Outline

Modeling Information through Ontologies

Outline

Outline

New challenges in information management

One of the key challenges in complex systems today is the management of information:

- The amount of information has increased enormously.
- The complexity of information has increased:
  - structured $\rightsquigarrow$ semi-structured $\rightsquigarrow$ unstructured
- The underlying data may be of low quality, e.g., incomplete, inconsistent, not *crisp*.
- Information is increasingly distributed and heterogeneous, but nevertheless needs to be accessed in a uniform way.
- Information is consumed not only by humans, but also by machines.

Traditional data management systems are not sufficient anymore to fulfill today's information management requirements.

Addressing information management challenges

Several efforts come from the database area:

- New kinds of databases are studied, to manage semi-structured (XML), and probabilistic data.
- Information integration is one of the major challenges for the future or IT. E.g., the market for information integration software has been estimated to grow from $\$2.5 billion$ in 2007 to $\$3.8 billion$ in 2012 (+8.7% per year) [IDC. Worldwide Data Integration and Access Software 2008-2012 Forecast. Doc No. 211636 (2008)].

On the other hand, management of complex kinds of information has traditionally been the concern of Knowledge Representation in AI:

- Research in AI and KR can bring new insights, solutions, techniques, and technologies.
- However, what has been done in KR needs to be adapted / extended / tuned to address the new challenges coming from today's requirements for information management.

Description logics

Description Logics [Baader et al., 2003] are an important area of KR, studied for the last 25 years, that provide the foundations for the structured representation of information:

- By grounding the used formalisms in logic, the information is provided with a formal semantics (i.e., a meaning).
- The logic-based formalization allows one to provide automated support for tasks related to data management, by means of logic-based inference.
- Computational aspects are of concern, so that tools can provide effective support for automated reasoning.

In this course we are looking into using description logics for data management.

Ontologies

Description logics provide the formal foundations for ontology languages.

Definition (Ontology)

An Ontology is a representation scheme that describes a formal conceptualization of a domain of interest.

The specification of an ontology usually comprises two distinct levels:

- Intensional level: specifies a set of conceptual elements and of constraints/axioms describing the conceptual structures of the domain.
- Extensional level: specifies a set of instances of the conceptual elements described at the intensional level.

Note: an ontology may contain also a meta-level, which specifies a set of modeling categories of which the conceptual elements are instances.

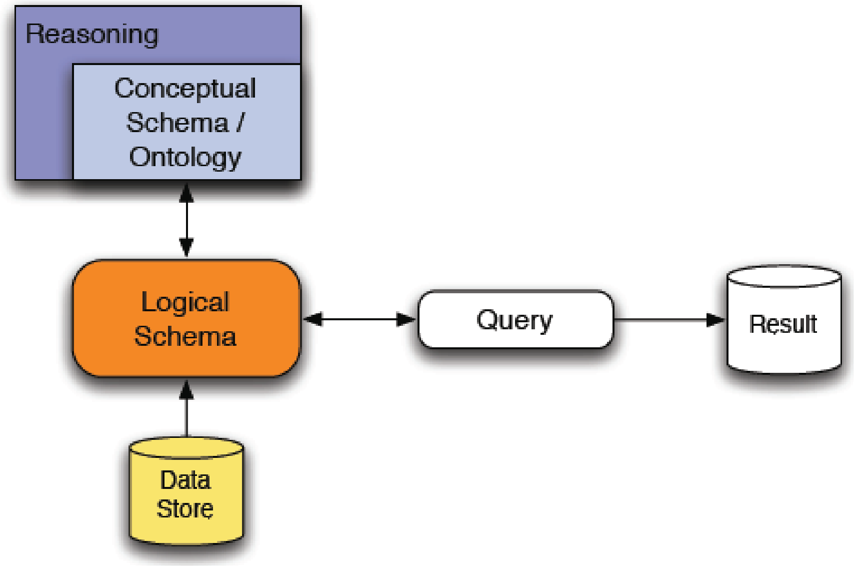
Outline

Conceptual schemas in information systems

Intensional information has traditionally played an important role in information systems.
Design phase of the information system:

1. From the requirements, a conceptual schema of the domain of interest is produced.
2. The conceptual schema is used to produce the logical data schema.
3. The data are stored according to the logical schema, and queried through it.

Conceptual schemas used at design-time



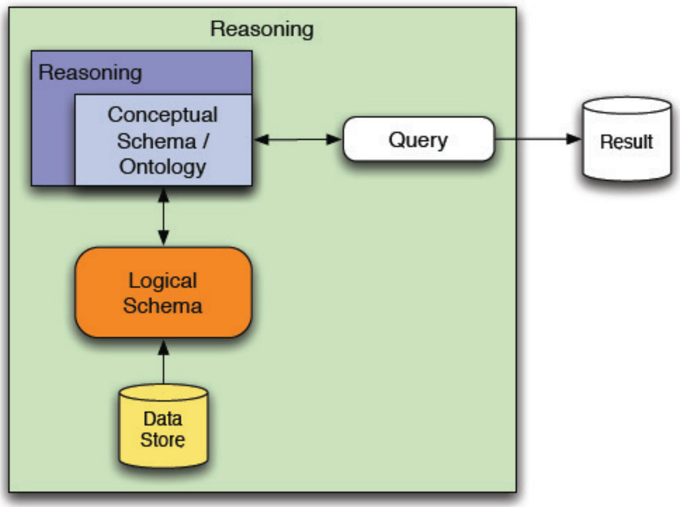
Ontologies in information systems

The role of ontologies in information systems goes beyond that of conceptual schemas. Ontologies affect the whole life-cycle of the information system:

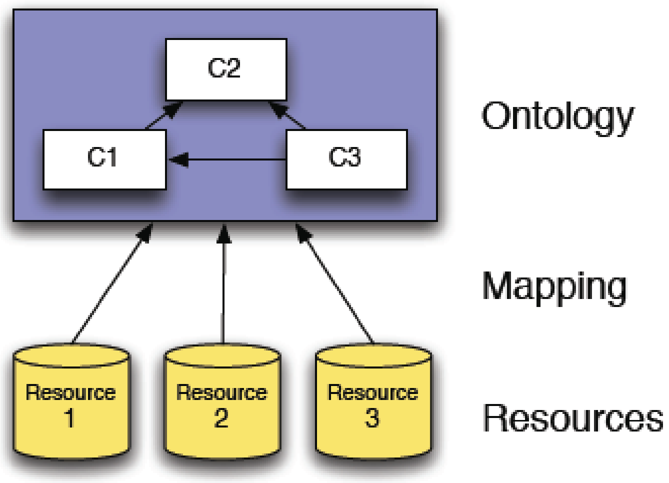
- Ontologies, with the associated reasoning capabilities and inference tools, can provide support at design time.
- The use of ontologies can significantly simplify maintenance of the information system's data assets.
- The ontology is used also to support the interaction with the information system, i.e., at run-time.

