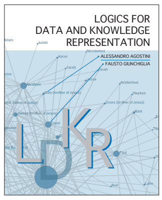
Logics for Data and Knowledge Representation

Alessandro Agostini
agostini@dit.unitn.it

Fausto Giunchiglia
fausto@dit.unitn.it

University of Trento

Semantic Matching



- Matching Problems
- Graphs Matching
  - Syntactic
  - Semantic
- Semantic Matching via SAT in ClassL

Introduction

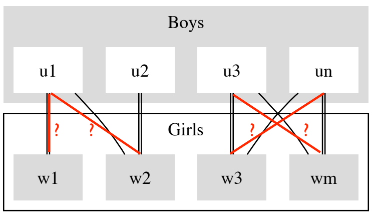
- An important problem on graphs that we will apply to richer representations like concept hierarchies, classifications, schemas and ontologies is "matching."
- This problem is also popular in semantic networks, where one may want to check whether a particular concept is present.

Introduction (cont')

- Some popular situations that can be modeled as a matching problem are:

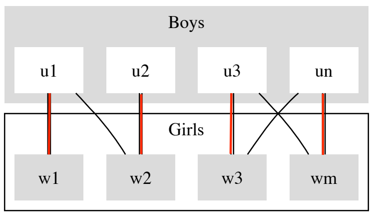
  - Marriage.
  - Concept matching in semantic networks.
  - Schema matching in distributed databases.
  - Ontology matching (ontology "alignment") in the Semantic Web.
  - ...

The Matching Problem (An Example)



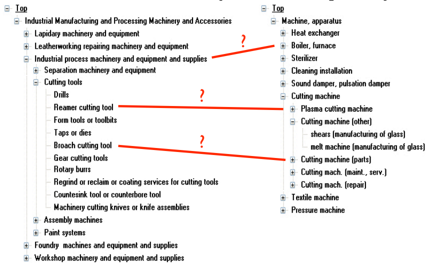
Red lines not a perfect matching (i.e. a 1-1mapping).

The Matching Problem (An Example)



Red lines: a perfect matching (i.e. a 1-1mapping).

The *Semantic* Matching Problem (Example)



Matching Problems

- There are two kinds of matching:
  - Syntactic: matching of nodes as objects or strings (so, as such, without meaning).
  - Semantic: matching of nodes as concepts.
- A Matching Problem (syntactic or semantic) is a problem on graphs summarized as: Given two finite graphs, is there a matching between the (nodes of the) two graphs?

Matching Problems

- A problem of matching can be decomposed in two steps:
  - 1. extract the graphs from the conceptual models under consideration;
  - 2. match the resulting graphs.
- Below we show some examples of step 1. (We follow [Giunchiglia & Shvaiko, 2007].)

Relational DB Schemas

- Let us consider the following relational database (RDB) model, say "BANK":

BRANCH

BN	Street	City	Zip
B8	Piazza Venezia	Trento	38100
B2	Piazza Cordusio	Milano	20123

STAFF

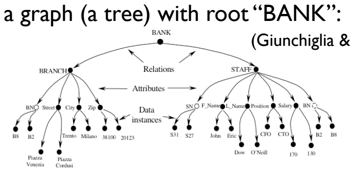
SN	F_Name	L_Name	Position	Salary	BN
S31	John	Dow	CFO	170	B2
S27	Eric	O'Neill	CTO	130	B8

(Giunchiglia & Shvaiko, 2007)

Relational DB Schemas Representation 1

- We can represent the RDB model "BANK" as a graph (a tree) with root "BANK":

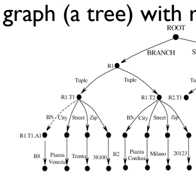
(Giunchiglia & Shvaiko, 2007)



- The RDB model is first partitioned into relations, then attributes and data instances.

Relational DB Schemas Representation 2

- We can represent the RDB model "BANK" as a graph (a tree) with root "BANK":



(Giunchiglia & Shvaiko, 2007)

- The model is partitioned into relations, then into tuples, attributes and data instances.

