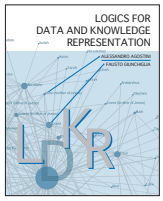
Logics for Data and Knowledge Representation

Alessandro Agostini
agostini@dit.unitn.it

Fausto Giunchiglia
fausto@dit.unitn.it

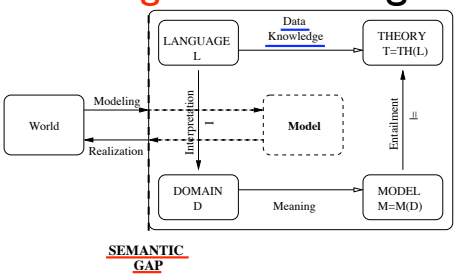
University of Trento

Outline



- Logical Modeling: Diagram
- Expressiveness
- Complexity
- Tradeoff
- Decidability

Fundamental Diagram of *Logical* Modeling



Logical Modeling Elements

- The basic elements in the diagram are:

  1. Domain (data, classes, relations, functions)
  2. Logical Language
  3. Interpretation (of the language)
  4. Model
  5. Theory / KnowledgeBase (knowledge)
  6. Truth-relation / logical entailment (|=)

- We illustrate each element in turn.

Domain

- Domain (D) = the chosen objects (data) from the world.
- **Example** (LDKR class): the members of the LDKR class define a domain D;
  - D is a finite set.
  - The "type" of the elements in D is: person.
- **NB**: we will deal only with finite domains!

Language

- Language (L) = a logical language, i.e.
  1. L's alphabet of symbols Σ contains at least one of the logical symbols: ∧, ∨, ¬, →, ∀, ∃;
  2. L has clear formation rules for formulas.
- **Example** (cont'): any logical language with =, (,), professor, student1, student2,... in Σ.
- Note that English *is* a propositional language, ...but it is *not logical* (informal syntax).

Logical Language (Syntax)

- The first step in setting up a logical language (viz. a formal language) is to list the symbols, that is, the alphabet of (formal) symbols (Σ).
  - formal symbol = a character, or group of characters taken from some alphabet.
- Symbols in Σ can be divided in 'descriptive' (nonlogical) and 'non-descriptive' (logical).

Example

- Take the Monkey-Bananas Problem.
- In the sentence "*There is a monkey in a laboratory with some bananas*":
  - Descriptive symbols are: '*monkey*', '*laboratory*','*bananas*', ...
  - Non-descriptive symbols are: *'there is*', *'with some*', *'a*', *'in*', ...

Language (Syntax)

- Descriptive symbols (the terminology is by Dag Prawitz, 1965)) refer to the specific language we need to represent the problem.
- Non-descriptive symbols refer to logic, i.e. the elements of the language that are related only to the structure of a specific logic and, as such, they are *not related* to the problem.

Remark

- The alphabet of symbols (Σ) is analogous structurally to the alphabet of a language,
  - although in a *formal* language many of the symbols correspond to entire words or phrases rather than to single letters.
- **Example:** In the MB problem, we model the monkey by using the symbol "Monkey", that is in fact word rather that a single letter.

Formal Syntax (Definition)

- Formal Syntax: the set of "rules" saying how to construct the expressions of the language from the alphabet of symbols, (i.e., the syntax) is a grammar (i.e., formal).
- **Example**: context-free grammars.
- Remark: Formal syntax is often called an abstract syntax, in contrast to the concrete syntax used, e.g., in implementations.

Interpretation

- Interpretation (I) = a mapping of L into D.
  - NB: the modeler always requires an effective (i.e., computable) mapping.
  - Note that with a *finite* alphabet of symbols Σ, an interpretation is always computable.
- **Example** (LDKR class, cont'):
  $I(prof_2)$ = Fausto,
  $I(student_2)$ = Audrey, ...

Model

- Model (M) = the abstract (mathematical sense) representation of the intended truths about D via interpretation I of language L.
  - M called L-model of D.
- **Example** (LDKR class, cont'):
  A model of the class would formalize that:
  a. I(prof2) is not I(student2),
  b. #letters I(prof2) = #letters I(student2), ...

Theory

- Theory T (also L-Theory) = set of facts of L.
- Since a fact defines a piece of knowledge (about D), if T is finite then it is called a knowledge base (denoted by **KB**).
  - A database (denoted by **DB**) is the simplest kind of knowledge base.
- **Example** (cont'): T = {¬(prof = student)}

Expressiveness of Language

Effectiveness & Expressiveness

- Effective (*Webster*). Adequate to accomplish a purpose; producing the intended result.
- When we deal with effectiveness to refer to the adequateness of a representation language for modeling purposes we use the more specific term expressiveness.
- Now we see some "degrees of expressiveness" proper of a *logical* language.

