

Numeric Reasoning in the Semantic Web

Chimène Fankam, Stéphane Jean, Guy Pierra

Presented by:

Giovanni Thenstead

Advanced Semantic Web CSCI-6965 (Spring 2009)

Rensselaer Polytechnic Institute

Feb-03-2009

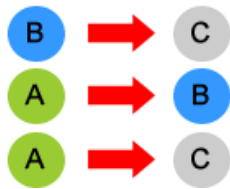
Aim of Presentation

- Introduce new ontology database for reasoning and query processing.
- Show alternative reasoning engine that utilizes various labeling techniques.
- Explore a new database architecture that supports mixed ontology and labeling schemes (i.e., a meta-scheme).
- Introduce new ontology-based query language.

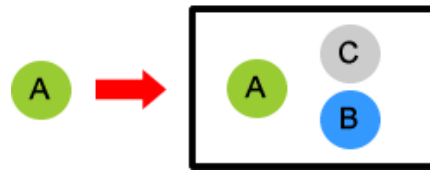
Background Knowledge

- Binary Relation Properties:

Transitivity



Reflexivity



Symmetry



- Eager and Lazy Reasoning:

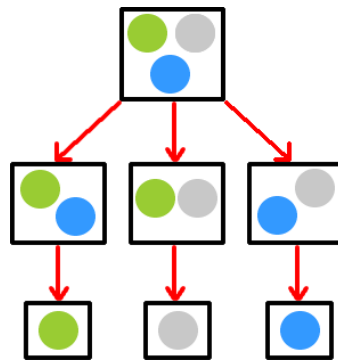
- Eager reasoning is used as an optimization technique for preprocessing deduced facts in order to speed-up future query requests.
- Lazy reasoning is the opposite of Eager reasoning whereby deduction is performed on-the-fly for each query request.

- Labeling schemes:

- Interval labeling
- Geometric labeling

Background Knowledge

- Ontology-Based Database (OBDB) Type-II
 - Class of databases that partitions (keeps separate) the ontology data tables from the instance data tables.
- Partially Ordered Sets/Tree Order for Subdivisoning



- Table Schemes:
 - Binary Table
 - Table Per Class

What is Numeric Reasoning?

Reasoning that uses number- (or string-oriented-) labeling for qualified data instances in an Type-III OBDB.

– Problems with this approach:

- Current Type-II OBDBs do not support numeric reasoning
 - However, new proposals are being made for Type-III OBDBs. These Type-III OBDBs may support:
 - » Multiple ontology models (OWL, DAML+OIL, PBLIB, etc.).
 - » New SQL-like query language.
 - » New database management system (DBMS) that can automatically transform qualified (are they transitive, antisymmetric, and reflexive? Spatial or temporal?) data instances, via labeling, into an appropriate extended form.
 - » Will add new meta-schemes (stored as new DBMS tables) for ontologies and labels.

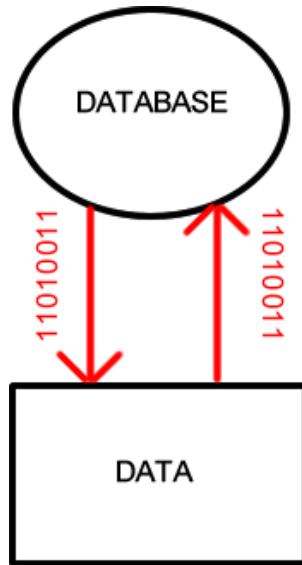
Issues with the Semantic Web

- Scaling on large-size data
 - Applications need to manage an amount of ontology data that doesn't fit in memory.
- Reasoning over large-size data
 - Often times, OBDBs that are very scalable to real world ontology and instance data are poor at reasoning, i.e., they have subpar response time when reasoning is being performed.

Why Numeric Reasoning?