Relational DB Schemas Remarks

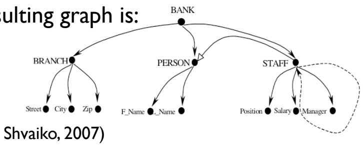
- Which of the two representations is more preferable depends on the concrete task.
- It is always possible to transform one representation into the other.
- In contrast to the example of RDB "BANK", DB schemas are seldom trees.
- More often, DB schemas are translated into Directed Acyclic Graphs (DAG's).

OODB Schemas

- Let us consider the RDB "BANK" in terms of an object-oriented DB (OODB) schema:

BRANCH (Street, City, Zip)
PERSON (F_Name, L_Name)
STAFF : PERSON (Position, Salary, Manager)

- The resulting graph is:



(Giunchiglia & Shvaiko, 2007)

OODB Schemas Remarks

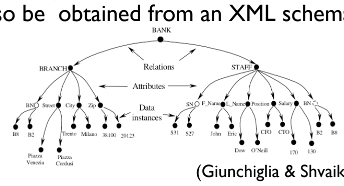
- OODB schemas capture more semantics than the relational DBs. In particular, an OODB schema:
  - explicitly expresses subsumption relations between elements;
  - admits special types of arcs for part/whole relationships in terms of aggregation and composition.

Semi-structured Data

- Neither RDBs nor OODBs capture all the features of semi-structured or unstructured data (Buneman, 1997):
  - semi-structured data do not possess a regular structure (schemaless);
  - the "structure" of semi-structured data could be partial or even implicit.
- Typical examples are: HTML and XML.

XML Schemas

- XML schemas can be represented as DAGs.
- The graph from the RDB "BANK" could also be obtained from an XML schema.



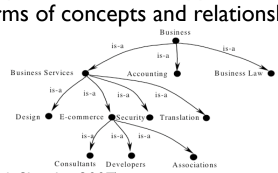
(Giunchiglia & Shvaiko, 2007)

XML Schemas Remarks

- Often XML schemas represent hierarchical data models.
  - In this case the only relationships between the elements are {is-a}.
- Attributes in XML are used to represent extra information about data. There are no strict rules telling us when data should be represented as elements, or as attributes.

Concept Hierarchies

- A concept hierarchy is a semi-formal conceptualization of an application domain in terms of concepts and relationships.



(Giunchiglia & Shvaiko, 2007)

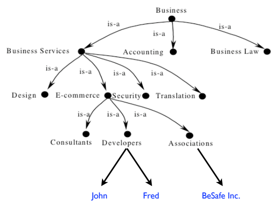
Concept Hierarchies Remarks

- Examples are classification hierarchies, e.g., Yahoo! and Google directories (catalogs).
- Classification hierarchies / Web directories are sometimes referred to as *lightweight ontologies* (Uschold & Gruninger, 2004). However:
  - they are *not* ontologies, as they lack of a formal semantics (semi-formal vs formal.)
  - they don't formalize class instances.

Lightweight Ontologies

- Definition. A lightweight ontology is a formal conceptualization of an application domain in terms of concepts and {is-a, instance-of} relationships.
- Remark: A lightweight ontology is thus a concept hierarchy with:
  - a formal semantics (semantic level)
  - {instance-of} relationships (syntactic level)
  - without relationships except {is-a} (")

Lightweight Ontologies Example



- Data instances of the concept (class) "Developers" are: John, Steve.
- Data instances "Associations" are: BeSafe Inc.
- We define a class instances the data instance of a class.

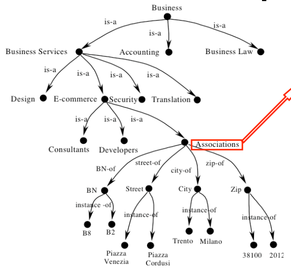
Lightweight Ontologies and Class Logic

- The logic of classes (ClassL) provides a formal language (syntax + semantics) to model lightweight ontologies, where:
  - concepts are modeled by propositions;
  - {is-a, instance-of} relationships are modeled, respectively, by subsumption ($\sqsubseteq$) and class-propositions (i.e., wffs like P(*a*)).
- ClassL ontologies $=_{df}$ lightweight ontologies.