$\rightsquigarrow$ Reasoning to take into account the constraints coming from the ontology has to be done at run-time.

Ontologies used at run-time



Ontologies at the core of information systems



The usage of all system resources (data and services) is done through the domain conceptualization.

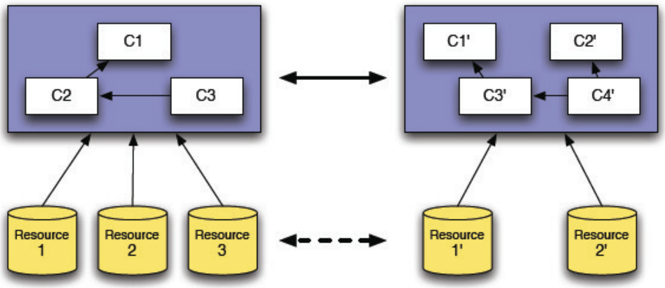
Ontology mediated access to data

Desiderata: achieve logical transparency in access to data:

- Hide to the user where and how data are stored.
- Present to the user a conceptual view of the data.
- Use a semantically rich formalism for the conceptual view.

This setting is similar to the one of Data Integration. The difference is that here the ontology provides a rich conceptual description of the data managed by the system.

Ontologies at the core of cooperation



The cooperation between systems is done at the level of the conceptualization.

Outline

Issues in ontology-based information management

1. Choice of the formalisms to adopt
2. Efficiency and scalability
3. Tool support

Issue 1: Formalisms to adopt

1. Which is the right ontology language?
    - many proposals have been made
    - differ in expressive power and in complexity of inference
2. Which languages should we use for querying?
    - requirements for querying are different from those for modeling
3. How do we connect the ontology to available information sources?
    - mismatch between information in an ontology and data in a data source

In the first part of this course:

- We present and discuss variants of ontology languages, and study their logical and computational properties.

Outline

Outline

What is a logic?

- The main objective of a logic (there is not a unique logic but many) is to express by means of a formal language the knowledge about certain phenomena or a certain portion of the world.
- The language of a logic is formal, since it is equipped with:
  - a formal syntax: it tells one how to write statements in the logic
  - a formal semantics: it tells one what the meaning of these statements is
- Considering the formal semantics, one can reason over given knowledge, and show which knowledge is a logical consequence of the given one.
- A logic often allows one to encode with a precise set of deterministic rules, called inference rules, the basic reasoning steps that are considered to be correct by everybody (according to the semantics of the logic).
- By concatenating applications of simple inference rules, one can construct logically correct reasoning chains, which allow one to transform the initial knowledge into the conclusion one wants to derive.

Real world, language, and mathematical structure

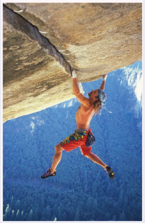
- Often we want to describe and reason about real world phenomena:
    - Providing a complete description of the real world is clearly impossible, and maybe also useless.
    - Typically one is interested in a portion of the world, e.g., a particular physical phenomenon, a social aspect, or modeling rationality of people, ...
- We use sentences of a language to describe objects of the real world, their properties, and facts that hold.
    - The language can be:
        - informal (natural lang., graphical lang., icons, . . .) or
        - formal (logical lang., programming lang., mathematical lang., . . .)
    - It is also possible to have mixed languages, i.e., languages with parts that are formal, and others that are informal (e.g., UML class diagrams)
- If we are also interested in a more rigorous description of the phenomena, we provide a mathematical model:
    - Is an abstraction of the portion of the real world we are interested in.
    - It represents real world entities in the form of mathematical objects, such as sets, relations, functions, . . .
    - Is not commonly used in everyday communication, but is commonly adopted in science, e.g., to show that a certain argumentation is correct

Language, real world, and mathematical models: Example

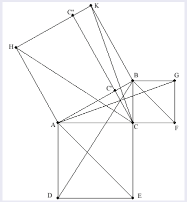
Language

In any right triangle, the area of the square whose side is the hypotenuse (the side opposite the right angle) is equal to the sum of the areas of the squares whose sides are the two legs (the two sides that meet at a right angle).

Real world



Mathematical model



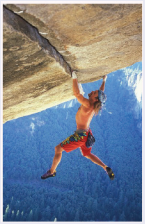
- Facts about euclidean geometry can be expressed in terms of natural language, and they can refer to one or more real world situations. (In the picture it refers to the composition of the forces in free climbing).
- However, the importance of the theorem lays in the fact that it describes a general property that holds in many different situations. All these different situations can be abstracted in the mathematical structure which is the euclidean geometry. So indeed the sentence can be interpreted directly in the mathematical structure.
- In this example the language is informal but it has an interpretation in a mathematical structure.

Language, real world, and mathematical models: Example 2

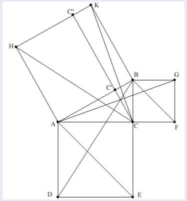
Language

In triangle ABC, if $\widehat{ABC}$ is right, then $\overline{AB}^2 + \overline{AC}^2 = \overline{BC}^2$

Real world



Mathematical model



- This example is obtained from the previous one by taking a language that is "more formal". Indeed the language mixes informal statements (e.g., "if... then..." or "is right") with some formal notation.
- E.g.,$\widehat{BAC}$ is an unambiguous and compact way to denote an angle. Similarly $\overline{AB}^2 + \overline{AC}^2 = \overline{BC}^2$ is a rigorous description of an equation that holds between the lengths of the triangle sides.

