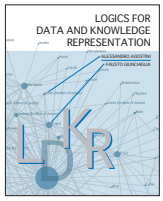
Logics for Data and Knowledge Representation

Alessandro Agostini
agostini@dit.unitn.it

Fausto Giunchiglia
fausto@dit.unitn.it

University of Trento

Formal Classification



- Classification
- Formal Classification (FC):
  - Labels
  - Links
- Query-answering on FCs.

Classification Hierarchies (1)

- Classifications hierarchies are easy to use... ... for humans.
- Classifications hierarchies are pervasive (Google, Yahoo, Amazon, our PC directories, email folders, address book, etc.).
- Classifications hierarchies are largely used in industry (Google, Yahoo, eBay, Amazon, BBC, CNN, libraries, etc.).

Classification Hierarchies (2)

- Classification hierarchies have been studied for very long (e.g., Dewey Decimal Classification system -- DCC, Library of Congress Classification system –LCC, etc.).
- Classifications hierarchies are lightweight (no roles, trees or simple dags, …).
- Classification hierarchies are a kind of concept hierarchies.

In Classification Hierarchies

- Labels are natural language sentences; useful but hard to deal with in an automated way.
- Links are of the kind "child-of" (e.g. "economy child-of Europe"), where in an ontology you would have, (instance-of}, or roles, or {is-a} links.
- Main Problem: No clear semantics for both labels at nodes and links.

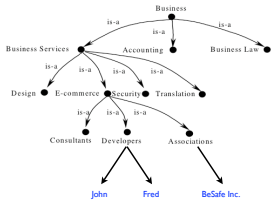
Lightweight Ontologies and Classification

- Classification hierarchies are semi-formal, so we need (lightweight) ontologies to automatically reason about classification.
- A theory of formal classification is needed for building the bridge from classifications to simple, i.e., "lightweight" ontologies.
- One result: Onto2Class, Class2Onto operators).

Lightweight Ontologies

- We know that a lightweight ontology is a formal conceptualization of a domain in terms of concepts and {is-a, instance-of} relationships.
- Lightweight ontologies (LOs) add a formal semantics and {instance-of} relationships to classification hierarchies.
- In short: LOs make classifications formal!

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

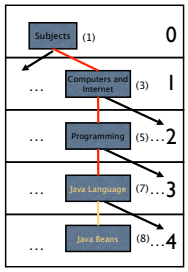
- The logic of classes (ClassL) provides a formal language (syntax + semantics) to model lightweight ontologies, where:
  - concepts are modeled by propositions;
  - {is-a, instance-of} relationships are modeled, respectively, by subsumption ($\sqsubseteq$) and class-propositions (i.e., wffs like $P(a)$).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

Formal Classifications

- A formal classification (FC) is a rooted tree, where each node is assigned a label represented by a proposition in class logic.
- A formal classification provides for
  - formalizing the meaning of labels, and
  - formalizing the meaning of links
- Both formalizations come from class logic!

Label Semantics

- Natural language words are often ambiguous. E.g. Java (an island, a beverage, an OO programming language)
- When used with other words in a label, wrong senses can be pruned. E.g., "Java Language" – only the 3rd sense of Java is preserved.



From NL Labels to Labels in Class Logic

- Several approaches to rewrite a natural language label into a class logic proposition.
- Following (Giunchiglia et al., 2007), we may distinguish four steps:

1. tokenization (get distinct words);
2. words stemming (get to a basic form);
3. rewrite each word into its proposition;
4. prune inconsistent senses.

From NL Labels to Labels in Class Logic

- Example1: "Java" becomes the proposition
  Java#1 ⊔ Java#2 ⊔ Java#3

  where Java#i is a propositional variable representing the *ith*-sense of the word "Java" according to a dictionary (e.g., WordNet).
- Example 2: "Java Beans" becomes:
  (Java#1 ⊔ Java#2 ⊔ Java#3)⊓(Bean#1 ⊔ Bean#2)

Advantages of Propositions (1)

- NL labels're ambiguous, propositions aren't!
- Extensional semantics of propositions naturally maps nodes to real world objects.
- Labels as propositions allow us to deal with the standard problems in classification (e.g., document classification, query-answering, and matching) by means of class logic's reasoning, mainly the SAT problem.

Advantages of Propositions (2)

- Observe that NL labels are always transformed into propositions in CNF/DNF.
- An alphabetically ordered CNF proposition provides a unique name for a concept / node represented by a set of NL labels.
- An alphabetically ordered DNF proposition provides a unique name for all meanings of a concept represented by a set of NL labels.