Degrees of Expressiveness

- **Example**: We may classify the *logical* languages according to some "degrees of expressiveness" (other degrees apply):
  - Propositional
  - Modal
  - First-order
  - Higher-order

Example

- Propositional: "I like skiing"
  wffs: *I-like-skiing*, *like-skiing(I)*, ...
- Modal: "I believe I like skiing"
  wffs: $B(I\text{-}like\text{-}skiing)$, $B_I(like\text{-}skiing)$, ...
- First-order: "Every person likes skiing"
  wffs: $\forall x.like\text{-}skiing(x)$, ...
- Second-order: "All people like skiing"
  wffs: $\forall X.like\text{-}skiing(X)$, ...

Example (cont')

- Propositional: "I like skiing"
  wffs: *I-like-skiing, like-skiing(I), ...*
- Modal: "I believe I like skiing"
  wffs: B(*I-like-skiing*), B_I(*like-skiing*), ...
- First-order: "Every person likes skiing"
  wffs: $\forall x$.*like-skiing(x), ...*
- Second-order: "All people like skiing"
  wffs: $\forall X$.*like-skiing(X), ...*

Example (cont')

- Propositional: "I like skiing" (*)
  wffs: *I-like-skiing, like-skiing(I), ...*
- Modal: "I believe I like skiing"
  wffs: B(*I-like-skiing*), B_I(*like-skiing*), ...
- First-order: "Every person likes skiing"
  wffs: $\forall x$.*like-skiing(x)*. *Note: x=I special case* (*)
- Second-order: "All people like skiing"
  wffs: $\forall X$.*like-skiing(X), ...*

Example (cont')

- Propositional: "I like skiing"
  wffs: *I-like-skiing, like-skiing(I), ...*
- Modal: "I believe I like skiing"
  wffs: B(*I-like-skiing*), B_I(*like-skiing*), ...
- First-order: "Every person likes skiing"
  wffs: $\forall x$.*like-skiing(x), ...*
- Second-order: "All people like skiing"
  wffs: $\forall X$.*like-skiing(X), ...*

Complexity of Reasoning

Efficiency & Complexity

- Efficient (*Webster*). Performing in the best possible manner; satisfactory and economical to use.
  - in modeling it applies to reasoning;
  - in this case we use the more specific terminology computational complexity (time, space,...) in place of term efficiency.

Complexity

- Definition. Complexity (or computational complexity) of reasoning is the difficulty - e.g. measured by the tools of Computational Complexity Theory (CCT) - to compute a reasoning task expressed by using a logic.
- In analogy with degrees of expressiveness, we may classify the *logical* languages according to some "degrees of complexity".

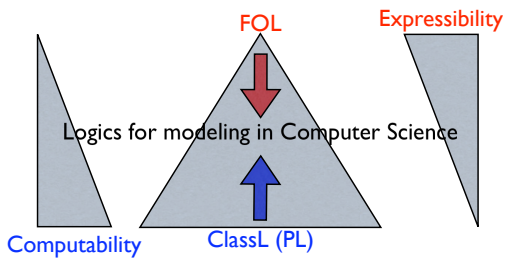
Degrees of Complexity

- The basic "degrees of complexity" are:
  - Untractable (undecidable)
  - Polynomial - the class P
  - Non-polynomial - the class NP
  - Other classes (PSPACE, coNP, LOG,...).
- We are not interested in the first degree!

An Important Trade-Off

- There is a trade-off between expressive power (expressiveness) and computational efficiency provided by a (logical) language.
  - This trade-off is a measure of the tension between specification and automation.
- To *use* logic for modeling, the modeler must trading off expressiveness in the language for more tractable forms of reasoning services.

A Pyramid of Logics



Decision Procedures

Decidability

- Definition. The question ("decision problem," from Hilbert, 1928) about the existance of an algorithm which would decide if a given "expression" P is provable or not.
  - P may be a proposition, a predicate, a concept, etc. (We' see it for each logic.)
  - For us (*semantic* approach, *finite* models): P is provable if and only if P is satisfiable.

Algorithm (Etimology)

- The term "algorithm" comes from the Persian Middle-Asian astronomer and mathematician Al-Khwarizmi (780- 850 A.C.)
- Al-Khwarizmi recognized the importance of special computational procedures solving mathematical problems (Wikipedia).

What is an Algorithm?

- Decidability is strictly related to the notion of "algorithm." There were two main approaches, and many equivalent definitions:

functional approach

- K. Gödel (1934): recursive functions.
  Alonzo Church (1936): lambda-calculus.
  S.C. Kleene (1936): functional equations.

computational approach

- Alan M. Turing (1936-37): Turing machines.
  Emil L. Post (1936): Post machines.

Decision Procedures Decidable Logics

- Definition. A decision procedure is an algorithm that, given a decision problem, terminates with the correct yes/no answer.
- Definition. A logic is decidable if there exists a decision procedure for that logic.
- In this course we focus on logics that are expressive enough to model real problems but are still decidable.