Language, real world, and mathematical models: Example 3

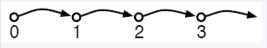
Language

$x - 2y + 3$

$x + y = 0$

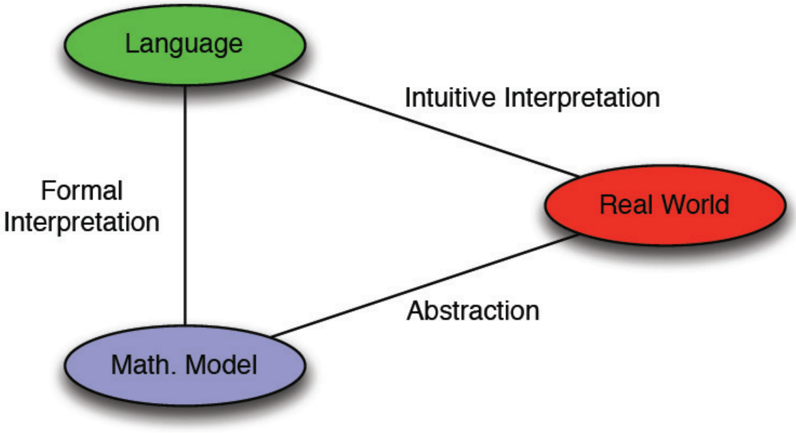
Real world

Mathematical model



- In this example the language is purely formal, i.e., the language of arithmetic.
- This abstract language is used to represent many situations in the real world (in the primary school we have many examples about apples, pears, and how much they cost, which are used by teachers to explain to kids the intuitive meaning of the basic operations on numbers).
- The mathematical model in this case is the structure of natural numbers.

Connections between language, world, and math. model



Connections between language, world, and math. model

Intuitive interpretation (or informal semantics)

When you propose a new language (or when you have to learn a new language) it is important to associate to any element of the language an interpretation in the real world. This is called the intuitive interpretation (or informal semantics). E.g., in learning a new programming language, you need to understand what is the effect in terms of execution of all the languages construct. For this reason the manual, typically, reports in natural language and with examples, the behavior of the language primitives. This is far to be a formal interpretation into a mathematical model. Therefore it is an informal interpretation.

Formal interpretation (or formal semantics)

Is a function that allows one to transform the elements of the language (i.e., symbols, words, complex sentences,...) into one or more elements of the mathematical structure. It is indeed the formalization of the intuitive interpretation (or the intuitive semantics).

Abstraction

Is the link that connects the real world with it's mathematical and abstract representation into a mathematical structure. If a certain situation is supposed to be abstractly described by a given structure, then the abstraction connects the elements that participate to the situation, with the components of the mathematical structure, and the properties that hold in the situation with the mathematical properties that hold in the structure.

Outline

Logic

Logic is a special case of the framework we have just seen, where the following important components are defined:

- The language is a logical language.
- The formal interpretation allows one to define a notion of truth.
- It is possible to define a notion of logical consequence between formulas. I.e., if a set Γ of formulas are true then also φ is true.

Formal language

- We are given a non-empty set Σ of symbols called alphabet.
- A formal language (over Σ) is a subset L of Σ^*, i.e., a set of finite strings of symbols in Σ.
- The elements of L are called well formed phrases.
- Formal languages can be specified by means of a grammar, i.e., a set of formation rules that allow one to build complex well formed phrases starting from simpler ones.

Logical language

A language of a logic, i.e., a logical language is a formal language that has the following characteristics:

- The alphabet typically contains basic symbols that are used to indicate the basic (atomic) components of the (part of the) world the logic is supposed to describe. Examples of such atomic objects are, individuals, functions, operators, truth-values, propositions,...
- The grammar of a logical language defines all the possible ways to construct complex phrases starting from simpler ones.
  - A logical grammar always specifies how to build formulas, which are phrases that denotes propositions, i.e., objects that can assume some truth value (e.g., true, false, true in certain situations, true with probability of 3%, true/false in a period of time,...).
  - Another important family of phrases which are usually defined in logic are terms which usually denote objects of the world (e.g., cats, dogs, time points, quantities,...).

Alphabet

The alphabet of a logical language is composed of two classes of symbols:

- Logical constants, whose formal interpretation is constant and fixed by the logic (e.g., $\wedge, \forall, =, \dots$).
- Non logical symbols, whose formal interpretation is not fixed by the logic, and must be defined by the "user".

We can make an analogy with programming languages (say C, C++, python):

- Logical constants correspond to reserved words (whose meaning is fixed by the interpreter/compiler).
- Non logical symbols correspond to the identifiers that are introduced by the programmer for defining functions, variables, procedures, classes, attributes, methods,...

The meaning of these symbols is fixed by the programmer.

Alphabet: Logical constants - Example

The logical constants depend on the logic we are considering:

- Propositional logic: $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$(negation), $\supset$ (implication), $\equiv$ (equivalence), $\perp$ (falsity). These are usually called propositional connectives.
- Predicate logic: in addition to the propositional connectives, we have quantifiers:
  - universal quantifier $\forall$, standing for "every object is such that ..."
  - existential quantifier $\exists$, standing for "there is some object that ..."
- Modal logic: in addition to the propositional connectives, we have modal operators:
  - $\Box$ standing for "it is necessarily true that..."
  - $\diamond$, standing for "it is possibly true that ..."

Alphabet: Non-logical symbols - Example

- Propositional logic: non logical symbols are called propositional variables, and represent (i.e., have intuitive interpretation) propositions. The proposition associated to each propositional variable is not fixed by the logic.
- Predicate logic: there are four families of non logical symbols:
  - Variable symbols, which represent any object.
  - Constant symbols, which represent specific objects.
  - Function symbols, which represent transformations on objects.
  - Predicate symbols, which represent relations between objects.
- Modal logic: non logical symbols are the same as in propositional logic, i.e., propositional variables.

Example of grammar: Language of propositional logic

Grammar of propositional logic

Allows one to define the unique class of phrases, called formulas (or well formed formulas), which denote propositions.

$$\begin{array}{lcll} Formula & \rightarrow & P & (P \text{ is a propositional variable}) \\ & \mid & (Formula \wedge Formula) & \\ & \mid & (Formula \vee Formula) & \\ & \mid & (Formula \rightarrow Formula) & \\ & \mid & (\neg Formula) & \end{array}$$

Example (Well formed formulas)

$(P \wedge (Q \rightarrow R))$ $\quad$ $((P \rightarrow (Q \rightarrow R)) \vee P)$

These formulas are well formed, because there is a sequence of applications of grammar rules that generates them.