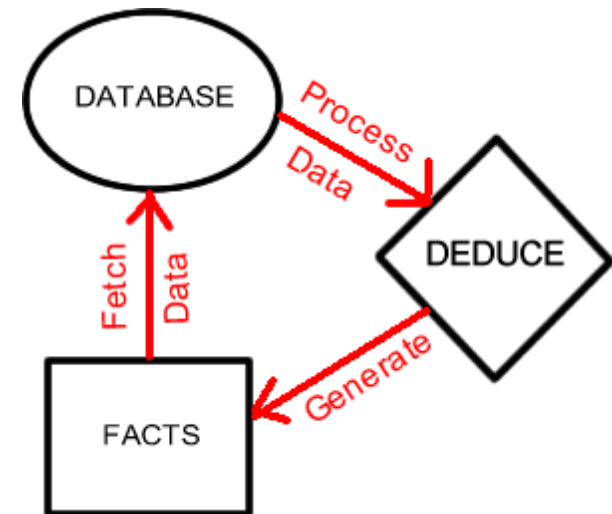
- Scalability over large-size data.
 - Labeling decreases instance data representation, making it more efficient to store in memory.
- OntoDB.
 - New Type-III OBDB that extends current Type-II OBDBs with new meta-scheming for handling multiple ontologies and labels.
- Labeling.
 - Eliminates the need for reasoning over deductive facts.
- Can exploit Eager Reasoning for optimization.
 - Because labeling is stable and requires less storage overheads, eager reasoning (preprocessing of instance data) can be used as an effective strategy for improving query processing.

Numeric Reasoning on Databases



Databases are efficient at handling large size data and also at performing numeric and string queries.

Eager reasoning can be expensive. It's better to represent data as numbers or strings instead of deduced facts which can get unstable (i.e. data size varies).



Deductive Capabilities of OBDBs

Popular approaches used for reasoning by current OBDBs:

– Eager Reasoning.

- Deduced facts such as transitive closures for all transitive relationships can be derived and then materialized (stored as, for example, database *views* – these are essentially virtual database tables created on-the-fly).
- **Drawback:**
 - requires extra storage and update overhead.

– Lazy Reasoning.

- Deduction is done on-the-fly using virtual deduced facts.
- Done on various database mechanisms such as views, labeling schemes, or sub-table relationships found on object-relational databases.
- **Drawback:**
 - requires extra processing cost but doesn't impose storage and update overheads.

– Subsumption Reasoning.

- Data instances that establish a subclass-like relationship with respect to another (i.e., *classOf*, *instanceOf* relationships.) – this can be useful for carrying-out transitive closures.

