Leveraging Data and Structure in Ontology Integration

O. Udrea L. Getoor R.J. Miller

Group 15
Enrico Savioli Andrea Reale Andrea Sorbini

DEIS
University of Bologna

Searching Information in Large Spaces Conference
March 11, 2009

Outline

Ontologies

a quest for meaning

- A very common problem in IT is **data modeling**
  - Critical in the field of Information Systems
  - A good data model enables good data usage
- Tools used to describe concepts *cannot express* the **implicit interpretation** of data
- *Semantic knowledge* is lost after the modeling process

Example

```
<musician name="F. Mercury" sings-for="Queen" />
<musician name="B. May" plays-for="Queen" />
```

- How can we find all of Queen's members?

Ontologies

a formal model

Ontology

An ontology is a *formal representation* of

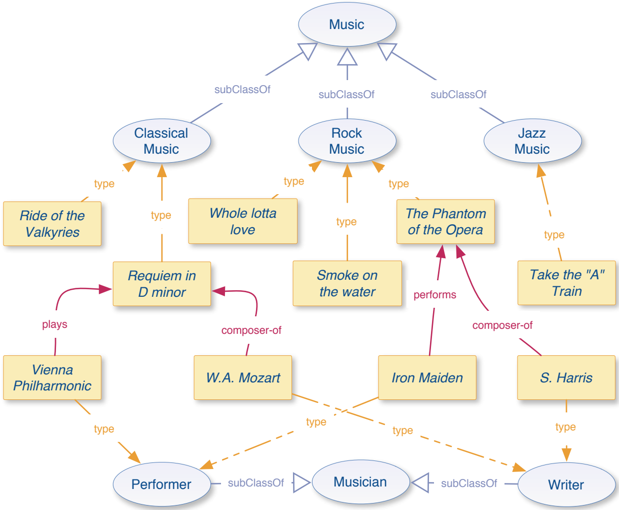
- **concepts** within a *domain* (Song, Performer, Composer...)
- **relations** between those concepts (plays-for, member-of, written-by...)
- individual **instances** (queen, f.mercury, b.may...)

- *Well-defined constructs* to enrich descriptions with semantics
- Formal models to enable *automatic reasoning* (**DL**, **FOL**)
  (member(Y,Z) :- sings-for(Y,Z) ; plays-for(Y,Z))

- **OWL** is the W3C standard for an *ontology language*

OWL

A simple example



Motivating Scenario

The AAA principle

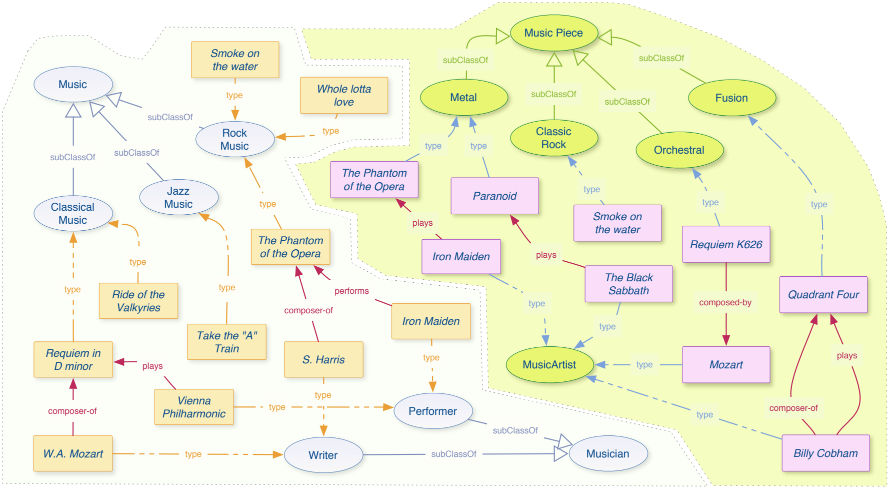
Anyone *can say* ***Anything*** *about* ***Any topic***

- The same thing might be described in different ways
- There is a strong need to *integrate heterogeneous schemas*
  - Not only in a web scenario

Example

- DB schemas
- XML schemas
- Ontology schemas

An Integration Problem



Integration Techniques

Structure Based Matching

Uses schema *meta-data* (e.g. informations about tables or concepts models) to discover mapping elements among them, both on a structural and element level

Instance Based Matching

Uses informations about *data instances* (e.g. contents of a table or individuals of an ontology) to discover mappings among entities representing them

Ontology Integration

- **Ontologies offer *further ways*** to uncover aligning relations between entities
- **Explicit theoretic model semantics** can be leveraged to improve integration quality

Example

- Assume that *composed-by* is a functional property
- *"Requiem K626"* and *"Requiem in D minor"* are the same
- If we have:
  - composed-by('Requiem K626', 'Mozart').
  - composed-by('Requiem in D minor', 'W.A. Mozart').
- Then we might say that aligning *"W.A. Mozart"* and *"Mozart"* is a good choice

