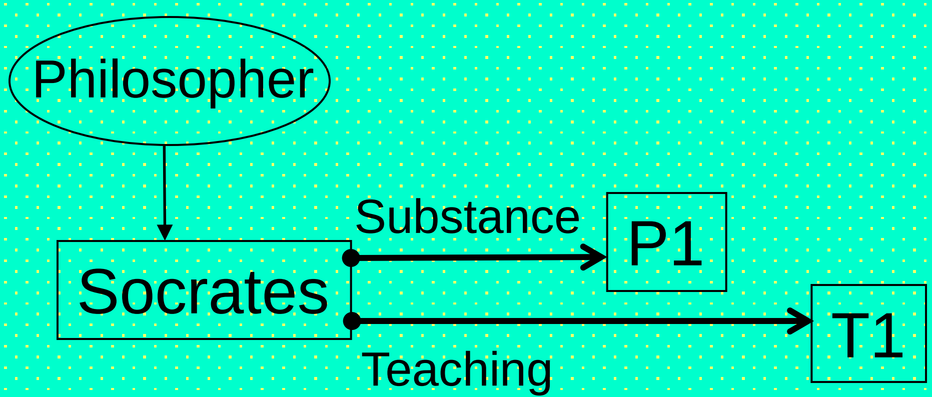
General: Graph (*bulleted* arcs) | (PSOA-like Frame) Logic



$inst_0$#$class($
$slot_1$->$inst_1$;
. . .
$slot_n$->$inst_n$)

$inst_0 \in class$,
$slot_1 = inst_1$,
. . .
$slot_n = inst_n$

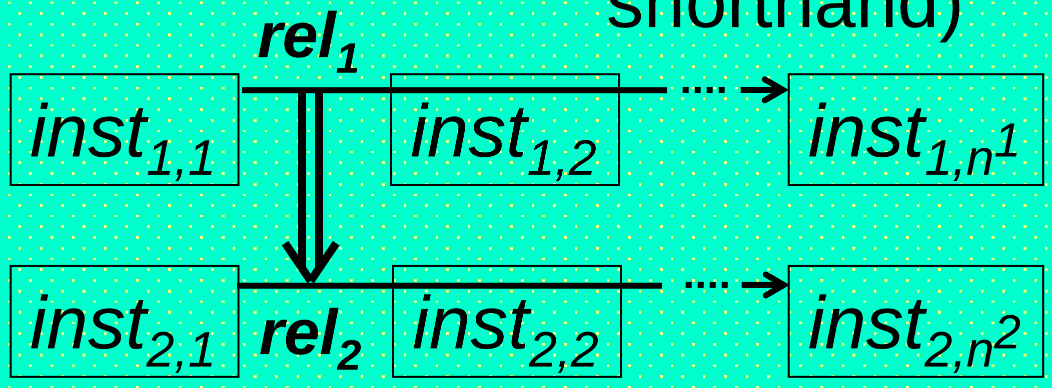
Example: Graph | (PSOA-like Frame) Logic



Socrates#Philosopher(
Substance->P1;
Teaching->T1)

Rules: Relations Imply Relations (1)

General: Graph (ground, shorthand)

