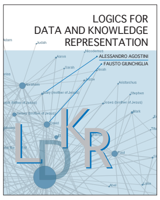
Logics for Data and Knowledge Representation

Alessandro Agostini
agostini@dit.unitn.it

Fausto Giunchiglia
fausto@dit.unitn.it

University of Trento

Outline



- Course Information
- Data
- Knowledge
- Representation
- Logics

Prof. Fausto Giunchiglia

Course Information

1. Staff
2. Scheduling
3. Reception times
4. Course website
5. Objective and Outcomes
6. Prerequisites
7. Contents
8. Lectures
9. Handouts & Slides
10. Readings
11. Other resources
12. Exam policy & Grading

Running Example

- What **data** are retrieved from such situation?
- What **knowledge** we gain from it?

Data
(Definition)

- Definition (*Webster*). A collection of facts from which conclusions may be drawn.
  - Useful irrelevant or redundant facts, which must be processed to be meaningful.
  - Used as a basis for reasoning, discussion or calculation (*Merriam-Webster*).

Data
(Example, cont')

- **Data** is the (physical) image, the set of pixels.
- But what about: the level of granularity?
- The quality of the camera? How much exposure?

Data
(IT view)

- *Factual output* produced by a sensing device.
- **Examples**:
  a unique address and a pointed single item, a sensor data, a number in real memory, a pixel in an image, a web page, an image, a move, a sound, ...

Knowledge
(Definition)

- Definition (*Webster*). The body of truths or facts (i.e., data) accumulated by human kind. The sum of what is known.
  - Thus, knowledge is data in context, or organized data, or also data in relationship.
  - **Remark**: "body of ... facts / data" in the definition indirectly refers to some form of organization (of facts / data).

Knowledge
(Example, cont')

- **Knowledge** is, e.g.:
  - a pixel is red
  - a set of pixels is a person
  - there are three persons
  - ...

Knowledge
(IT view)

- A "statement" that a class is in relation with other classes.
- **Examples**:
  - An ontology, a classification
  - a knowledge base, a set of "statements"
  - a database relation.

Data vs. Knowledge
Example

- Data = bare facts - A telephone number.
- Knowledge is (organized) data that facilitates action - A phone book.
  - E.g., *recognizing a phone number belongs to a good client* (who needs to be called once per week to get his orders).
    - connecting a phone number to a client.

Representation
(Definition)

- Definition (*Webster*). The expression or designation by some term, character, symbol, or the like.
  - The type of representors we are most concerned with are "symbols."
  - **Example**:
    the digit '3' stands for the number 3, as does the group of letters 'III', `three', `tre',..

Representation

- Of special concern to us is when a group of symbols stands for a sentence (proposition).
  - Example: 'John loves Mary' stands for the proposition that John loves Mary.
- **Remark**: The use of symbols to represent expressions like sentences, predicates, data and concepts characterizes *logical* modeling.
  - logical modeling motivates the *use* of logic!

Representation of Data

- The expression or designation of data by some term, character, symbol, or the like.
- **Example**: a pixel of the photo can be represented by "p," (term), "red(3,7)" (the Cartesian coordinates of the red-pixel w.r.t. the photo's dimension), etc.
- A database is a representation of data and their relationships (see the next example).

Data Representation (Example, cont')

These databases represent data.

	red(x1,y1)	
	red(x2,y2)	
	white(x3,y3)	
	orange(x4,y4)	
	...	

Denotation of a pixel with color orange and position as determined by Cartesian coordinates (x4,y4)

	Benedetta
	Eleonora
	Valentina

Denotation of a set of pixels.

Note the different levels of data represented by the two tables.

Representation of Knowledge

- The expression or designation of knowledge by some term, character, symbol, or the like.
- **Example**: a set of related pixels of the photo can be represented by terms like "Valentina," "mountain," "background," "snow," near(Benedetta,Eleonora)," etc.
- A knowledge-base is a representation of (a body of) knowledge (see the next example).