FCA-Merge and COMA++

FCA-Merge (human-aided tool to merge ontologies)

- Collects domain related natural language documents
- Searches those documents for ontologies' concepts occurrences
- Derives an alignment that *has to be* validated by an operator

COMA++ (framework with multiple match strategies)

- Fragment based matching
- Reuse of previous matching results
- Comprehensive GUI for results evaluation

What's missing

- Both COMA++ and FCA-Merge use *only structure level matching*
- Moreover none of them makes use of the semantic potential offered by ontologies
- There exist other solutions using **reasoning support**, however ...
  - it is used only for an *a-posteriori consistency check* of the result
- An interesting improvement might be to *leverage it for the actual matching process*
- **That's where ILIADS kicks in!**

ILIADS

Integrated Learning In Alignment of Data and Schema

- Takes **two OWL Lite ontologies** as *input*
- *Combines* "traditional" schema matching approaches with a **logical inference algorithm**
  - Inference results are *used to influence confidence* in a presumed mapping
- Makes use of both schema (structure) and data (individuals)
- *Outputs* a set of axioms (the **alignment**) that tights the input ontologies together

Algorithm Overview

INPUT: Consistent Ontologies O_1 and O_2
OUTPUT: Alignment A^*

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

The Algorithm

Structures initialization

- The algorithms groups equivalent entities in **clusters**
- Clusters are classified by the type of their entities
  - Clusters of *Classes*
  - Clusters of *Properties*
  - Clusters of *Individuals*
- A new alignment can result in
  - **Merging** of clusters
  - A new **subsumption** relationship between clusters
- At the beginning a cluster is created for each entity

The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

The Algorithm

Similarity Computation

Entities similarity score

$$sim(e, e') = \lambda_x \cdot sim_{lex}(e, e') + \lambda_s \cdot sim_{struct}(e, e') + \lambda_e \cdot sim_{ext}(e, e')$$

- **lexical**: Jaro-Winkler (similar to edit distance) and thesauri
- **structural**: Jaccard for *neighborhoods* ($Jac_d(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$)
- **extensional**: Jaccard on *extensions*

- The set of parameters $\{\lambda_x, \lambda_s, \lambda_e\}$ is *different for each entity type*

- Similarity between clusters is computed combining the similarities of their entities

The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

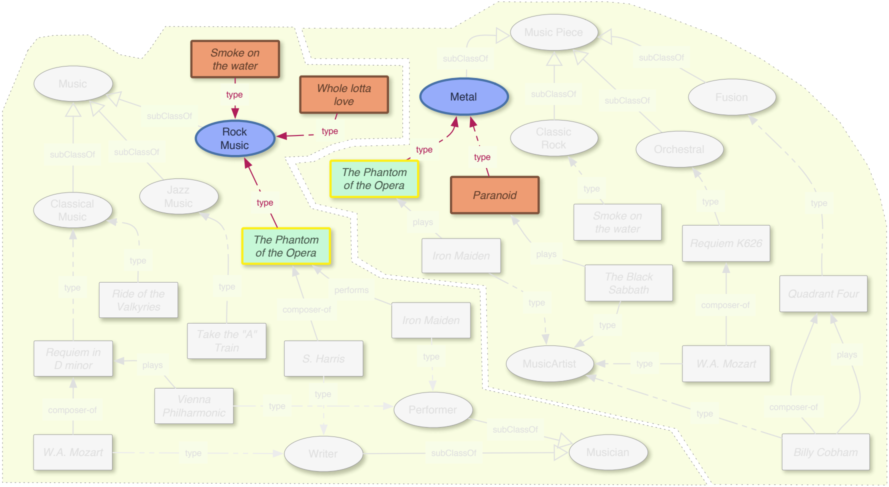
The Algorithm

Selecting a relationship

- Iterates over each couple (c, c') such that $sim(c, c')$ is above a **threshold** λ_t
- For each of those a candidate relationship is chosen between:
  - **Equivalence**
    (Two concepts are said to be equivalent if they denote the same concept)
  - **Subsumption**
    (A concept subsumes another concept if it always denotes a superset of the second)
- The selection is done by looking at the intersection of entities' *extensions*:
  - The set of its instance individuals, for a class
  - The couples of individuals involved, for a property