OBDB Approaches: Type-I OBDB

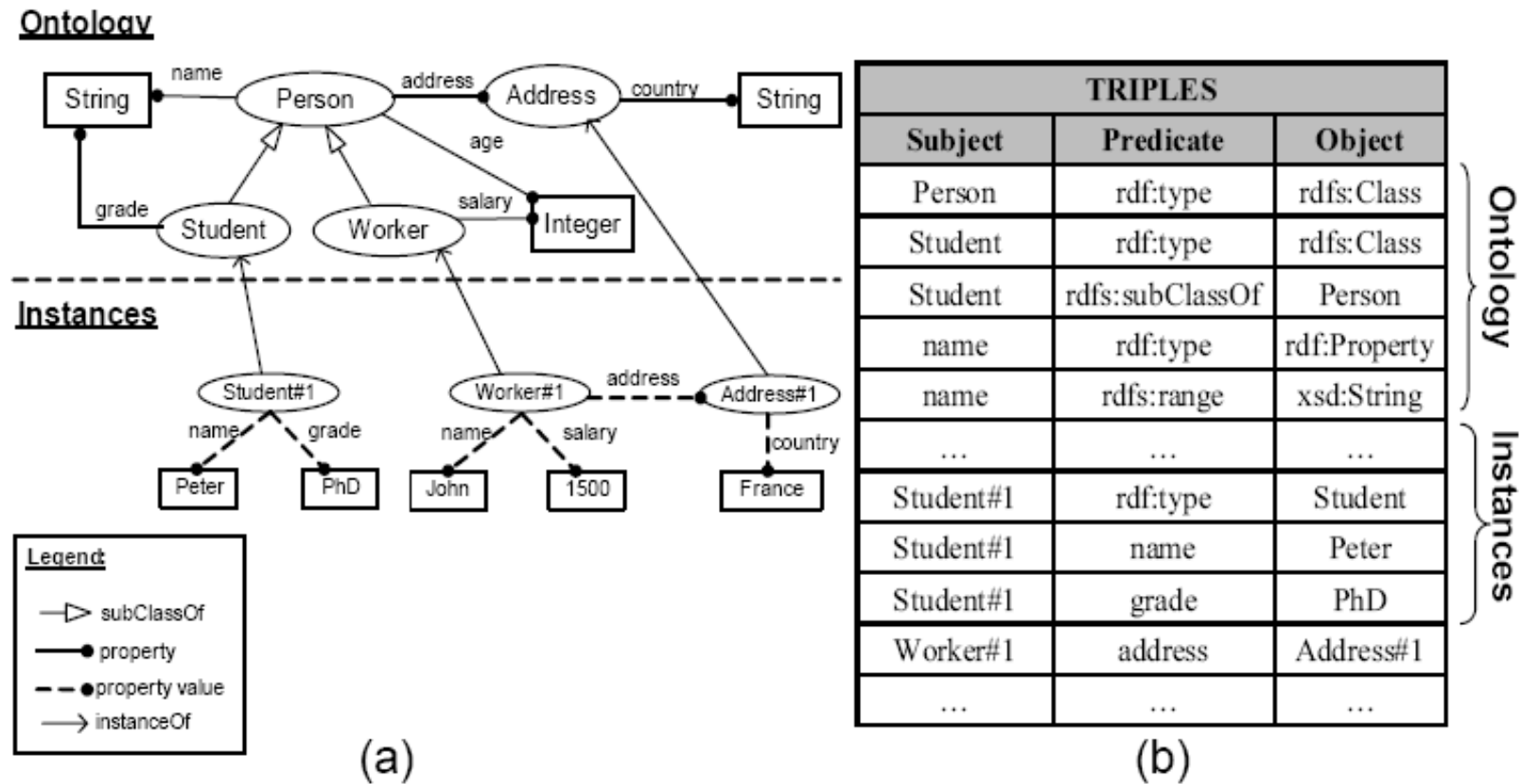


Fig. 1. Type 1 OBDBs approach

OBDB Approaches: Type-II OBDB

Ontology

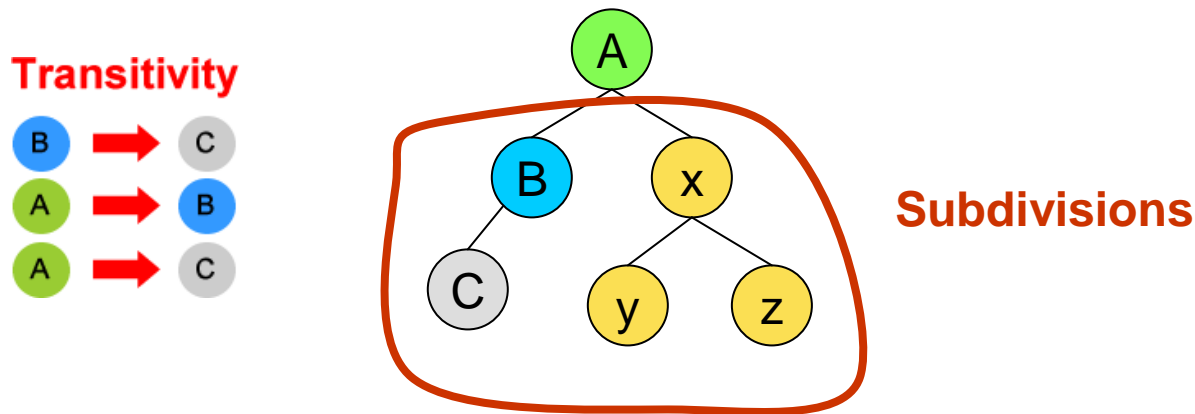
Class		SubClassOf		Property		Domain		Range	
ID	Name	Sub	Sup	ID	Name	prop	class	prop	type
1	Person	2	1	1	name	1	1	1	xsd:string
2	Student	3	1	2	age	2	1	2	xsd:integer
3	Worker			3	grade	3	2	3	xsd:string
4	Address		