Knowl. Representation (Example, cont')

These "statements", *together in a set*, define a knowledge-base.

- Benedetta, Valentina and Eleonora are girls.
- Benedetta and Eleonora are near each other.
- There is snow in the background.
- The skiboots of Valentina are orange.

Knowl. Representation (Example, cont')

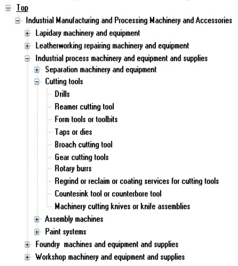
These database relations represent knowledge.

Girl	Valentina	...
	red(x1,y1)	
	red(x2,y2)	
	white(x3,y3)	
	orange(x4,y4)	
	...	

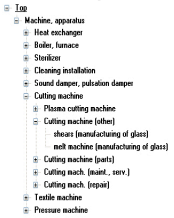
Girl	Name	pixelCoord
	Eleonora	(x1,y1)
	Eleonora	(x2,y2)
	Benedetta	(x3,y3)
	Valentina	(x4,y4)
	...	...

Background	Snow	...
	white(x1,y1)	
	white(x2,y2)	
	white(x3,y3)	
	white(x4,y4)	
	...	

Knowl. Representation (Example, cont')



- Top
  - Industrial Manufacturing and Processing Machinery and Accessories
    - Lapidary machinery and equipment
    - Leatherworking repairing machinery and equipment
    - Industrial process machinery and equipment and supplies
      - Separation machinery and equipment
      - Cutting tools
        - Drills
        - Reamer cutting tool
        - Form tools or toolbits
        - Taps or dies
        - Broach cutting tool
        - Gear cutting tools
        - Rotary burrs
        - Regrind or reclaim or coating services for cutting tools
        - Countesink tool or counterbore tool
        - Machinery cutting knives or knife assemblies
      - Assembly machines
      - Paint systems
    - Foundry machines and equipment and supplies
    - Workshop machinery and equipment and supplies



- Top
  - Machine, apparatus
    - Heat exchanger
    - Boiler, furnace
    - Sterilizer
    - Cleaning installation
    - Sound damper, pulsation damper
    - Cutting machine
      - Plasma cutting machine
      - Cutting machine (other)
        - shears (manufacturing of glass)
        - melt machine (manufacturing of glass)
      - Cutting machine (parts)
      - Cutting mach. (maint., serv.)
      - Cutting mach. (repair)
    - Textile machine
    - Pressure machine

Languages for Representation

Introduction

- Languages for representation we have seen so far are:
  - English, with expressions as "the skiboots of Valentina are orange";
  - 'Symbolized' English, with expressions as "orange(Valentina,skiboots)";
  - Tables + 'symbolized' English, with expressions as " "

Girl	Valentina	...
	orange(x4,y4)	

Language's Components

- Definition. A language for representation is a set of expressions build from a set of terms, characters, symbols, and the like.
- A language for representation is defined accordig to its syntax and its semantics:
  - Syntax: the way the language is written.
  - Semantics: the way the language is interpreted (i.e., what expressions *mean*).

Language's Components Syntax and Semantics

- Syntax: the way a language is written.
  - Syntax is determined by a set of "rules" saying how to construct the expressions of the language from the set of atomic tokens (i.e., terms, characters, symbols).
  - The set of atomic token is called alphabet of symbols, or simply the alphabet).

Syntax Example

- Suppose we want to represent the fact that Benedetta and Eleonora are near each other.
  - By using English we may write (syntax): Benedetta is near to Eleonora.
  - By using a 'symbolized' English we may write (syntax): near(B,E), or extensively near(Benedetta,Eleonora)

Language's Components Syntax and Semantics

- Semantics: the way a language is interpreted.
  - determines the meaning of syntactic constructs (expressions), that is, the relationship between syntactic constructs and the elements of some universe of meanings.
  - such relationship is called interpretation.