Exercise: list the rules in each case.

Example (Non well formed formulas)

$P(Q \rightarrow R)$

$(P \rightarrow \vee P)$

Example of grammar: Language of first order logic

Grammar of first order logic

$$
\boxed{
\begin{array}{lcl}
Term & \rightarrow & x \quad (x \text{ is a variable symbol}) \\
 & \mid & c \quad (c \text{ is a constant symbol}) \\
 & \mid & f(Term, \ldots, Term) \ (f \text{ is a function symbol}) \\
Formula & \rightarrow & P(Term, \ldots, Term) \ (P \text{ is a predicate symbol}) \\
 & \mid & Formula \wedge Formula \\
 & \mid & Formula \vee Formula \\
 & \mid & Formula \rightarrow Formula \\
 & \mid & \neg Formula \\
 & \mid & \forall x(Formula) \ (x \text{ is a variable symbol}) \\
 & \mid & \exists x(Formula) \ (x \text{ is a variable symbol})
\end{array}
}
$$

The rules define two types of phrases:

- terms denote objects (they are like noun phrases in natural language)
- formulas denote propositions (they are like sentences in natural language)

Exercise

Give examples of terms and formulas, and of phrases that are neither of the two.

Example of grammar: Language of description logic

Grammar of the description logic $\mathcal{ALC}$

$$
\begin{array}{lcl}
\text{Concept} & \rightarrow & A \quad (A \text{ is a concept symbol}) \\
 & \mid & Concept \sqcup Concept \\
 & \mid & Concept \sqcap Concept \\
 & \mid & \neg Concept \\
 & \mid & \exists Role.Concept \\
 & \mid & \forall Role.Concept \\
\text{Role} & \rightarrow & R \quad (R \text{ is a role symbol}) \\
\text{Individual} & \rightarrow & a \quad (a \text{ is an individual symbol}) \\
\text{Formula} & \rightarrow & Concept \sqsubseteq Concept \\
 & \mid & Concept(Individual) \\
 & \mid & Role(Individual; Individual)
\end{array}
$$

Example (Concepts and formulas of the DL $\mathcal{ALC}$)

- Concepts: $A \sqcap B$, $A \sqcup \exists R.C$, $\forall S.(C \sqcup \forall R.D) \sqcap \neg A$
- Formulas: $A \sqsubseteq B$, $A \sqsubseteq \exists R.B$, $A(a)$, $R(a, b)$, $\exists R.C(a)$

Outline

Intuitive interpretation of a logical language

While, non logical symbols do not have a fixed formal interpretation, they usually have a fixed intuitive interpretation. Consider for instance:

Type	**Symbol**	**Intuitive interpretation**
propositional variable	$rain$	it is raining
constant symbol	$MobyDick$	the whale of a novel by Melville
function symbol	$color(x)$	the color of the object x
predicate symbol	$Friends(x, y)$	x and y are friends

The intuitive interpretation of the non logical symbols does not affect the logic itself.

- In other words, changing the intuitive interpretation does not affect the properties that will be proved in the logic.
- Similarly, replacing these logical symbols with less evocative ones, like r, $M, c(x), F(x, y)$ will not affect the logic.

Interpretation of complex formulas

The intuitive interpretation of complex formulas is done by combining the intuitive interpretations of the components of the formulas.

Example

Consider the propositional formula:

$$(raining \lor snowing) \rightarrow \neg go\_to\_the\_beach$$

If the intuitive interpretations of the symbols are:

symbol	**intuitive meaning**
$raining$	it is raining
$snowing$	it is snowing
$go\_to\_the\_beach$	we go to the beach
$\lor$	either ... or ...
$\rightarrow$	if ... then ...
$\neg$	it is not the case that ...

then the above formula intuitively represent the proposition:

- if (it is raining or it is snowing) then it is not the case that (we go to the beach)

Formal model

- Class of models: The models in which a logic is *formally interpreted* are the members of a class of algebraic structures, each of which is an abstract representation of the relevant aspects of the (portion of the) world we want to formalize with this logic.
- Models represent only the components and aspects of the world which are relevant to a certain analysis, and abstract away from irrelevant facts. Example: if we are interested in the average temperature of each day, we can represent time with the natural numbers and use a function that associates to each natural number a floating point number (the average temperature of the day corresponding to the point).
- Applicability of a model: Since the real world is complex, in the construction of the formal model, we usually do simplifying assumptions that bound the usability of the logic to the cases in which these assumptions are verified. Example: if we take integers as formal model of time, then this model is not applicable to represent continuous change.
- Each model represents a single possible (or impossible) state of the world. The class of models of a logic will represent all the (im)possible states of the world.

Formal interpretation

- Given a structure S and a logical language L, the formal interpretation in S of L is a function that associates an element of S to any non logical symbol of the alphabet.
- The formal interpretation in the algebraic structure is the parallel counterpart (or better, the formalization) of the intuitive interpretation in the real world.
- The formal interpretation is specified only for the non logical symbols.
- Instead, the formal interpretation of the logical symbols is fixed by the logic.
- The formal interpretation of a complex expression e, obtained as a combination of the sub-expressions $e_1, \ldots, e_n$, is uniquely determined as a function of the formal interpretation of the sub-components $e_1, \ldots, e_n$.

Truth in a structure: Models

- As said, the goal of logic is the formalization of what is true/false in a particular world. The particular world is formalized by a structure, also called an interpretation.
- The main objective of the formal interpretation is that it allows to define when a formula is true in an interpretation.
- Every logic therefore defines the satisfiability relation (denoted by $\models$) between interpretations and formulas.
- If $\mathcal{I}$ is an interpretation and φ a formula, then

$$\mathcal{I} \models \varphi$$

  stands for the fact that $\mathcal{I}$ satisfies φ, or equivalently that φ is true in $\mathcal{I}$.
- An interpretation $\mathcal{M}$ such that $\mathcal{M} \models \varphi$ is called a model of φ.