Ontologies

- An ontology is a formal conceptualization of an application domain in terms of concepts, attributes, and relationships.
  - Relations can be defined by the user.
  - Pre-defined relationships with known semantics are: {is-a, part-of, instance-of}.
- An ontology is a lightweight ontology with attributes and a wider set of relationships.

Ontologies Example



- Attributes of the concept "Associations" are: BN, City, Street, Zip.
- Data instances are: B8 and B2.
- Data instances have fixed attributes values, e.g.: City="Trento"

Matching of Graphs

- A matching of graphs is the process of compute a computable function *Match* (matcher), which takes two graphs G_1,G_2 and returns a set of mappings (or "mapping elements") of the form (n_1, n_2, R), where:
  - n_1 is a node of G_1 and n_2 is a node of G_2,
  - R is a binary relation R(n_1,n_2).

Matching of Graphs: Syntactic vs Semantic

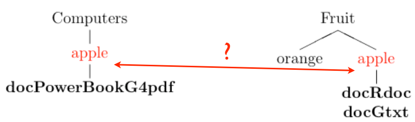
- There are two kinds of graph matching:
  - Syntactic: matching of nodes as objects or strings (so, as such, without meaning):
    - R(n_1,n_2) is a syntactic similarity metric;
    - we are *not* interested in this matching.
  - Semantic: matching of nodes as concepts.
    - R(n_1,n_2) is a semantic similarity metric.

Syntactic vs Semantic Matching

- A key difference (Giunchiglia & Shvaiko, 2003):
  - syntactic matching: in matching two nodes, the meaning that we (implicitly) attach to them depends only on their labels, independently of their position.
  - semantic matching: the nodes' position matters, i.e. the meaning that we attach to them depends on the the position.

Syntactic vs Semantic Matching (Example)

- A syntactic matcher would answer (apple, apple, =), while a semantic matcher would answer (apple, apple, ≠)



(Agostini & Pan, 2007)

Matchers as a Decision Function

- A matcher as a decision function is a (computable) function

  Match(G_1,G_2, n_1,n_2, R)

  which takes in input:
  - two graphs G_1,G_2,
  - two nodes n_1,n_2 with n_1 in G_1 and n_2 in G_2
  - a binary relation R = R(n_1,n_2)

  and returns a Yes/No answer.

Remark on Matchers

- Most matchers in the literature are defined by the function

  Match(G_1, G_2)

  which takes two graphs and returns a set of mapping elements defined by (n_1, n_2, R).
- Exercise: define a matcher *Match*(G_1, G_2) by using a matcher *Match*(G_1,G_2, n_1,n_2, R).

Solution of the Exercise

- Exercise: define a matcher *Match*(G_1, G_2) by using a matcher *Match*(G_1,G_2, n_1,n_2, R).
- Solution (naive): triple loop on the nodes of the graphs G_1, G_2 and on the set of proposed relations and, at each loop, call *Match*(G_1,G_2, n_1,n_2, R).

Semantic Matching Relations (1)

- Given graphs G_1,G_2, we define the following semantic relations $R(n_1,n_2)$ between a node n_1 in G_1 and a node n_2 in G_2:

  $n_1 \supseteq n_2$, $n_1 \subseteq n_2$ (n_1 is more general/specific than n_2)

  $n_1 \doteq n_2$ (n_1 is equivalent to n_2)

  $n_1 \perp n_2$ (n_1 and n_2 mismatch)
- We represent n_1 and n_2 by a proposition, say P and Q, and then use class-logic' semantics.