Instances

Person	Student	Name	Grade
ID	ID	ID	Value
	Student#1	Student#1	Peter
		Worker#1	John
Address	Worker	Salary	Country
ID	ID	ID	Value
Address#1	Worker#1	Worker#1	1500
		Address#1	France

Fig. 2. Type 2 OBDBs approach

Partial Order/Tree-Order



By the properties of **Transitivity** and **Inclusion**:

- **Subdivisions** can be established based on the topographic hierarchy.
 - As such, a partial order on these subdivisions can be used to exploit their positional-relationship to create some range (i.e. unique interval under which a subdivision's domain falls in respect to its scope in an hierarchy).

Numeric Reasoning over Partially Ordered Sets

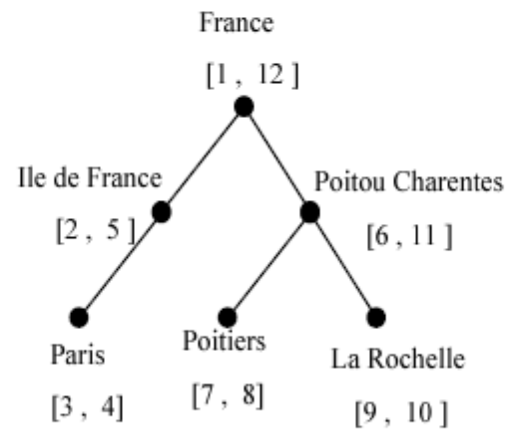
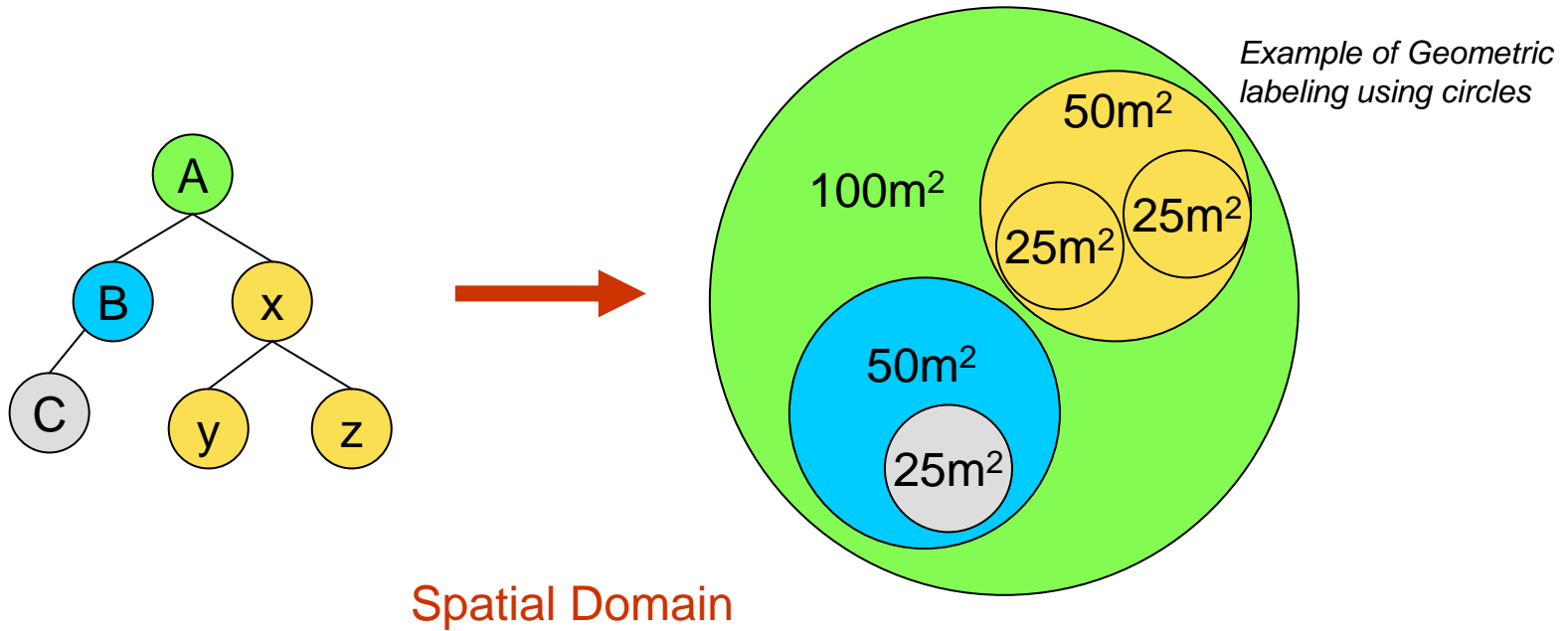
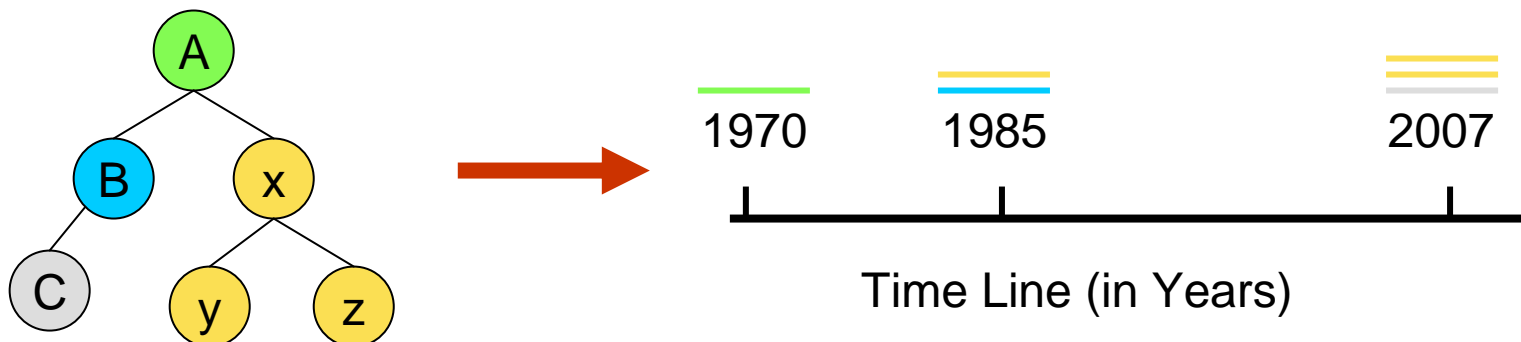


Fig. 5. translation of a tree structure into two numeric values

Numeric Reasoning over Spatial and Temporal Domains



Temporal Domain



Implementation: Labeling Scheme

LABELING SCHEME						
schemeld	numberOfColumns	listColumnsNames	listColumnsTypes	label	less or eq	defaultScheme
5	2	{bound1, bound2}	{int, int}	interval	include	TRUE
geo_rectangle	4	{xmin, xmax, ymin, ymax}	{float, float, float, float}	NULL	MBRContains	FALSE

PROPERTY SCHEMES			
propId	schemeld	listProperties	activeScheme
3	5	{7,8}	TRUE
4	*geo_rectangle*	{9,10,11,12}	FALSE