Advantages of Propositions (2, cont')

- Example 1: "Java Beans" or "Java and Beans" or "Beans and Java" gain a unique name by using the alphabetically ordered CNF wff:
  (Bean#1 ⊔ Bean#2)⊓(Java#1 ⊔ Java#2 ⊔Java#3)
- Example 2: All meanings of "Java Beans" get a unique name by using an alphabetically ordered DNF wff: (Bean#1⊓Java#1) ⊔ (Bean#1⊓Java#2)⊔(Bean#1⊓Java#3)⊔(Bean#2⊓Java#1) (Bean#2⊓Java#2)⊔(Bean#2⊓Java#3).

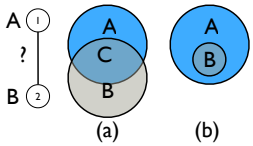
Formalizing the Meaning of Links

- Child nodes in a classification are always considered in the context of their parent nodes.
- Child nodes therefore specialize the meaning of the parent nodes.
- Contextuality property of classifications.

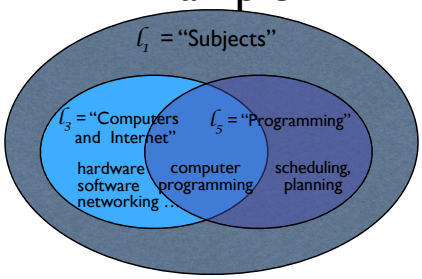
Formalizing the Meaning of Links

(a) General intersection relationship: can be used to represent facets.
The meaning of node 2 is C = A ⊓ B.

(b) Subsumption relationship: child nodes are specific case of the parent nodes.
The meaning of node 2 is B.



General Intersection Example



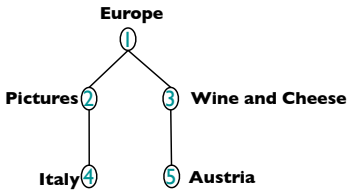
Concept at a Node

- Parental contextuality is formalized in class logic by the notion of "concept at a node."
- A concept C_r at the root node r is the class proposition (label) used to denote the node.
- A concept C_i at a node n_i is the conjunction of a proposition P_i (label of n_i) and the concept C_j at node n_j parent to n_i.
  In classL: $P_i \sqcap C_j$.

Concept at a Node

- A concept at a node n_i can be computed as the conjunction of all the labels from the root of the classification hierarchy to n_i.
- Concepts at nodes capture the classification semantics by using the meaning of labels (propositions defined by using WordNet and a linguistic analysis) and the nodes' position.

Concept at a Node Example



In class logic: $C_4 = C_{europe} \sqcap C_{pictures} \sqcap C_{italy}$

Normalized Formal Classifications

- A normalized formal classification (NFC) is a rooted tree, where each node is assigned the concept at a node of the corresponding formal classification.
- A NFC is a taxonomy in the sense that each label of a child node (a proposition) is subsumed by the labels (propositions) of the parent nodes.

Document Classification

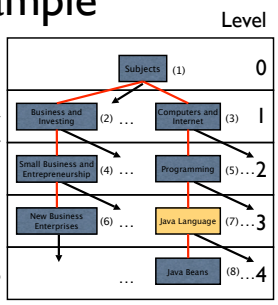
- Each document d in a classification is assigned a proposition C^d in class logic. C^d is called document concept.
- C^d is build from d in two steps:
  - keywords are retrieved from d by using standard text mining techniques.
  - keywords are converted into propositions by using methodology discussed above.

Document Classification "Get specific" Rule

- For any given document d and its concept C^d we classify d in each node n_i such that:
- $\models C_i \sqsupseteq C^d$ (i.e. the concept at node n_i is more general than C^d); and
- there is no node n_j ($j \neq i$), whose concept at node C_j is more specific than C_i and more general than C^d: $\models C_j \sqsubseteq C_i$ and $\models C_j \sqsupseteq C^d$.

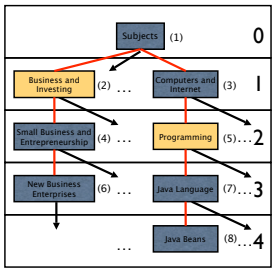
Example

- Suppose we need to classify "Professional Java, JDK-5th Edition" by W. Clay Richardson *et al.*
- The document concept of such document d is: C^d = Java#3⊓ Programming#2.
- The node 7 is the only node which conforms to the "get specific" rule.



Example (cont')

- Suppose we need to classify "Visual Basic.Net Programming for Business" by Philip A. Koneman.
- The document concept of such document d is: C^d = VisualBasicNet#1⊓ Programming#2⊓ Business#1
- The nodes 2,5 conform to the "get specific" rule.