Semantics Example

- Suppose we want to represent the fact that Benedetta and Eleonora are near each other.
  - For suppose we write (syntax): near(B,E).
- To fix the semantics of "near(B,E)" we need to fix an interpretation I of it, i.e.,
  "near" by I means near (spatial relation)
  "B" by I means Benedetta (a girl)
  "E" by I means Eleonora (a girl)

Formal Semantics (Definition)

- Formal Semantics: the relationship between syntactic constructs and the elements of an universe of meanings (i.e., the semantics) is a function in mathematical sense (i.e., formal).
- **Example**: $$\begin{tikzcd} \text{Java} \arrow[r, "I_1"] \arrow[dr, "I_2"'] & \text{Island} \\ & \text{programming language} \end{tikzcd}$$

  Either I_1 or I_2 define a formal semantics of the term "Java", but I_1 and I_2 do not!

Levels of Formalization

Levels of Formalization Syntax and Semantics

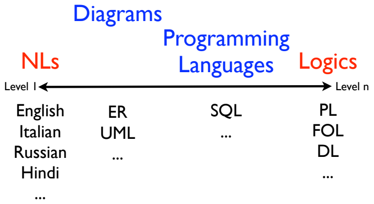
- Syntax can be formal or informal.
  - it depends on the formal/informal rules used to construct the expressions of the language from the alphabet.
- Semantics can be formal or informal as well.
  - it depends on the formal/informal relationship between expressions and the elements of some universe of meanings.

Levels of Formalization (a *continuum*)

- There is a *continuum* of languages for representation: Level 1 ⟷ Level n
- Level 1 (informal languages): informal syntax and informal semantics.
- Levels 2 to n-1 (semi-formal languages): informal syntax or informal semantics.

  with different degrees!
- Level n (formal languages): formal syntax and formal semantics.

Levels of Formalization Examples



NLs	Diagrams	Programming Languages	Logics
English	ER	SQL	PL
Italian	UML	...	FOL
Russian	...		DL
Hindi			...
...			

Efficiency and Effectiveness

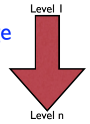
- It is an important function of the problem solver (and the modeler) to find the most appropriate representation for the problem.
- The use of the "appropriate" (i.e., effective) representation is capable of having spectacular effects on problem solving efficiency.
- This is an important tradeoff in modeling!

Efficiency, Effectiveness (Definition, *Webster*)

- Effective. Adequate to accomplish a purpose; producing the intended result.
  - in modeling it applies to representation; expressiveness of language.
- Efficient. Performing in the best possible manner; satisfactory and economical to use.
  - in modeling it applies to reasoning; computational complexity (time, space,...).

Levels of Formalization (a *continuum*)

- The modeler must choose the appropriate level (i.e. the appropriate representation language) at each stage of modeling.
- We now consider three classes of languages:



  - Natural Language - informal
  - Diagrams - semi-formal
  - Logic - formal

Use of Basic Classes of Languages

- Natural Languages: typically used at the very beginning of modeling, when no other "easier" representation language is known.
- Diagrams : typically used in graph-based languages (e.g. IHs, UML, RDF,...).
- Logics: typically used in logic-based ontology languages (e.g. DLs, FOL, rules,...)
  - used when a lot of the problem is known.

Levels of Formalization Crucial Factor

The more you formalize
the higher is the cost of representation!

(factor of 100, at least as a ball-park)

Natural Language

Examples

- Consider two "famous" problems:

  MC - Missionaries-Cannibals (Amarel, 1968)
  MB - Monkey-Bananas (McCarthy, 1969)
- First examples in Artificial Intelligence (AI) of problems with an "analytical" formulation.
- Knowledge Representation (KR) begins as an independent research area of AI.

Example (MC)

- Let's represent the scenario in natural language (English):
- The question to solve is: How shall the missionaries cross the river?
- To answer we need to reason about data and knowledge (see text).