Query Processing

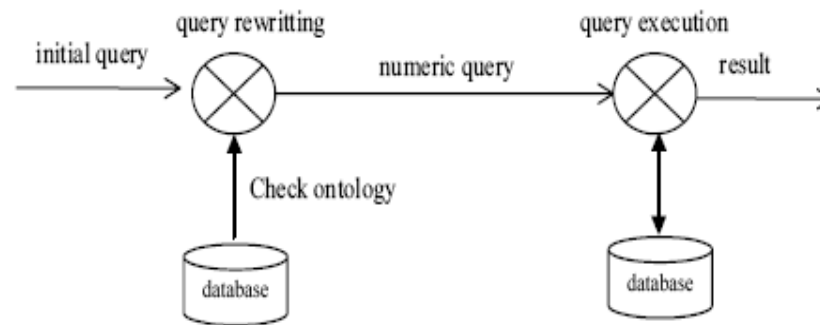


Fig. 6. Query processing steps.

Query processing on a *Table Per Class* backend can be 10 times as fast compared to its *Binary Represented* counterpart – and provided the queries are being carried out with a class name specified.

Implementation: Query Rewriting

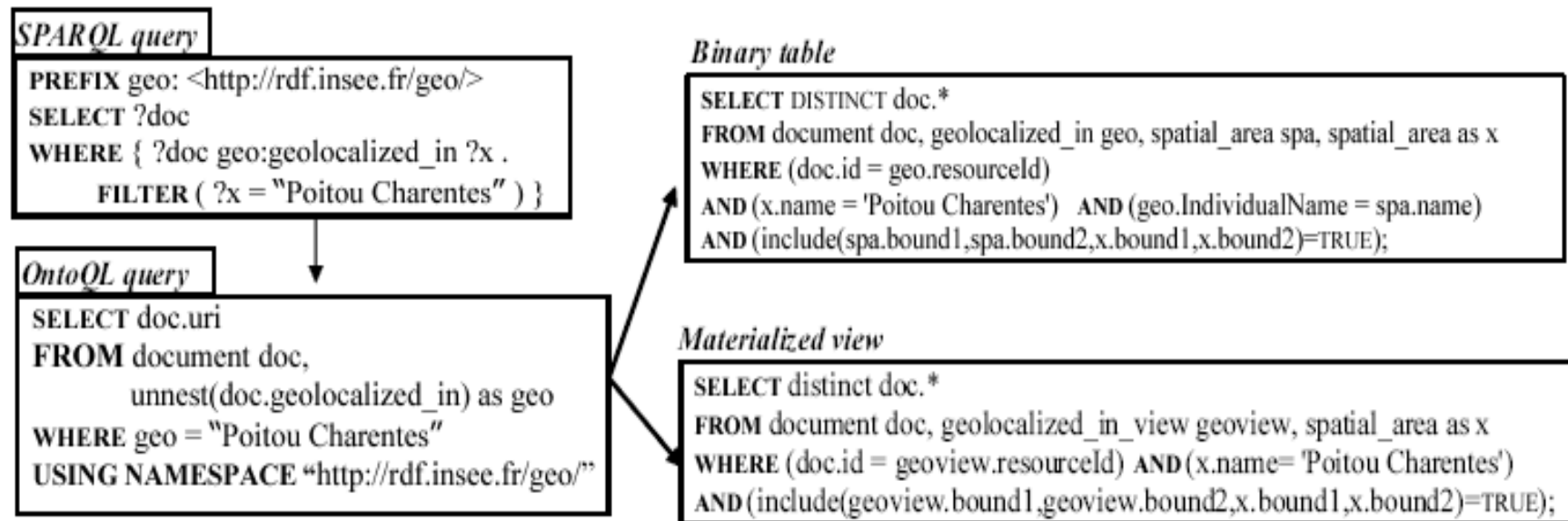


Fig. 11. COG - OntoDB : example query rewriting

Queries that are not apart of OntoQL (proposed Type-III OBDB query language) are automatically converted to their OntoQL counterparts. This is important so that the correct ontologies in the OntoDB DBMS are queried, given their namespace.