(Un)satisfiability and validity

On the basis of truth in an interpretation ($\models$) the following notions are defined in any logic:

- φ is satisfiable if it has model, i.e., if there is a structure $\mathcal{M}$ such that $\mathcal{M} \models \varphi$
- φ is un-satisfiable if it is not satisfiable, i.e., it has no models.
- φ is valid, denoted $\models \varphi$, if is true in all interpretations.

Logical consequence (or implication)

- The notion of logical consequence (or implication) is defined on the basis of the notion of truth in an interpretation.
- Intuitively, a formula φ is a logical consequence of a set of formulas (sometimes called assumptions) Γ (denoted $\Gamma \models \varphi$) if such a formula is true under this set of assumptions.
- Formally, $\Gamma \models \varphi$ holds when:

  For all interpretations $\mathcal{I}$, if $\mathcal{I} \models \Gamma$ then $\mathcal{I} \models \varphi$.

  In words: φ is true in all the possible situations in which all the formulas in Γ are true.
- Notice that the two relations, “truth in a model” and “logical consequence” are denoted by the same symbol $\models$ (this should remind you that they are tightly connected).

Outline

Difference between $\models$ and implication ($\rightarrow$)

At a first glance $\models$ looks like implication (usually denoted by $\rightarrow$ or $\supset$). Indeed in most of the cases they represent the same relation between formulas.

Similarity

- For instance, in propositional logic (but not only) the fact that φ is a logical consequence of the singleton set $\{\psi\}$, i.e., $\{\psi\} \models \varphi$, can be encoded in the formula $\psi \rightarrow \varphi$.
- Similarly, the fact that φ is a logical consequence of the set of formulas $\{\varphi_1, \ldots, \varphi_n\}$ i.e.,$\{\varphi_1, \ldots, \varphi_n\} \models \varphi$ can be encoded by the formula $\varphi_1 \wedge \cdots \wedge \varphi_n \rightarrow \varphi$

Difference

- When $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$ is an infinite set of formulas, the fact that φ is a logical consequence of Γ cannot be represented with a formula $\gamma_1 \wedge \gamma_2 \wedge \cdots \rightarrow \varphi$ because this would be infinite, and in logic all the formulas are finite. (Actually there are logics, called infinitary logics, where formulas can have infinite size.)

Logical consequence, validity and (un)satisfiability

Exercise

Show that if $\Gamma = \emptyset$*, then* $\Gamma \models \varphi \iff \varphi$ *is valid.*

Solution

($\Longrightarrow$) *Since* Γ *is empty, every interpretation* $\mathcal{I}$ *satisfies all the formulas in* Γ*. Therefore, if* $\Gamma \models \varphi$*, then every interpretation* $\mathcal{I}$ *must satisfy* φ*, hence* φ *is valid.*
($\Longleftarrow$) *If* φ *is valid, then every* $\mathcal{I}$ *is such that* $\mathcal{I} \models \varphi$*. Hence, whatever* Γ *is (in particular, when* $\Gamma = \emptyset$*), every model of* Γ *is also a model of* φ*, and so* $\Gamma \models \varphi$*.*

Exercise

Show that if φ *is unsatisfiable then* $\{\varphi\} \models \psi$ *for every formula* ψ *.*

Solution

If φ *is unsatisfiable then it has no model, which implies that each interpretation that satisfies* φ *(namely, none) satisfies also* ψ*, independently from* ψ*.*

Properties of logical consequence

Exercise

Show that the following properties hold for the logical consequence relation defined above:

Reflexivity:	$\Gamma \cup \{\varphi\} \models \varphi$
Monotonicity:	$\Gamma \models \varphi$ *implies that* $\Gamma \cup \Sigma \models \varphi$
Cut:	$\Gamma \models \varphi$ *and* $\Sigma \cup \{\varphi\} \models \psi$ *implies that* $\Gamma \cup \Sigma \models \psi$

Solution

Reflexivity: *If $\mathcal{I}$ satisfies all the formulas in $\Gamma \cup \{\varphi\}$ then it satisfies also φ, and therefore $\Gamma \cup \{\varphi\} \models \varphi$.*

Monotonicity: *Let $\mathcal{I}$ be an interpretation that satisfies all the formulas in $\Gamma \cup \Sigma$. Then it satisfies all the formulas in Γ, and if $\Gamma \models \varphi$, then $\mathcal{I} \models \varphi$. Therefore, we can conclude that $\Gamma \cup \Sigma \models \varphi$.*

Cut: *Let $\mathcal{I}$ be an interpretation that satisfies all the formulas in $\Gamma \cup \Sigma$. Then it satisfies all the formulas in Γ, and if $\Gamma \models \varphi$, then $\mathcal{I} \models \varphi$. This implies that $\mathcal{I}$ satisfies all the formulas in $\Sigma \cup \{\varphi\}$. Then, since $\Sigma \cup \{\varphi\} \models \psi$, we have that $M \models \psi$. Therefore we can conclude that $\Gamma \cup \Sigma \models \psi$.*

Outline

Checking logical consequence

Problem

Does there exist an algorithm that checks if a formula φ is a logical consequence of a set of formulas Γ?

Solution 1: If Γ is finite and the set of models of the logic is finite, then it is possible to directly apply the definition by checking for every interpretation $\mathcal{I}$, that if $\mathcal{I} \models \Gamma$ then , $\mathcal{I} \models \varphi$.

Solution 2: If Γ is infinite or the set of models is infinite, then Solution 1 is not applicable as it would run forever.

An alternative solution could be to generate, starting from Γ, all its logical consequences by applying a set of rules.

Checking logical consequence

- Propositional logic: The method based on truth tables can be used to check logical consequence by enumerating all the interpretations of Γ and φ and checking if every time all the formulas in Γ are true then φ is also true. This is possible because, when Γ is finite then there are a finite number of interpretations.
- First order logic: A first order language in general has an infinite number of interpretations. Therefore, to check logical consequence, it is not possible to apply a method that enumerates all the possible interpretations, as in truth tables.
- Modal logic: presents the same problem as first order logic. In general for a set of formulas Γ, there is an infinite number of interpretations, which implies that a method that enumerates all the interpretations is not effective.

Checking logical consequence - Deductive methods

- An alternative method for determining if a formula is a logical consequence of a set of formulas is based on inference rules.
- An inference rule is a rewriting rule that takes a set of formulas and transforms it in another set of formulas.
- The following are examples of inference rules.