"Three missionaries and three cannibals come to a river. A rowboat that seats two is available. If the cannibals ever outnumber the missionaries on either bank of the river, the missionaries will be eaten."

Example (MB)

- Let's represent the scenario in natural language (English):
- The question to solve is: How shall the monkey reach the bananas?
- To answer we need to reason about data and knowledge (see text).

"There is a monkey in a laboratory with some bananas hanging out of reach from the ceiling. A box is available that will enable the monkey to reach the bananas if he climbs on it. Initially, the monkey is at A, the bananas at B, and the box at C. The monkey and box have height Low, but if the monkey climbs onto the box he will have height High, the same as the bananas. [...]"

Remark

- A solution to a problem would start to represent data ("facts") and knowledge ("relations on data") in an explicit way.
  - Some data (red) and knowledge (blue) in the MB problem are:
    - the monkey, the bananas, the positions A, B, C;
    - Low and High; Go, ClimbUp, etc.

Example (Matching on Bi-Graphs)

- Let's represent the scenario in natural language (English):
- The question to solve is: Given a bipartite graph, does it have a matching?
- Exercise:
  1. what is data?
  2. what is knowledge?

"Define a bipartite graph to be a triple, where the first element of the triple is a set of boys, the second element is a set of girls, and the third element is a set of links between boys and girls. A (perfect) matching in the bipartite graph is ..."

Example (Organization)

- Let's represent the scenario in natural language (English):
- The question to solve is, for example:
  Given the firm organized as described by the text on the right,
  is there a manager who works for at least one project?

*"There is a firm producing cars.
The firm is organized according to a hierarchy; area managers and managers are managers, and managers are employee.
Every manager manages just one project, which is referred by a code ..."*

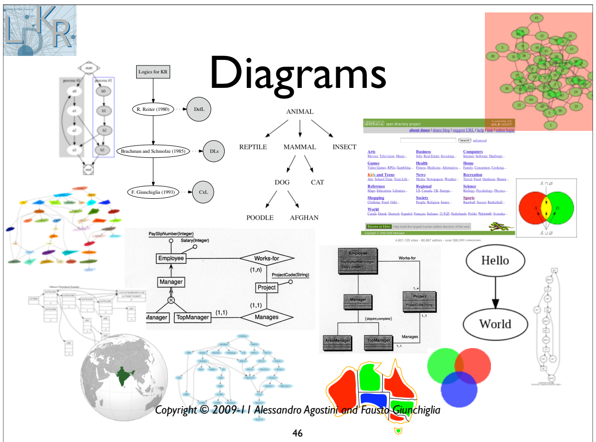
Diagrams

Diagrams

- We consider three typologies of diagrams:

  1. Graphs (w-w/o labels).
  2. Entity-Relationship (E-R) schemas.
  3. UML Class diagrams.
- Syntactically, 2. and 3. are special cases of 1.
- 1. do *not* have semantics.
- 2. and 3. are graph-based ontology languages.

Diagrams



Diagrams (DB tables)

Teach	Teacher	Student
	a	A
	b	B
	c	C
	a	B

Girl	Valentina	...
	red(x1,y1)	
	red(x2,y2)	
	white(x3,y3)	
	orange(x4,y4)	
	...	

Girl	Benedetta	...
	red(x1,y1)	
	pink(x2,y2)	
	braun(x3,y3)	
	yellow(x4,y4)	
	...	

Girl	Eleonora
	yellow(x1,y1)
	gray(x2,y2)
	yellow(x3,y3)
	yellow(x4,y4)
	...

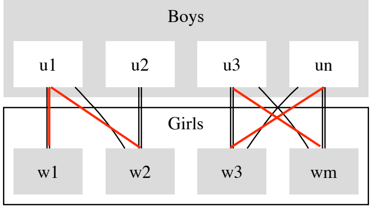
Background	Snow	...
	white(x1,y1)	
	white(x2,y2)	
	white(x3,y3)	
	white(x4,y4)	
	...	