Implementation: Type-III OBDB

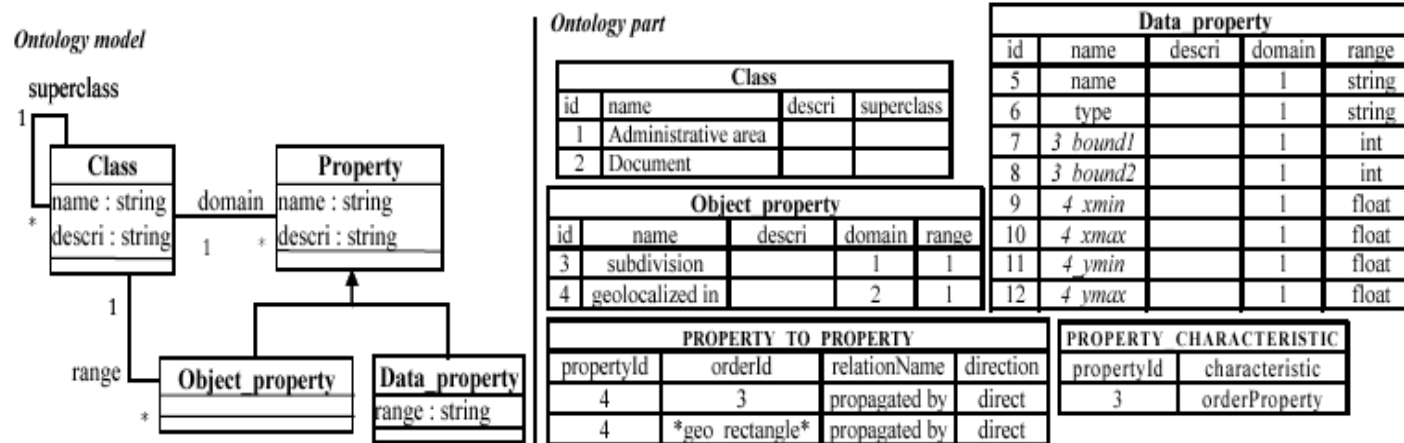


Fig. 7. COG - OntoDB : ontology part

subdivision	
Subdivision1	Subdivision2
France	Ile de france
France	Poitou Charentes
Ile de france	Paris
Poitou Charentes	La rochelle
Poitou Charentes	Poitiers

spatial area					
id	name	type	...	bound1	bound2
11	France	country		1	12
12	Ile de France	department		2	5
13	Paris	town		3	4
14	Poitou Charentes	department		6	11
15	Poitiers	town		7	8
16	La Rochelle	town		9	10

Document
id
21
22
23

Fig. 9. COG - OntoDB : the data part

Questions

- Does the labeling scheme support more complex geographic features (i.e., with holes, or irregular geometries) than what is provided?

Gregory Todd Williams

- While it may be possible to extend the current labeling scheme definitions (via the `label_schemes` table) to include other shapes, what you are asking does not appear to be a possibility that can be achieved given the unique dynamics properties that would have to be considered.

- How is sub-classing supported?

Jesse Weaver

- As mentioned earlier, this can be achieved through *Subsumption Reasoning* which utilizes transitivity closures on derived data instances with antisymmetric, transitive, and reflexive properties.

- How reliable is the automatic transformation process for non-OntoDB ontology and instance data?

Joshua Shinavier

- There is a default mechanism, so if no implementation entry (ontology scheme) exists for an unknown instance data, then whatever tricks available to the DBMS for default events may be applied.

Application to Research

- This may apply to my research interest, “Distributed Reasoning,” in the follow ways:
 - In the case of interval labels:
 - finding possible vertical partitioning-points may be used to help distribute workload across multiple reasoners.
 - Maybe interesting to distribute the underlying OntoDB DBMS so that instead of storing the ontology and their respective instance data on one engine, they may be spread across multiple reasoning engines

References

- Chimène Fankam, Stéphane Jean, Guy Pierra. "Numeric reasoning in the Semantic Web." *SeMMA* 346(2008): 84-103.

Questions?