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi} \qquad \frac{\varphi \quad \psi}{\varphi \rightarrow \psi} \qquad \frac{\forall x.\varphi(x)}{\varphi(c)} \qquad \frac{\exists x.\varphi(x)}{\varphi(d)}$$

- Differently from truth tables, which apply a brute force exhaustive analysis not interpretable by humans, the deductive method simulates human argumentation and provides also an understandable explanation (i.e., a deduction) of the reason why a formula is a logical consequence of a set of formulas.

Inference rules to check logical consequence Example

Let $\Gamma = \{p \rightarrow q, \neg p \rightarrow r, q \vee r \rightarrow s\}$.
The following is a deduction (an explanation of) the fact that s is a logical consequence of Γ, i.e., that $\Gamma \models s$, which uses the following inference rules:

$$\frac{\varphi \rightarrow \psi \qquad \neg\varphi \rightarrow \vartheta}{\psi \vee \vartheta}\ (*) \qquad\qquad \frac{\varphi \qquad \varphi \rightarrow \psi}{\psi}\ (**)$$

Example of deduction

(1)	$p \rightarrow q$	Belongs to Γ.
(2)	$\neg p \rightarrow r$	Belongs to Γ.
(3)	$q \vee r$	By applying $(*)$ to (1) and (2).
(4)	$q \vee r \rightarrow s$	Belongs to Γ.
(5)	s	By applying $(**)$ to (3) and (4).

Hilbert-style inference methods

In a Hilbert-style deduction system, a formal deduction is a finite sequence of formulas

$$\begin{array}{c}\varphi_1\\ \varphi_2\\ \varphi_3\\ \vdots\\ \varphi_n\end{array}$$

where each φ_i

- is either an axiom, or
- it is derived from previous formulas $\varphi_{j_1}, \dots, \varphi_{j_k}$ with $j_1, \dots, j_k < i$, by applying the inference rule

$$\frac{\varphi_{j_1}, \dots, \varphi_{j_k}}{\varphi_i}$$

Hilbert axioms for classical propositional logic

Axioms

$$
\begin{array}{ll}
\textbf{A1} & \varphi \to (\psi \to \varphi) \\
\textbf{A2} & (\varphi \to (\psi \to \theta)) \to ((\varphi \to \psi) \to (\varphi \to \theta)) \\
\textbf{A3} & (\neg\psi \to \neg\varphi) \to ((\neg\psi \to \varphi) \to \psi)
\end{array}
$$

Inference rule(s)

$$\frac{\varphi \qquad \varphi \to \psi}{\psi}\ (\mathrm{MP})$$

Example (Proof of $A \to A$)

$$
\begin{array}{lll}
1. & A1: & A \to ((A \to A) \to A) \\
2. & A2: & (A \to ((A \to A) \to A)) \to ((A \to (A \to A)) \to (A \to A)) \\
3. & MP(1,2): & (A \to (A \to A)) \to (A \to A) \\
4. & A1: & (A \to (A \to A)) \\
5. & MP(4,3): & A \to A
\end{array}
$$

Refutation

Reasoning by refutation is based on the principle of "Reductio ad absurdum".

Reductio ad absurdum

In order to show that a proposition φ is true, we assume that it is false (i.e., that $\neg\varphi$ holds) and try to infer a contradictory statement, such as $A \wedge \neg A$ (usually denoted by $\perp$, i.e., the false statement).

Reasoning by refutation is one of the most important principles for building automated decision procedures. This is mainly due to the fact that, proving a formula φ corresponds to the reduction of $\neg\varphi$ to $\perp$.

Propositional resolution

Propositional resolution is the most simple example of reasoning via refutation. The procedure can be described as follows:

Definition (Propositional resolution)

INPUT: a propositional formula φ
OUTPUT: $\models \varphi$ or $\not\models \varphi$

1. Convert $\neg\varphi$ to conjunctive normal form, i.e., to a set C of formulas (called clauses) of the form

$$p_1 \vee \cdots \vee p_k \vee \neg p_{k+1} \vee \cdots \vee \neg p_n$$

   also represented as $[p_1 \ldots, p_k, \neg p_{k+1}, \ldots, \neg p_n]$ that is logically equivalent to φ.

2. Apply exhaustively the following inference rule

$$\frac{c \vee p \quad \neg p \vee c'}{c \vee c'} \text{ (Resolution)}$$

   and add $c \vee c'$ to C

3. if C contains two clauses p and $\neg p$ then return $\models \varphi$ otherwise return $\not\models \varphi$

Inference based on satisfiability checking

In order to show that $\models \varphi$ (i.e., that φ is valid) we search for a model of $\neg\varphi$, i.e., we show that $\neg\varphi$ is satisfiable.
If we are not able to find such a model, then we can conclude that there is no model of $\neg\varphi$, i.e., that all the models satisfy φ, which is: that φ is valid.

Inference based on satisfiability checking

There are two basic methods of searching for a model for φ:

SAT based decision procedures

- This method incrementally builds a model.
- At every stage it defines a "partial model" μ_i and does an early/lazy check if φ can be true in some extension of μ_i.
- At each point the algorithm has to decide how to extend μ_i to μ_{i+1} until constructs a full model for φ.

Tableaux based decision procedures

- This method builds the model of φ via a "top down" approach.
- I.e., φ is decomposed in its sub-formulas $\varphi_1, \ldots, \varphi_n$ and the algorithm recursively builds n models $M_1, \ldots, M_n$ for them.
- The model M of φ is obtained by a suitable combination of $M_1, \ldots, M_n$

SAT based decision procedure - Example

We illustrate a SAT based decision procedure on a propositional logic example.

Example

To find a model for $(p \vee q) \wedge \neg p$, we proceed as follows:

partial model	lazy evaluation	result of lazy evaluation
$\mu_0 = \{p = true\}$	$(true \vee q) \wedge \neg true$	$false \quad (backtrack)$
$\mu_1 = \{p = false\}$	$(false \vee q) \wedge \neg false$	$q \quad (continue)$
$\mu_2 = \{p = false, q = true\}$	$(false \vee true) \wedge \neg false$	$true \quad (success!)$

Soundness and Completeness

Let R be an inference method, and let '$\vdash_R$ denote the corresponding inference relation.