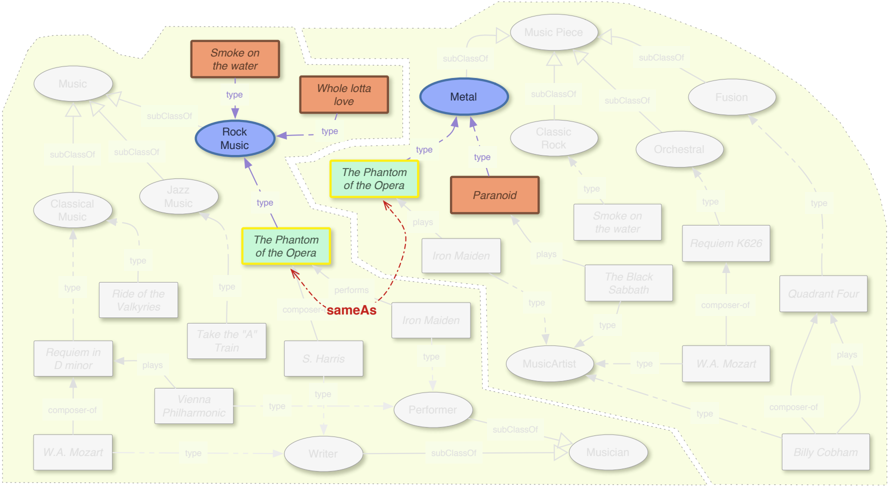
The Algorithm

Selecting a relationship - Example (1)



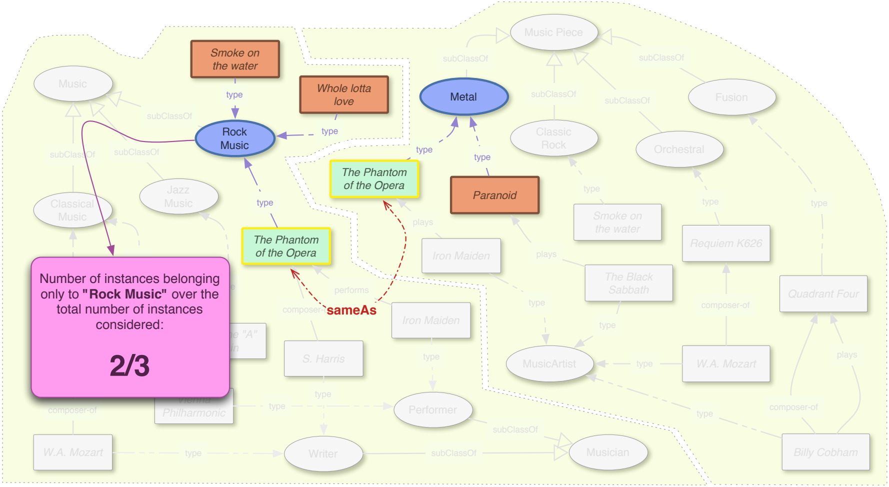
The Algorithm

Selecting a relationship - Example (2)



The Algorithm

Selecting a relationship - Example (3)

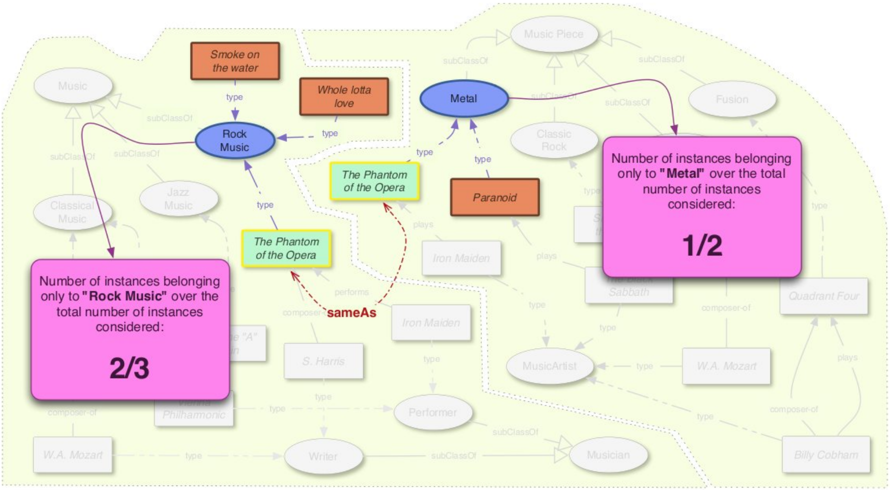


Number of instances belonging only to **"Rock Music"** over the total number of instances considered:

2/3

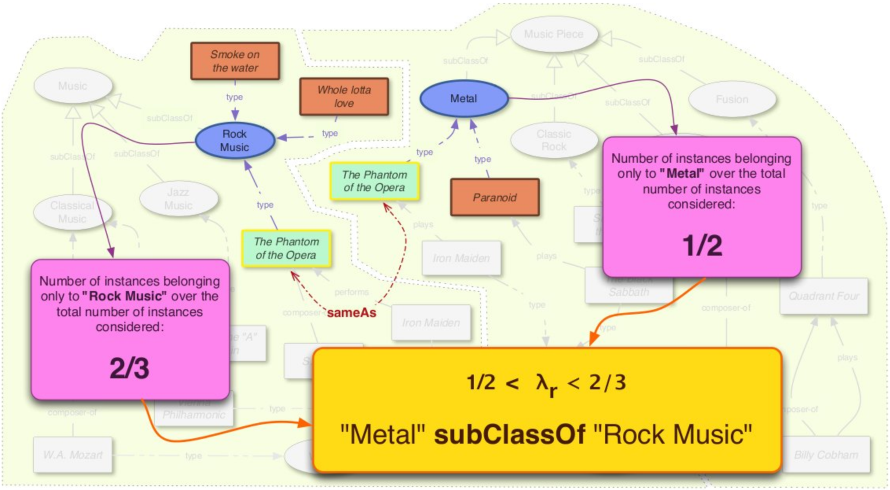
The Algorithm

Selecting a relationship - Example (4)



The Algorithm

Selecting a relationship - Example (5)



The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

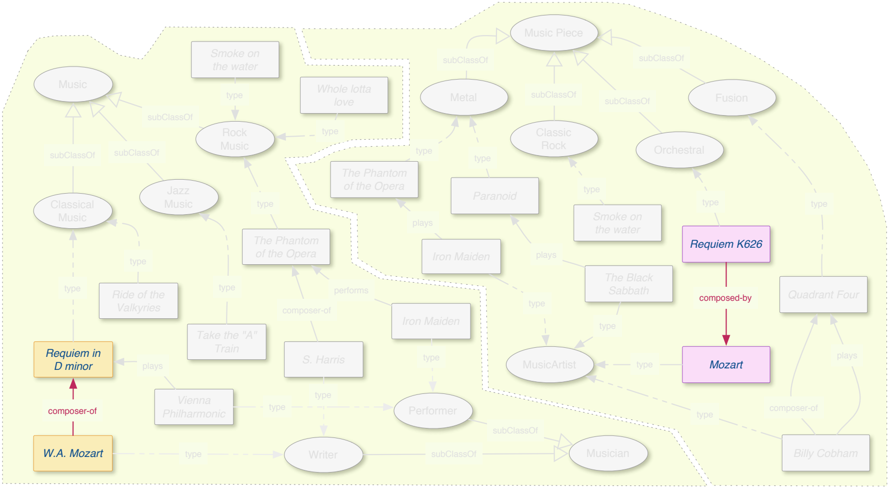
The Algorithm

Incremental Logical Inference

- The inference step is used to:
  - Look for **inconsistencies** of the candidate relationship
  - Infer **logical consequences** of the new axiom
  - Possibly enforce the confidence in it
- The inference is *not complete* (it would be EXPTIME in OWL Lite)
  - Only a *small number of steps* is actually performed
  - However this may cause inconsistencies not to be found

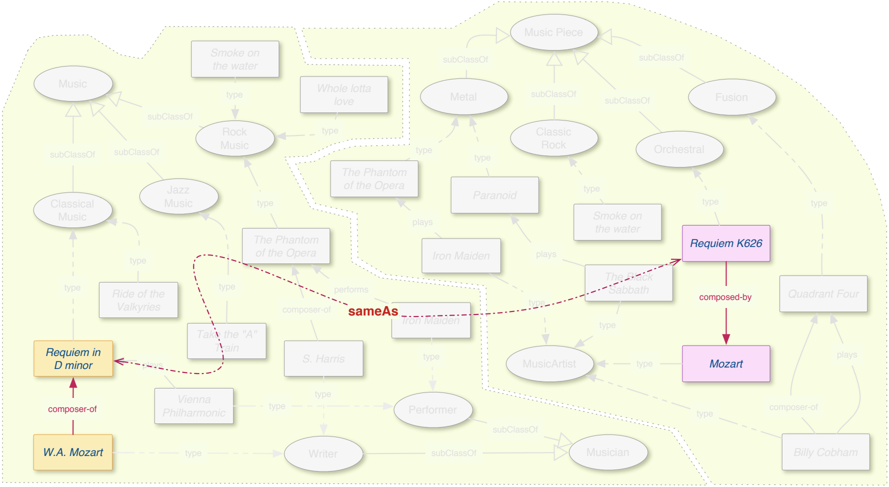
The Algorithm

Incremental Logical Inference - Example (1)