Semantic Matching Relations (2)

- For all class-valuations σ, we define $R(n_1,n_2)$:

  $n_1 \supseteq n_2$ if $\sigma(P) \supseteq \sigma(Q)$;

  $n_1 \subseteq n_2$ if $\sigma(P) \subseteq \sigma(Q)$;

  $n_1 \doteq n_2$ if $\sigma(P) = \sigma(Q)$;

  $n_1 \perp n_2$ if $\sigma(P) \cap \sigma(Q) = \varnothing$.
- Notation: Be aware of the misleading use of symbols $\supseteq$, $\subseteq$, and $\perp$ on the left-hand side.

Semantic Relations and SAT Problem in ClassL

- A semantic relation $R(n_1,n_2)$ can be checked by class logic's reasoning as a SAT problem:

  $n_1 \supseteq n_2$ iff $\models P \sqsupseteq Q$; ($\sqsupseteq$ defined symbol)

  $n_1 \subseteq n_2$ iff $\models P \sqsubseteq Q$;

  $n_1 \doteq n_2$ iff $\models P \sqsubseteq Q$ and $\models Q \sqsubseteq P$;

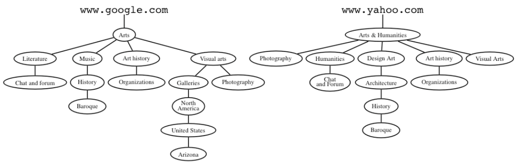
  $n_1 \perp n_2$ iff $\models P \sqcap Q \sqsubseteq \perp$,

  where P and Q represent n_1 and n_2 in classL.

Semantic Matching via SAT in ClassL

- The SAT problem to solve to compute *Match*(G_1,G_2, n_1,n_2, R) is built in three steps:
  - First, select the portion T of knowledge (*background* theory) relevant to transform nodes n_1,n_2 into two propositions P, Q.
  - Second, select a semantic relation $R(n_1,n_2)$ and rewrite it as a SAT problem $\models r(P, Q)$.
  - Third, run a SAT solver on $T \models r(P, Q)$.

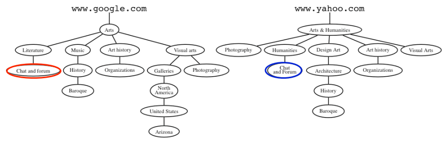
Example: Concept Hierchies



(Serafini *et. al.*, 2003)

Example (cont')

- Suppose we want to discover the relation R between *Chat and Forum* in the Google directory (left) and *Chat and Forum* in the Yahoo directory (right):



Example (cont')

- Step 1 : transformation of nodes n_1 = *Chat and Forum* and n_2 = *Chat and Forum* to propositions, P and Q; selection of the portion T of knowledge (*background* theory) relevant to the application of a SAT solver.
  - WordNet is used at this step to build T.
- Step 2 : select the relation R between n_1 and n_2 among the semantic relations of interest.

Example (cont')

- Step 3 : The question "Is *Chat and Forum* less general than *Chat and Forum*?" becomes the SAT problem "Is T |= $P \sqsubseteq Q$?" where:

P = (art#1 $\sqcap$ literature#2 $\sqcap$ (chat#1 $\sqcup$ forum#1)),
Q = (art#1 $\sqcup$ humanities#1) $\sqcap$ humanities#1 $\sqcap$ (chat#1 $\sqcup$ forum#1),
T = {art#1 $\sqsubseteq$ humanities#1, humanities#1 $\sqsupseteq$ literature#2}

References

(Preprints available at http://dit.unitn.it/~ldkr/#Biblio)

- F. Giunchiglia, P. Shvaiko, "Semantic matching." *Knowledge Engineering Review*, 18(3):265-280, 2003.
- L. Serafini, P. Bouquet, B. Magnini, S. Zanobini, "An Algorithm for Matching Contextualized Schemas via SAT." IRST Technical Report 0301-06, ITC, January 2003.
- F. Giunchiglia, M. Marchese, I. Zaihrayeu. "Encoding Classifications into Lightweight Ontologies." *J. of Data Semantics VIII*, Springer-Verlag LNCS 4380, pp 57-81, 2007.