Definition (Soundness of an inference method)

An inference method R is sound if

$$\begin{array}{rcll} \vdash_R \varphi & \Longrightarrow & \models \varphi & \\ \Gamma \vdash_R \varphi & \Longrightarrow & \Gamma \models \varphi & \text{(strongly sound)} \end{array}$$

Definition (Completeness of an inference method)

An inference method R is complete if

$$\begin{array}{rcll} \models \varphi & \Longrightarrow & \vdash_R \varphi & \\ \Gamma \models \varphi & \Longrightarrow & \Gamma \vdash_R \varphi & \text{(strongly complete)} \end{array}$$

Clausal form: dealing with quantifiers

Clausal form as before, but atom is $P(t_1, t_2, \ldots, t_n)$, where t_i may contain variables
Interpretation as before, but variables are understood *universally*

- Example: $\{[P(x), \neg R(a, f(b, x))], [Q(x, y)]\}$
  - interpreted as
- $\forall x \forall y \{[R(a, f(b, x)) \supset P(x)] \wedge Q(x, y)\}$

Substitutions: $\theta = \{\nu_1/t_1, \nu_2/t_2, \ldots, \nu_n/t_n\}$
Notation: If ρ is a literal and θ is a substitution, then $\rho\theta$ is the result of the substitution (and similarly, $c\theta$ where c is a clause)

- Example: $\theta = \{x/a, y/g(x, b, z)\}$
- $P(x, z, f(x, y))\theta = P(a, z, f(a, g(x, b, z)))$

A literal is ground if it contains no variables.
A literal ρ is an instance of ρ', if for some $\theta, \rho = \rho'\theta$

Generalizing CNF

Resolution will generalize to handling variables
But, to convert wffs to CNF, we need additional steps:

1. eliminate $\supset$ and $\equiv$
2. push $\neg$ inward using also $\neg\forall x.\alpha \Rightarrow \exists x.\neg\alpha$ etc.
3. standardize variables: each quantifier gets its own variable
   - e.g. $\exists x[P(x)] \wedge Q(x) \Rightarrow \exists z[P(z)] \wedge Q(x)$ where z is a new variable
4. eliminate all existentials (*discussed later*)
5. move universals to the front using $(\forall x \alpha) \wedge \beta \Rightarrow \forall x(\alpha \wedge \beta)$
   - where β does not use x
6. distribute $\wedge$ over $\vee$

Get universally quantified conjunction of disjunction of literals

- then drop all the quantifiers...

First-order resolution

Main idea: a literal (with variables) stands for all its instances; so allow all such inferences

- So given $[P(x,a), \neg Q(x)]$ and $[\neg P(b,y), \neg R(b,f(y))]$, want to infer $[\neg Q(b), \neg R(b,f(a))]$ among others
  - since $[P(x,a), \neg Q(x)]$ has $[P(b,a), \neg Q(b)]$ and $[\neg P(b,y), \neg R(b,f(y))]$ has $[\neg P(b,a), \neg R(b,f(a))]$

Resolution:

- Given clauses: $\{\rho_1\} \cup C_1$ and $\{\overline{\rho_2}\} \cup C_2$.
- Rename variables, so that distinct in two clauses.
- For any θ such that $\rho_1\theta = \rho_2\theta$, can infer $(C_1 \cup C_2)\theta$.
  - We say that ρ_1 unifies with ρ_2 and that θ is a unifier of the two literals

Resolution derivation: as before

Theorem: $S \rightarrow []$ iff $S \models []$ iff S is unsatisfiable

- Note: There are pathological examples where a slightly more general definition of Resolution is required. We ignore them for now...

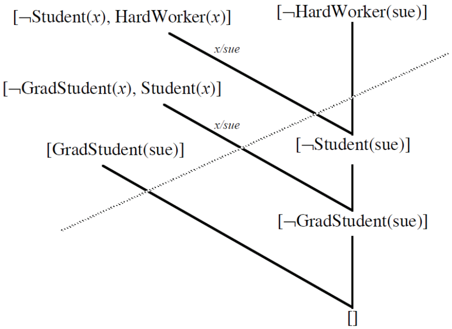
Example 3

?

$KB \models HardWorker(sue)$

KB

$\forall x \quad GradStudent(x) \supset Student(x)$
$\forall x \quad Student(x) \supset HardWorker(x)$
$GradStudent(sue)$



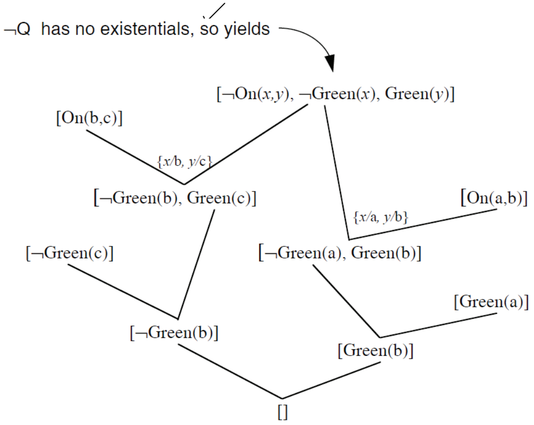
- Label each step with the unifier
- Point to relevant literals in clauses

The 3 block example

$KB = \{On(a,b), On(b,c), Green(a), \neg Green(c)\}$

- already in CNF

$Query = \exists x \exists y [On(x,y) \wedge Green(x) \wedge \neg Green(y)]$



- Note: Need to use $On(x,y)$ twice, for 2 cases

Arithmetic

KB: $Plus(zero, x, x)$

$Plus(x, y, z) \supset Plus(succ(x), y, succ(z))$

Q: $\exists u Plus(2, 3, u)$

For readability, we use

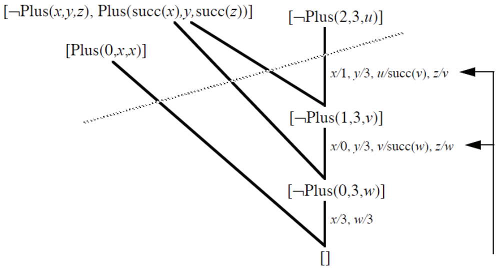
- 0 for $zero$,
- 1 for $succ(zero)$,
- 2 for $succ(succ(zero))$
- etc.