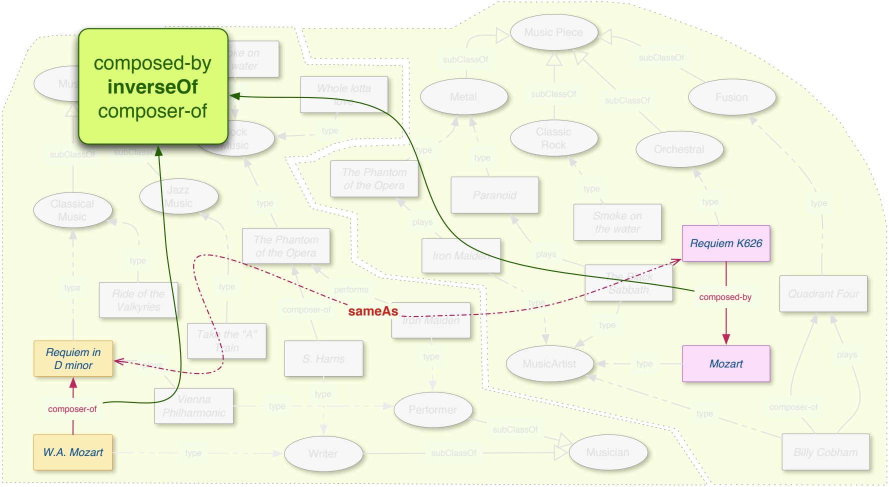
The Algorithm

Incremental Logical Inference - Example (2)



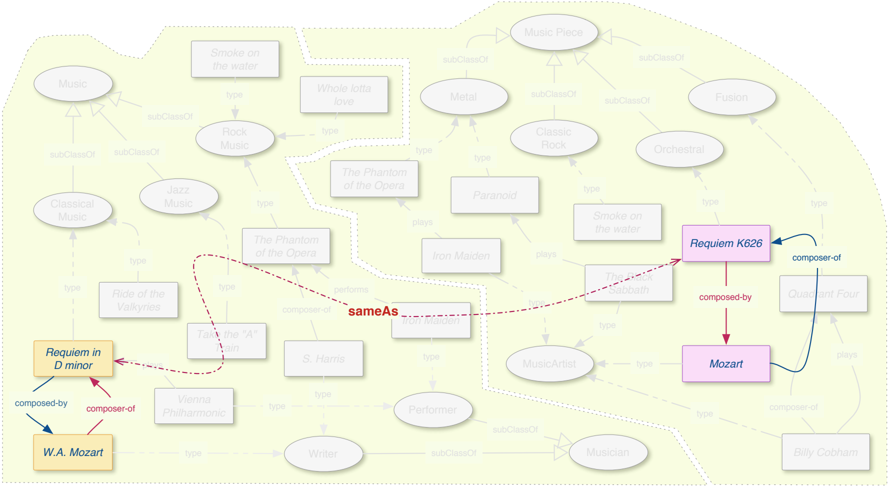
The Algorithm

Incremental Logical Inference - Example (3)



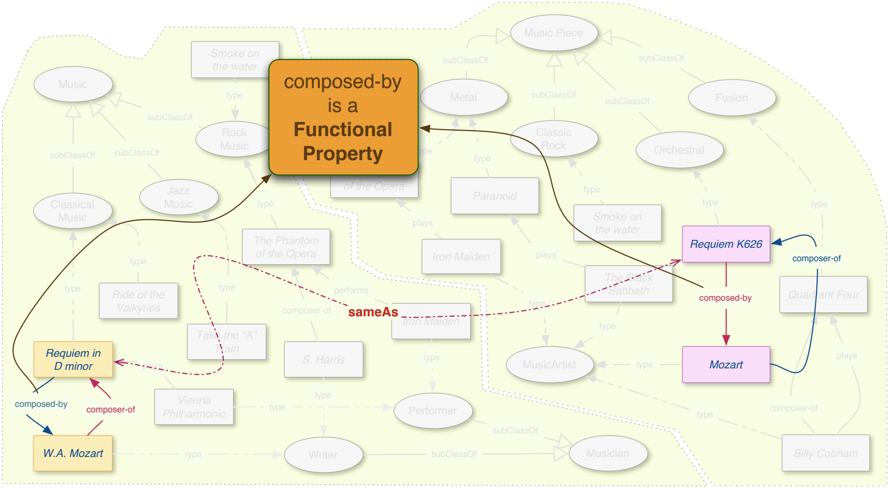
The Algorithm

Incremental Logical Inference - Example (4)