Graphs

- Ubiquitous in computer science (e.g., social networks, knowledge structures, ...).
- Important problems in CS are (variants of) a graph-problem, e.g. the Matching Problems.
- These are "bipartite graphs" problems.
- Def. A (undirected) bi-partite graph (or "bigraph") is a pair

$$B = \langle (U, V), E \rangle$$

Example
(Matching on Bi-Graphs)



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

E-R Schemas

- Data representation models for databases design under the Entity-Relationship Model.
  - The E-R Model (Chen, 1976) is a conceptual data model.
- It provides basic "constructs" for schemas: Entity (or Class), Relation (or Association), Attribute (simple and composed), etc.

The E-R Constructs (1)

- Constructs represents "piece of the world."
- Entity: class of objects with same properties. These are data in modeling with E-R.
- Relation: logical relationships among entities. These are knowledge in modeling with E-R.
- Attribute: basic properties of the entities or relations of interest in the application (*basic* data or knowledge in modeling with E-R).

The E-R Constructs (2)

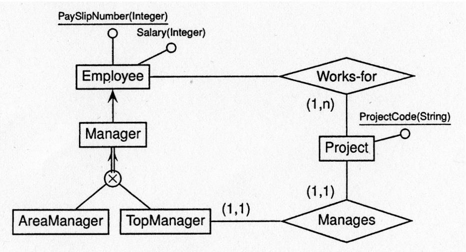
- Cardinality of Relation: min/max number of objects in an entity that may be related by a relation to an object in a different entity.
- Cardinality of Attribute: range of values of an attribute of an entity or a relation.
- Entity Identificator: attribute or entity that provides unique identification of an entity.

The E-R Constructs (3)

- Generalization: logical relationship between an entity E (father entity) and a set of entities E_1...E_i...E_n(sons entities). For each i,
  - E generalizes E_i and E_i specializes E.
  - E_i is a subset of E (IS-A relation).
  - Properties of E are also of E_i (inheritance), but not necessarily viceversa.

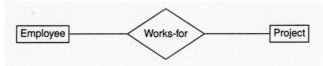
Example

This example is due to Enrico Franconi.
Source: http://www.inf.unibz.it/~franconi/dl/course/



E-R Semantics (Example, cont’)

- Meaning of Relations:

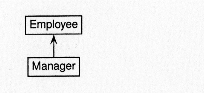


is:

  - informal (natural language) semantics: every employee works for a project.

E-R Semantics (Example, cont’)

- Meaning of IS-A relation:

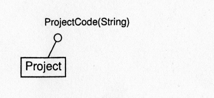


is:

  - informal (natural language) semantics: every manager is an employee.

E-R Semantics (Example, cont’)

- Meaning of Attributes:

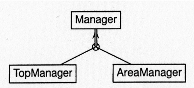


is:

  - informal (natural language) semantics: every project has a code, which is a string.

E-R Semantics (Example, cont’)

- Meaning of Generalizations:

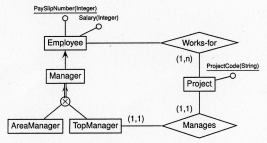


is:

  - informal (natural language) semantics: Total: every manager is either a top manager or an area manager. Disjoint: there is no topmanager that is an area maneger and vice versa.

E-R Semantics (Exercise)

- Represent the semantics of the E-R schema below by using a natural language (English or Italian).



UML Class Diagrams

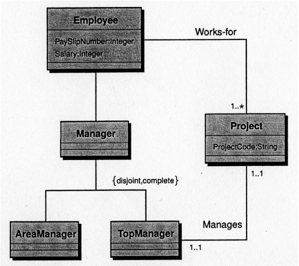
- UML = Unified Modeling Language.
- Representation language to model data, operations, processes, architectures.
- From software engineering under the paradigm of object-oriented programming.
- UML class diagrams useful in database design as an “alternative” to the E-R schemas.