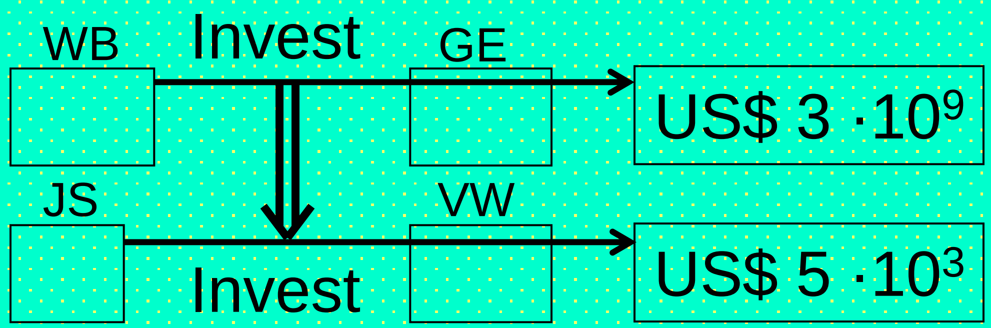


Logic

$rel_1(inst_{1,1}, inst_{1,2}, \ldots, inst_{1,n^1}) \Rightarrow$

$rel_2(inst_{2,1}, inst_{2,2}, \ldots, inst_{2,n^2})$

Example: Graph



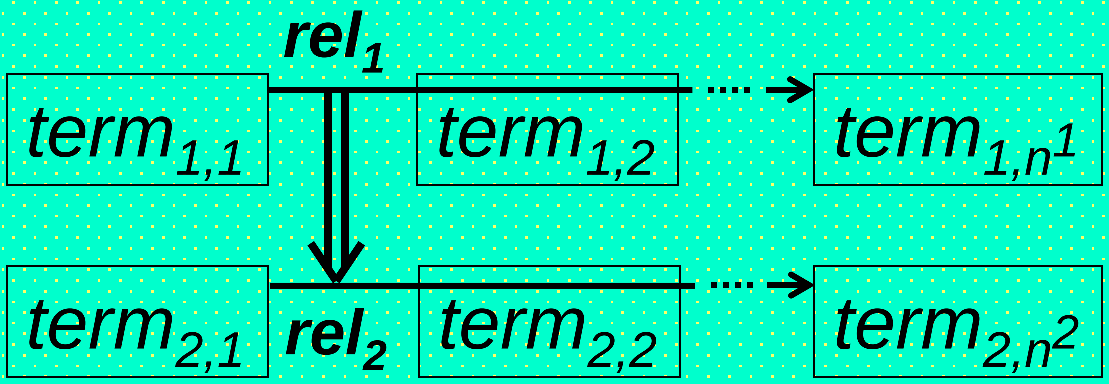
Logic

Invest(|WB, |GE, US$ $3 \cdot 10^9$) $\Rightarrow$

Invest(|JS, |VW, US$ $5 \cdot 10^3$)

Rules: Relations Imply Relations (3)

General: Graph (inst/var terms)



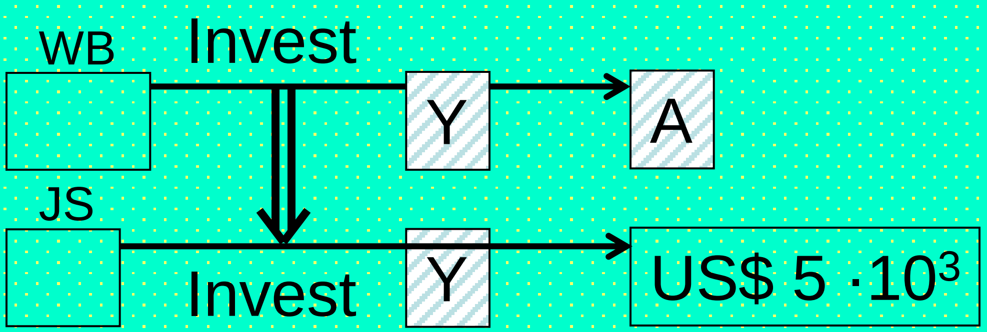
Logic

$(\forall var_{i,j})$

$rel_1(term_{1,1}, term_{1,2}, \ldots, term_{1,n^1}) \Rightarrow$

$rel_2(term_{2,1}, term_{2,2}, \ldots, term_{2,n^2})$

Example: Graph



Logic

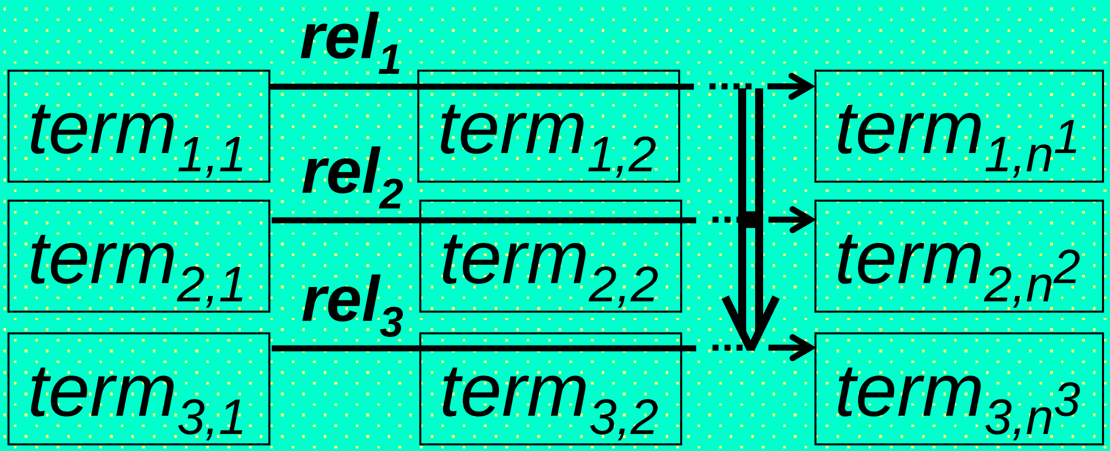
$(\forall\ Y, A)$

Invest(|WB, Y, A) $\Rightarrow$

Invest(|JS, Y, US\$ $5 \cdot 10^3$)

Rules: Conjuncts Imply Relations (1)