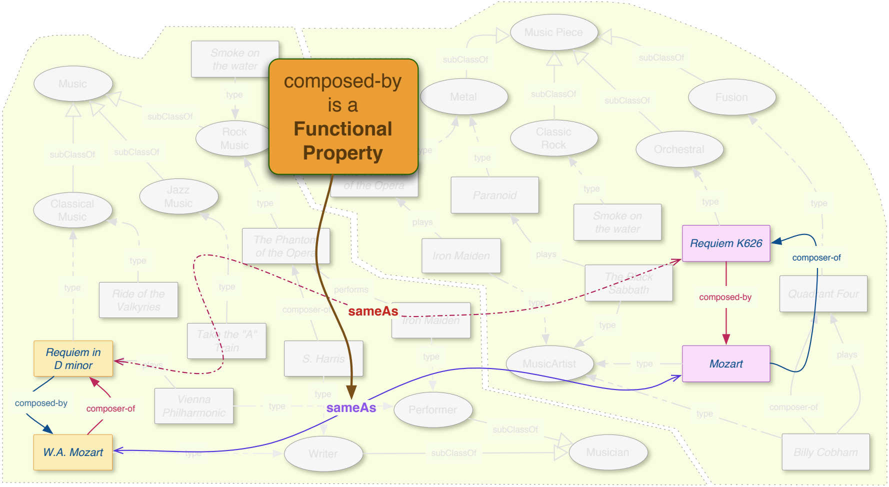
The Algorithm

Incremental Logical Inference - Example (5)



The Algorithm

Incremental Logical Inference - Example (6)



The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

The Algorithm

Update similarity score

- This is the key point of the algorithm
  - A value f - the *influence factor* of the inference - is computed

The "f" factor

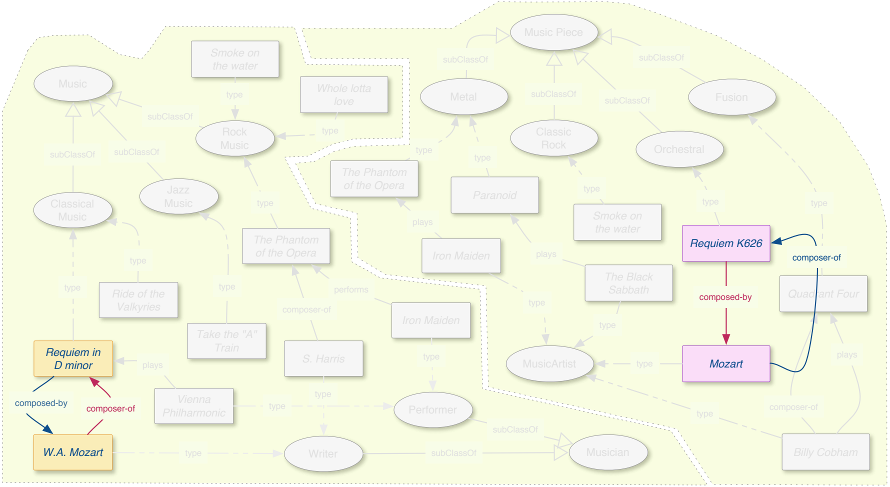
$$f = \prod_{(e_1, e_2) \in Q} \frac{sim(e_1, e_2)}{1 - sim(e_1, e_2)}$$

Q: the set of entity pairs that became equivalent as a consequence of inference

- f is used to **update the similarity** score for the current couple
  - $sim_{inf}(c, c') = min(f \cdot sim(c, c'), 1)$

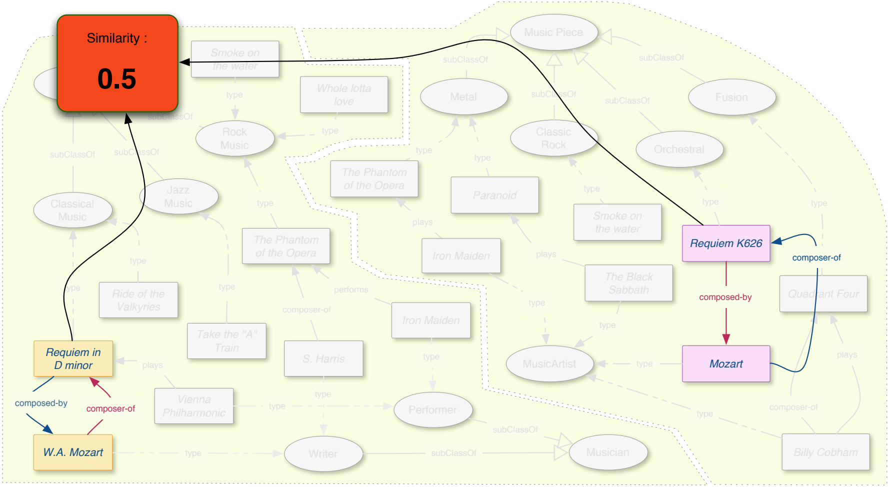
The Algorithm

Update similarity score - Example (1)