UML Class Diagrams

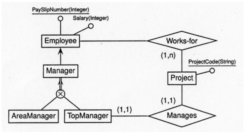
- Basic components of an UML class diagram
  - Class
  - Association.
- Classes and associations correspond to entities and relations in the E-R schemas.
- The elements in a class are called instances.
- The world here is a set of instances.

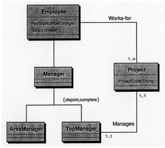
Example

This example is due to Enrico Franconi.
Source: http://www.inf.unibz.it/~franconi/dl/course/



Exercise

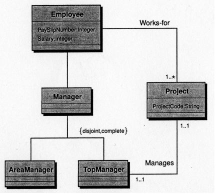




- Compare the UML class diagram with the E-R schema.
- What are the similarities and differences?

UML Class Diagrams (Exercise)

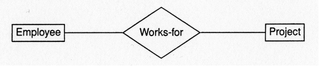
- Represent the semantics of the UML class diagram below by using English or Italian.



Logic

E-R Semantics (Example, cont')

- Meaning of Relations:

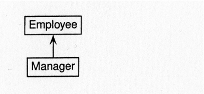


is:

- informal (natural language) semantics: every employee works for a project.
- formal (set-theoretic) semantics: Works-for ⊆ (Employee × Projects)

E-R Semantics (Example, cont')

- Meaning of IS-A relation:

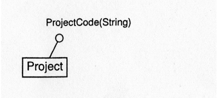


is:

  - informal (natural language) semantics: every manager is an employee.
  - formal (set-theoretic) semantics: $\text{Manager} \subseteq \text{Employee}$

E-R Semantics (Example, cont')

- Meaning of Attributes:

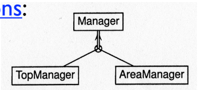


is:

  - informal (natural language) semantics: every project has a code, which is a string.
  - formal (set-theoretic) semantics: $\text{Project} \subseteq \{p : |\,\text{ProjectCode} \cap (\{p\} \times \text{String})\,| \geq 1\}$

E-R Semantics (Example, cont')

- Meaning of Generalizations:



is (formal, set-theoretic):

- Total: $\text{Manager} \subseteq (\text{TopManager} \cup \text{AreaManager})$
- Disjoint: $\text{TopManager} \cap \text{AreaManager} = \varnothing$

E-R Formal Semantics (Example, summary)

$\text{Works-for} \subseteq \text{Employee} \times \text{Project}$

$\text{Manages} \subseteq \text{TopManager} \times \text{Project}$

$\text{Employee} \subseteq \{e : |\,\text{PaySlipNumber} \cap (\{e\} \times \text{Integer})\,| \geq 1\}$

$\text{Employee} \subseteq \{e : |\,\text{Salary} \cap (\{e\} \times \text{Integer})\,| \geq 1\}$

$\text{Project} \subseteq \{p : |\,\text{ProjectCode} \cap (\{p\} \times \text{String})\,| \geq 1\}$

$\text{TopManager} \subseteq \{m : 1 \geq |\,\text{Manages} \cap (\{m\} \times \Omega)\,| \geq 1\}$

E-R Semantics (Exercise)

1. Complete the definition of the formal, set-theoretic semantics for the E-R schema given in the foregoing example.
2. Represent the formal semantics of the previous slide informally by using natural language (English).