Query-Answering

- Query-answering on a classification hierarchy of documents based on a query q as a set of keywords is defined in two steps:
  1. the proposition C^q is build from q by converting q's keywords as said above.
  2. The set of answers (retrieval set) to q is defined as a SAT problem in class logic:
     $A^q =_{df} \{d \text{ document} \mid \models C^d \sqsubseteq C^q\}$.

Query-Answering A Problem

- Searching on *all* the documents may be expensive (millions of documents classified).
- We define a set of *nodes* which contain only answers to a query q as follows:
  $N_s^q =_{df} \{n_i \text{ node} \mid \models C_i \sqsubseteq C^q\}$
- Remark: Each document d in n_i in N_s^q *is* an answer to the query q, since $\models C^d \sqsubseteq C_i$ by definition of classification. Thus $N_s^q \subseteq A^q$.

Query-Answering Classification Set

- We extend N_s^q (called sound classification answer) by adding a set of nodes (called query classification set) defined as:

$Cl^q =_{df} \{n_i \text{ node} \mid d \in n_i \text{ and } \models C^d \equiv C^q\}$

(i.e., the nodes which constitute the classification set of a document d, whose concept C^d is equivalent to C^q).

Query-Answering Sound Answer Set

- The set of answers (retrieval set) to q is finally defined as the following set:

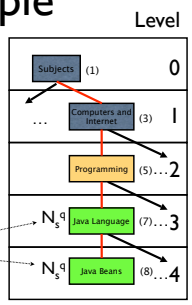
  $A_s^q =_{df} \{d \in n_i \mid n_i \in N_s^q\} \cup$

  $\{d \in n_i \mid n_i \in Cl^q \text{ and } \models C^d \sqsubseteq C^q\}$.

- Under this definition, an answer to a query are documents from nodes whose concepts are more specific than the query concept.

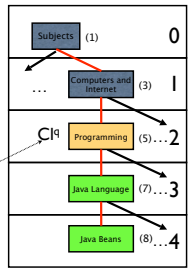
Example

- Suppose that a user makes a query q, which is converted into C^q = Java#3⊓Cobol#1, where Cobol#1 is "common business-oriented language."
- It can be shown that N_s^q = {7,8}.
- Exercise: show it.



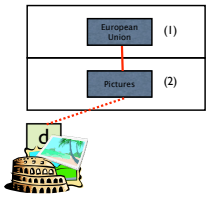
Example (cont')

- It can be shown that N_s^q = {7,8}.
- "Java for COBOL programmers, 2nd ed." is classified in node 2, so it is *not* an aswer by using only N_s^q = {7,8}.
- We then consider Cl^q to compute more answers, among others are the documents in node 5.



Sound Answer Set Remark

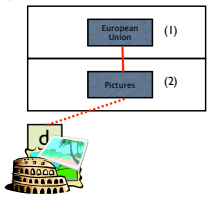
- The set A_s^q is sound (i.e., contains answers to q), but not complete (i.e., does not contain all the answers to q).
- See the next example.



Sound Answer Set Example

- Suppose that a user makes a query q like "video or pictures of Italy," which is converted into C^q = Italy#1⊓(Video#2⊔Pictures#1).
- C^q is equivalent to:

  C^q_1 = Video#2⊓Italy#1,
  C^q_2 = Pictures#1⊓Italy#1.



Sound Answer Set Example (cont')

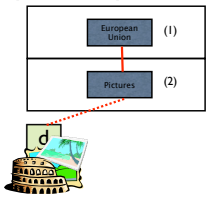
- But not $\models C_2 \sqsubseteq C^q_1$, hence a document d in 2 about Rome, with C^d = Pictures#1⊓Rome#1

  is not retrieved, since:

  $N_s^q = \{n_i \mid \models C_i \sqsubseteq C^q\} = \varnothing$

  and $Cl^q = \{1\}$, so $d \notin A_s^q$.

  (A_s^q is not complete)



Final Remarks

- The edge structure of a NFC is *not* considered neither for document classification nor for query answering.
  - The edges information becomes redundant, as it is *implicitly encoded* in the "concept at a node" notion.
- There are more than one way to build a NFC from a set of concepts at nodes.

References & Credits

- **References:**
- F. Giunchiglia, M. Marchese, I. Zaihrayeu. "Encoding Classifications into Lightweight Ontologies." *J. of Data Semantics VIII*, Springer-Verlag LNCS 4380, pp 57-81, 2007.

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

- **Thanks to:**
  - Ilya Zaihrayeu, who wrote the slides that we have used here and modified for the purposes of this lecture.