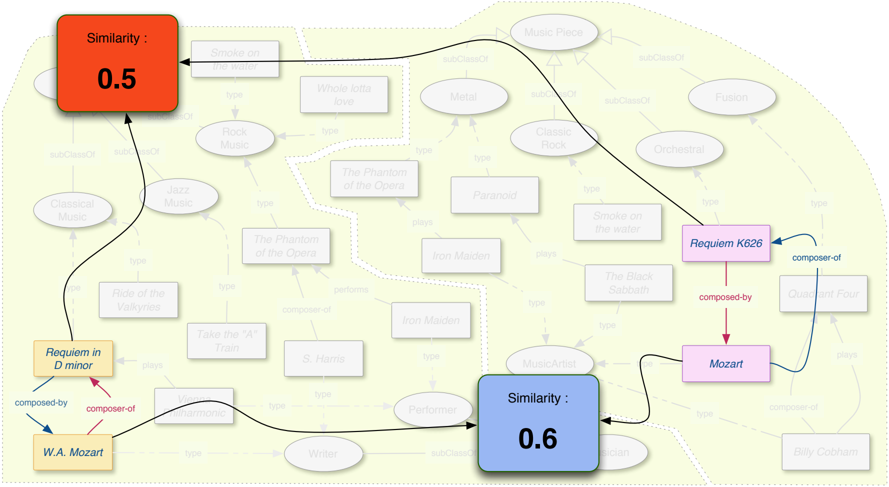
The Algorithm

Update similarity score - Example (2)



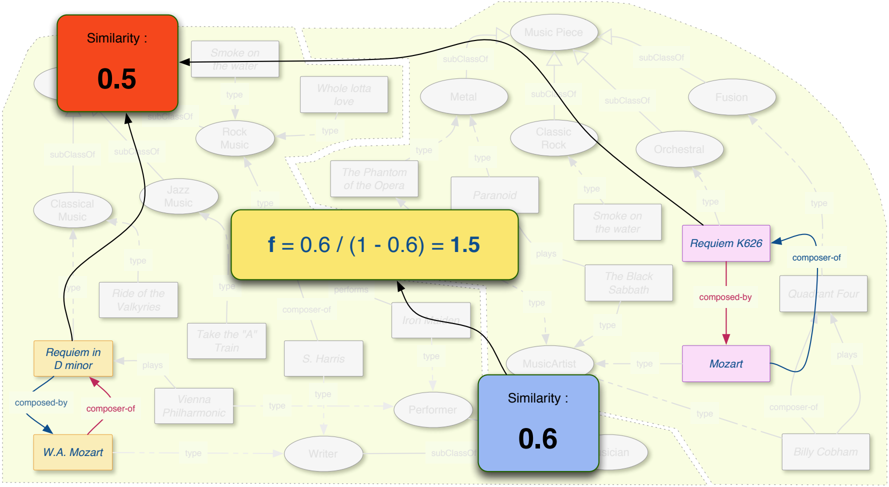
The Algorithm

Update similarity score - Example (3)



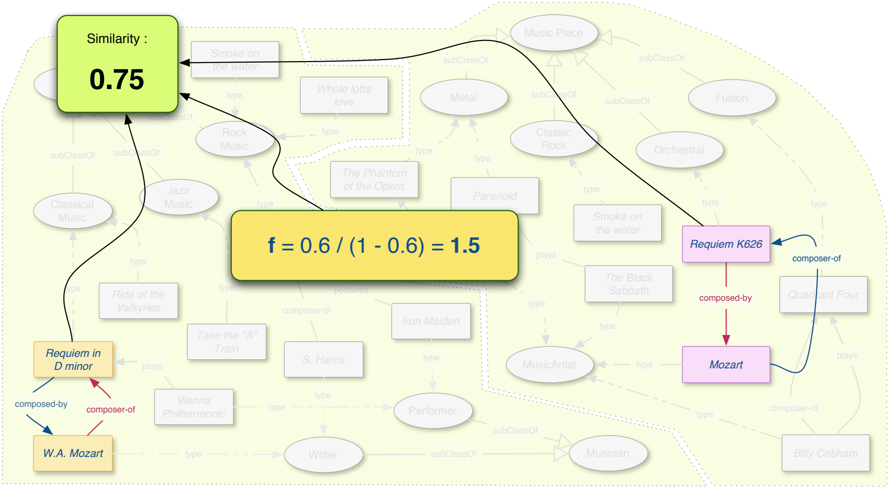
The Algorithm

Update similarity score - Example (4)



The Algorithm

Update similarity score - Example (5)



The Algorithm

Step by step

```
Initialize algorithm's structures (O is O_1 ∪ O_2)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type do
    Determine a candidate relationship a_(c,c')
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A*
until there are clusters with similarity > λ_t
return A*
```

The Algorithm

Building the alignment

- By now for each candidate pair of clusters of a given type
  - A possible relationship has been explored
  - A set of consequences has been inferred
  - An “inference-weighted“ similarity has been computed
- Before restarting the loop:
  - The axiom $a^*_{(c,c')}$ with the highest similarity score is chosen
  - It is added to the output alignment A^*
  - If $a^*_{(c,c')}$ is an equivalence c and c' are merged