E-R Semantics (Solution of 1)

$\text{Project} \subseteq \{p : 1 \geq |\,\text{Manages} \cap (\Omega \times \{p\})\,| \geq 1\}$

$\text{Project} \subseteq \{p : 1 \geq |\,\text{Works-for} \cap (\Omega \times \{p\})\,| \geq 1\}$

$\text{Manager} \subseteq \text{Employee}$

$\text{AreaManager} \subseteq \text{Manager}$

$\text{TopManager} \subseteq \text{Manager}$

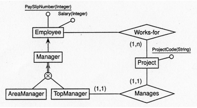
$\text{TopManager} \cap \text{AreaManager} = \varnothing$

$\text{Manager} \subseteq (\text{TopManager} \cup \text{AreaManager})$

Example

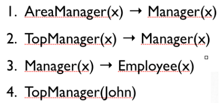
Suppose we know that:

1. AreaManager(x) → Manager(x)
2. TopManager(x) → Manager(x)
3. Manager(x) → Employee(x)
4. TopManager(John)



Example (cont’)

Then we can deduce the following:



1. AreaManager(x) → Manager(x)
2. TopManager(x) → Manager(x)
3. Manager(x) → Employee(x)
4. TopManager(John)

5.Manager(John)

6. Employee(John)

7.TopManager(x) → Employee(x)

- 5, 6, 7, represent new knowledge obtained by deduction from previous knowledge.

Example (cont’) Take-away Point

- Logic represents knowledge already captured by the E-R constructs (inheritance)

➡ but it extends the expressive power of the E-R schemas (e.g., Manager(John)).

- Logic uncovers (ontological) knowledge making it explicit by derivation.
- To be useful in applications, formalization (by logic) must be computationally efficient.

Specification and Automation

Specification and Automation

- There are two purposes in modeling:
  - Specification: proper representation
    - requires at least an *informal* syntax and an *informal* semantics.
  - Automation: representation of reasoning that can be automated (computed).
    - requires at least a *formal* semantics.

Why Natural Languages?

- Used as informal specification languages
  - often used at the very beginning of problem solving, when we need a direct, “flexible”, well-understood language
  - its syntax is used more than its semantics
  - semantics is informal, largely ambiguous
- Pragmatically inefficient for computation

Why Diagrams?

- Used as semi-formal specification languages
  - informal/formal syntax (depends on the kind of diagram)
  - informal semantics
  - largely structured and organized; Example: NL to UML
  - Pragmatically inefficient for computation

Why Logic?

- Used as formal specification languages
  - formal syntax, formal semantics
  - well-understood
- Used as (formal) languages for automation
  - "reasoning services" (see the next slides)
- Pragmatically efficient according to the definition and properties of the semantics.

Where Logic?

- Formal methods:
  - e.g., safety critical, security, ...
- Automation needed:
  - e.g., semantic web, scheduling, ...

Logics for Data and Knowledge Representation

Logics

- A logic is a representation language with
  - a *formal* syntax;
  - a *formal* semantics.
- Because of this, logics are *formal* languages.
- As formal languages, logics are suitable for representing (specification) of and reasoning (automation) about data and knowledge.

Logics for Specification

- Logic as a formal language: is good for the specification (representation) of knowledge.
  - it provides a formal syntax.
- Starting from a precise syntax logic provides a formal (i.e., unambiguous) semantics.
- A *general* source on specification languages::
  http://en.wikipedia.org/wiki/Specification_Language

Logics for Specification

- Logic as a formal semantics: is good for specification of declarative knowledge.
  - meaning of sentences is declaratively defined, i.e. with logic we describe what holds (declarative knowledge) without caring about how it can be deduced.
- A specification issue is to represent the "reasoning services" of interest (see below).

Logics for Reasoning

- Logic provides a "reasoner" that can be used to deduce (infer) conclusions from a given knowledge base (i.e. a set of "premises").
  - This makes implicit knowledge explicit.
- *Automated* deduction provides automated explanations for conclusions ("tracking").
- The "reasoning services" of a logic-based system depend on a suitable automation.

Logics for *Automated* Reasoning

- Logic provides a "reasoner" that can be used to deduce (infer) conclusions from a given knowledge base (i.e. a set of "premises").
  - This makes implicit knowledge explicit.
- *Automated* deduction provides automated explanations for conclusions ("tracking").
- The "reasoning services" of a logic-based system depend on a suitable automation.