Can find the answer in the derivation

- $u/succ(succ(3))$

that is: $u/5$

Can also derive $Plus(2, 3, 5)$



Answer predicates

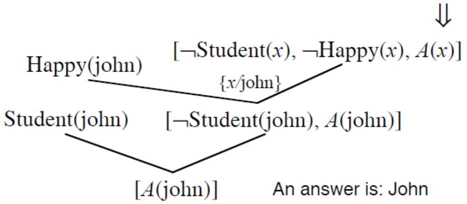
In full FOL, we have the possibility of deriving $\exists x P(x)$ without being able to derive $P(t)$ for any t.

- e.g. the three-blocks problem
  - $\exists x \exists y [On(x, y) \wedge Green(x) \wedge \neg Green(y)]$
  - but cannot derive which block is which

Solution: answer-extraction process

- replace query $\exists x P(x)$ by $\exists x [P(x) \wedge \neg A(x)]$
  - where A is a new predicate symbol called the answer predicate
- instead of deriving [], derive any clause containing just the answer predicate
- can always convert to and from a derivation of []

KB : $Student(john)$
$Student(jane)$
$Happy(john)$
Q :
$\exists x [Student(x) \wedge Happy(x)]$



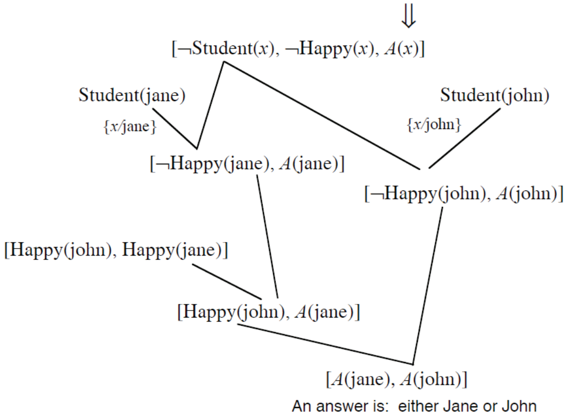
Disjunctive answers

KB :

- $Student(john)$
- $Student(jane)$
- $Happy(john) \vee Happy(jane)$

$Query$:

- $\exists x[Student(x) \wedge Happy(x)]$



Note:

- can have variables in answer
- need to watch for Skolem symbols... (next)

Skolemization

So far, converting wff to CNF ignored existentials

- e.g. $\exists x \forall y \exists z P(x, y, z)$

Idea: names for individuals claimed to exist, called Skolem constant and function symbols

- there exists an x, call it a
- for each y, there is a z, call it $f(y)$
  - get $\forall y P(a, y, f(y))$

replace $\forall x_1(\ldots \forall x_2(\ldots \forall x_n(\ldots \exists y[\ldots y \ldots] \ldots) \ldots) \ldots)$ by
$\forall x_1(\ldots \forall x_2(\ldots \forall x_n(\ldots [\ldots f(x_1, x_2, \ldots, x_n) \ldots] \ldots) \ldots) \ldots)$

- f is a new function symbol that appears nowhere else

Skolemization does not preserve equivalence

- e.g. $\not\models \exists x P(x) \equiv P(a)$

But it does preserve satisfiability

- α is satisfiable iff α' is satisfiable (where α' is the result of Skolemization) sufficient for resolution!

Variable dependence

Show that $\exists x \forall y R(x,y) \models \forall y \exists x R(x,y)$

- show $\{\exists x \forall y R(x,y), \neg \forall y \exists x R(x,y)\}$ unsatisfiable
  - $\exists x \forall y R(x,y) \Rightarrow \forall y R(a,y)$
  - $\neg \forall y \exists x R(x,y) \Rightarrow \exists y \forall x \neg R(x,y) \Rightarrow \forall x \neg R(x,b)$
- then $\{[R(a,y)], [\neg R(x,b)]\} \rightarrow []$ with $\{x/a, y/b\}$.

Show that $\forall y \exists x R(x,y) \not\models \exists x \forall y R(x,y)$

- show $\{\forall x \exists x R(x,y), \neg \exists x \forall y R(x,y)\}$ satisfiable
  - $\forall y \exists x R(x,y) \Rightarrow \forall y R(f(y),y)$
  - $\neg \exists x \forall y R(x,y) \Rightarrow \forall x \exists y \neg R(x,y) \Rightarrow \forall x \neg R(x, g(x))$
- then get $\{[R(f(y),y)], [\neg R(x,g(x))]\}$
  - where the two literals do <u>not</u> unify

Note: important to get dependence of variables correct

- $R(f(y),y) vs. R(a,y)$ in the above

A problem

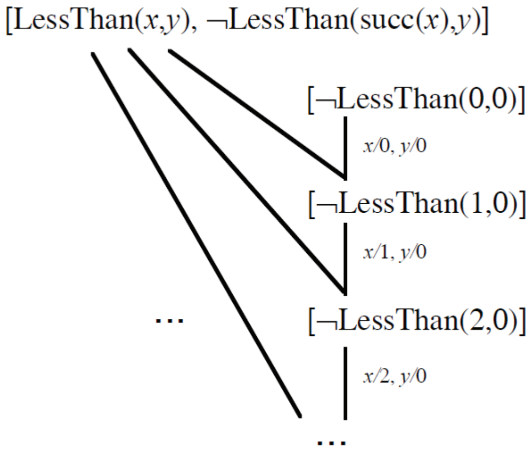
KB :

- $LessThan(succ(x), y) \supset LessThan(x, y)$

$Query$:

- $LessThan(zero, zero)$

Should fail since $KB \not\models Q$



- Infinite branch of resolvents cannot use a simple depth-first procedure to search for []

Undecidability

Is there a way to detect when this happens?
No! FOL is very powerful

- can be used as a full programming language
- as there is no way to detect in general when a program is looping

There can be no procedure that does this:

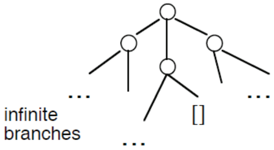
- $Proc[Clauses] =$
- If $Clauses$ are unsatisfiable
    - then return YES
    - else return NO

However: Resolution is complete

- some branch will contain [], for unsatisfiable clauses

So breadth-first search guaranteed to find []

- search may not terminate on satisfiable clauses


infinite branches
. . .
[]
. . .
. . .