The Algorithm

Step by step

Initialize algorithm's structures (O is $O_1 \cup O_2$)
repeat:
  Compute similarity scores between clusters
  Heuristically select a type of clusters
  for each couple (c, c') of that type **do**
    Determine a candidate relationship $a_{(c,c')}$
    Perform incremental inference
    Update similarity score
  Select the best couple, update O and A^*
until there are clusters with similarity $> \lambda_t$
return A^*

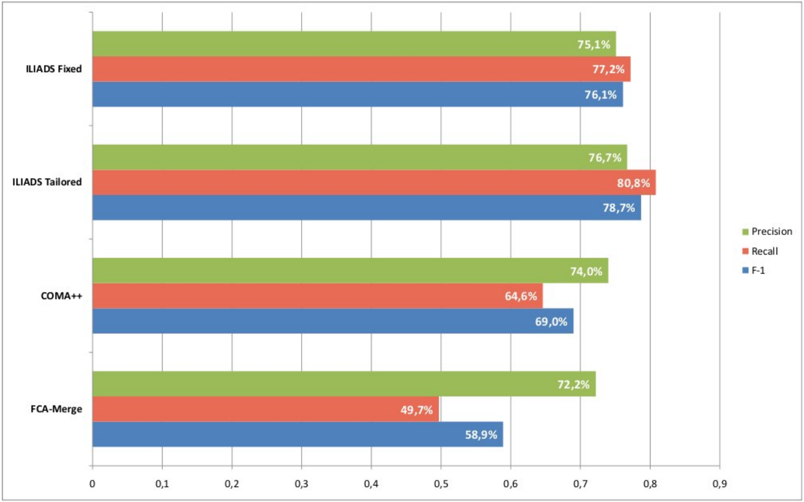
The Algorithm

The End(ing)

- The algorithm halts when there are *no more candidate clusters*
  - When there are no clusters having similarity greater than the threshold λ_t
  - Remaining clusters are not likely to share any relationship
- Intuitively ILIADS is **guaranteed to terminate** because
  - Previously used cluster pairs are not re-used unless their score changes
  - The merging process decreases the number of clusters

Comparative results

Precision, Recall and F-1



Tests Analysis

- The interplay between structure and instance integration delivers higher quality
  - Significant **improvement of recall**
  - Tests on ontologies without instance data showed comparable results with the other systems
- **Lambda-tuning** allows ILIADS to adapt itself better to particular pairs of ontologies
- Inconsistent alignments due to limited inference steps were found only in the .5% of tests

Summary

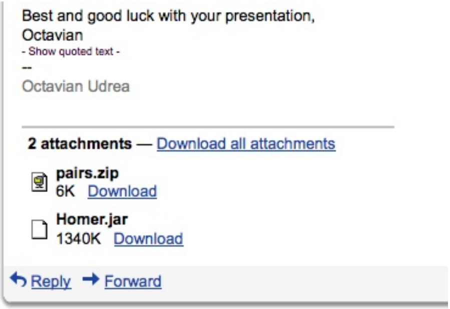
- **Explicit semantics** provided by ontologies can improve the quality of data integration
- **Instance data** exploitation could significantly enhance traditional matching techniques
- **ILIADS** leverages both these opportunities and shows promising results

Future developments

- Intra-ontology differentiated λ parameters
- Alignment of ”distant” sections of the ontologies in parallel

Demo

And now an **ultra-fancy demo**



Thank you.

Group 15

A. Sorbini E. Savioli A. Reale

For Further Reading I

W3C OWL resources.
http://www.w3.org/2004/OWL/.

D. Aumueller, H.H. Do, S. Massmann, and E. Rahm.
Schema and ontology matching with COMA++.
In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908. ACM New York, NY, USA, 2005.

I. Horrocks, P.F. Patel-Schneider, and F. van Harmelen.
From SHIQ and RDF to OWL: the making of a Web Ontology Language.
Web Semantics: Science, Services and Agents on the World Wide Web, 1(1):7–26, 2003.

For Further Reading II

Y. Kalfoglou and M. Schorlemmer.
Ontology mapping: the state of the art.
The knowledge engineering review, 18(01):1–31, 2003.

E. Rahm and P.A. Bernstein.
A survey of approaches to automatic schema matching.
The VLDB Journal The International Journal on Very Large Data Bases, 10(4):334–350, 2001.

P. Shvaiko and J. Euzenat.
A survey of schema-based matching approaches.
Lecture Notes in Computer Science, 3730:146–171, 2005.

For Further Reading III

G. Stumme and A. Maedche.
FCA-MERGE: Bottom-Up Merging of Ontologies.
In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 17, pages 225–234. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

O. Udrea, L. Getoor, and R.J. Miller.
HOMER: Ontology alignment visualization and analysis.

F. Baader.
The description logic handbook: theory, implementation, and applications.
Cambridge University Press, 2003.