Reasoning Services (1)

- The basic reasoning tasks (or "services") we would like to compute are the following:
  - **Model Checking (EVAL)**: Is a sentence ψ true according to a model M?
  - **Validity**: Is a sentence ψ true according to *every possible* model M?
  - **Satisfiability (SAT)**: Is KB satisfiable?

Satisfiability (Example)

- Let KB be the set of "expressions" we may use to represent the picture on the right in some language L.
- **Satisfiability (SAT)**: Can we validate all the expressions in KB by using the semantics of the language L we used?

Reasoning Services (2)

- **Entailment**: Is ψ true in a model M if the world represented in KB is true in M?
- **Consistency**: Is ψ consistent in KB?
- **Other services:** subsumption, instance checking, equivalence, concept coherence, ...
- **NB:** not all reasonig services are provided by a single logic (e.g., instance checking).

Entailment (Example)

- Let KB be the set of “expressions” of a suitable language (e.g., English) that we may use to represent the knowledge (partial or complete) we gained from the picture.

- **Entailment**:
  Are the skiboots of Valentina orange if Valentina is the right-most girl?

Consistency (Example)

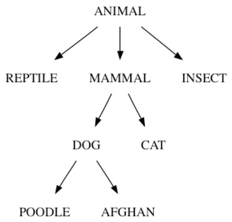
- Let KB be the set of “expressions” we use to represent the knowledge (partial or complete) we have from the picture.

- **Consistency**:
  Is it consistent w.r.t. the knowledge represented in KB to assume that Valentina’s skiboots are yellow?

Reasoning Services (3) (specific of DLs)

- **Classification of concepts**: determine sub-concept / super-concept relationships (subsumption relationships) between the concepts of a given terminology.
- **Classification of objects**: determine whether a given individual is always an instance of a certain concept.

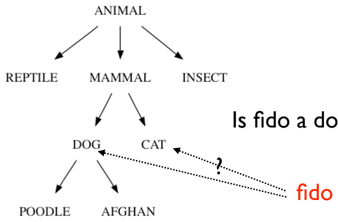
Concepts Classification Example



Are insects animals?
Are dogs insects?
...

(Etherington & Reiter, 1983)

Objects Classification Example (cont’)



Is fido a dog or a cat?

(Etherington & Reiter, 1983)

This Course (Summary)

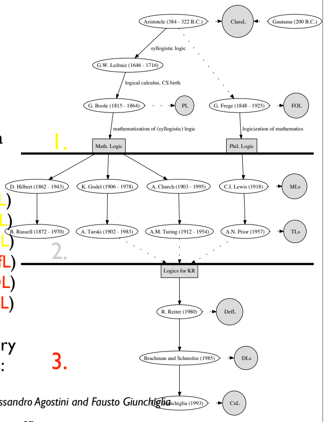
- Logic as a general modeling language for
  - specification (i.e., representation)
  - automation (i.e., automated reasoning services)
- Logics for
  - Data
  - Knowledge
- How to use logic in modeling.

Course Outline

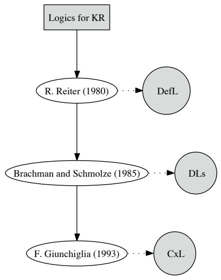
- We'll see the basics of a number of logics:

Propositional Logic (PL)
Class Logic (ClassL)
First-Order Logic (FOL)
Default Logic (DefL)
Description Logic (DL)
Context Logic (CxL)

- We can divide the history of logic into three parts:

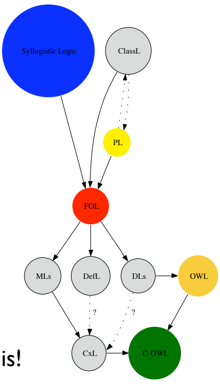


Logics of Knowledge



- These logics are the most important for knowledge modeling in computer science
- All are rooted in first-order logic
- We shall see how!

Logics Dependencies



In this course we'll see all this!