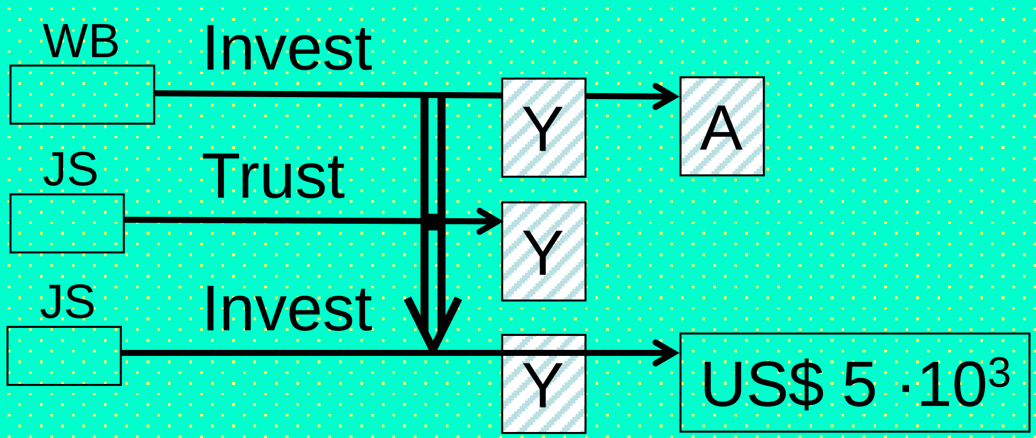
General: Graph (shorthand)



Logic

$(\forall var_{i,j})$
$rel_1(term_{1,1}, term_{1,2}, \ldots, term_{1,n^1}) \wedge$
$rel_2(term_{2,1}, term_{2,2}, \ldots, term_{2,n^2}) \Rightarrow$
$rel_3(term_{3,1}, term_{3,2}, \ldots, term_{3,n^3})$

Example: Graph



Logic

$(\forall\ Y, A)$
Invest(|WB, Y, A) $\wedge$
Trust(|JS, Y) $\Rightarrow$
Invest(|JS, Y, 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

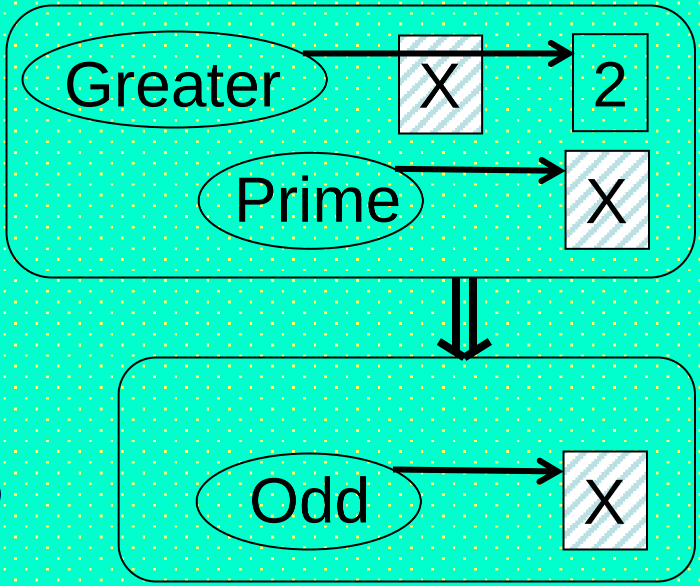
$(\forall\ X)$

$Greater(X, 2) \wedge$

$Prime(X) \Rightarrow$

$Odd(X)$

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>
```

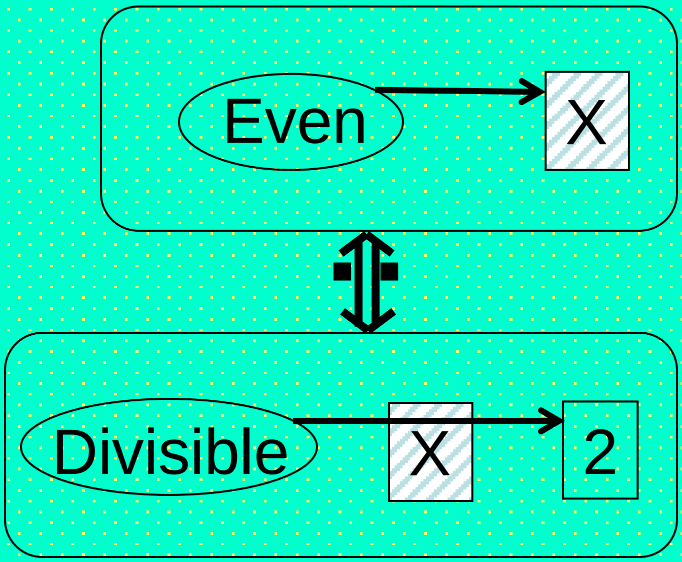
Logic

$(\forall X)$

Even(X) :⇔

Divisible(X, 2)

Graph (prenormal)



Graph ':⇔' arrow normalizes to RuleML-like closure="universal"

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*
  - E.g. “Logical Foundations of Cognitive Science”: http://www.ict.tuwien.ac.at/lva/Boley_LFCS/index.html
- *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)
  - Defining mappings to/fro UML structure diagrams + OCL, adopting UML behavior diagrams (http://www.uml.org)
- Implementing tools, e.g. as use case for (Functional) RuleML (http://ruleml.org/fun) engines
  - More mappings between graphs, logic & RuleML/XML
  - Graph indexing & querying (cf. 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 4th (temporal) dimension of animations to visualize logical inferences, graph processing, etc.
- Submitting for standardization
- See also: http://ruleml.org/#Grailog