Knowledge Representation: Logic

Wojciech Jaworski

Institute of Informatics
University of Warsaw
Expressing knowledge of certain domain in an explicit, organized and computer-readable manner.

Knowledge Representation is making use of logic and ontology to construct computable models for given domains.
  - **Logic** provides the formal structure and rules of inference
  - **Ontology** defines the kinds of things
  - **Computable models** (e.g. relational database model) facilitate the implementation of logic and ontology in computers.

Application examples
  - database designing
  - object methods
  - natural language processing
  - knowledge engineering
1. Knowledge Representation in logic

2. Varieties of Logic

3. Names, Types, and Measures
Logic gives processable form to all the information that can be precisely expressed in any other language.

Logic allows to express all the information that can be stored in computer memory.

If some knowledge is not logic-conveyable it cannot be processed by computers no matter the notation.
Every car has 4 wheels.

Representation in propositional logic:

\[ p \]

- the simplest representation form
- no details about the cars, the wheels, the number 4 and their interrelationships

The loss of details may be an advantage in some applications.
Let us consider the following syllogism (ancient inference rule):

- Every car has 4 wheels.
- Some Corvettes are cars.
- Therefore, some corvettes have 4 wheels.

We can express it in logic in the following way:

\[(\forall x)(\text{car}(x) \Rightarrow \text{fourWheeler}(x))\]
\[(\exists x)(\text{Corvette}(x) \land \text{car}(x))\]
\[(\exists x)(\text{Corvette}(x) \land \text{fourWheeler}(x))\]

Notice that sentences of the type *Every A is B* are translated to

\[\forall x(A(x) \Rightarrow B(x))\]

and sentences of the type *Some A are B* are translated to

\[\exists x(A(x) \land B(x))\]
The predicate "fourWheeler" doesn’t convey the number 4 nor its relation to wheels.

We can replace it with a two-place predicate:

\[ \text{numberOfWheeles}(x, n), \]

which means "the number of wheels of } x \text{ is } n \text{ or } } x \text{ has } n \text{ wheels}".

The sentence *Every car has 4 wheels* has the logical form

\[ (\forall x)(\text{car}(x) \Rightarrow \text{numberOfWheeles}(x, 4)) \]

which may be read as *For every } x, \text{ if } x \text{ is a car, the number of wheels of } x \text{ is } 4*
In the predicate

\[ \text{numberOfWheeles}(x, n) \]

\(x\) refers to cars, \(n\) refers to numbers but no variable refers to wheels.

The names of the predicates are but meaningless labels for the computer system.

If the notion of a wheel is important for some application, a more detailed selection of predicates is necessary.
Choice of Predicates

Let us consider the predicates:

- `car(x)`  
  `x` is a car
- `wheel(x)`  
  `x` is a wheel
- `part(x, y)`  
  `y` is a part of `x`
- `set(s)`  
  `s` is a set
- `count(s, n)`  
  The count of elements in `s` is `n`
- `member(x, s)`  
  `x` is a member of the set `s`

The car formula then becomes:

\[
(\forall x)(\text{car}(x) \Rightarrow (\exists s)\,(\text{set}(s) \land \text{count}(s, 4) \\
\land (\forall w)(\text{member}(w, s) \Rightarrow (\text{wheel}(w) \land \text{part}(x, w))))))
\]

the formula may be read:

*For every* `x`, *if* `x` *is a car, then there exists* `s`, *where* `s` *is a set of count 4 and for every* `w`, *if* `w` *is a member of* `s`, *then* `w` *is a wheel and* `w` *is a part of* `x`.  

Wojciech Jaworski (MIM UW) Logic 9 / 36
The formula

\[(\exists s)(\text{set}(s) \land \text{count}(s, 4) \land (\forall w)(\text{member}(w, s) \Rightarrow (\text{wheel}(w) \land \text{part}(x, w)))))]

describes the property of having 4 wheels.

We can use it to define the predicate fourWheeler\((x)\):

\[\text{fourWheeler}(x) = (\exists s)(\text{set}(s) \land \text{count}(s, 4) \land (\forall w)(\text{member}(w, s) \Rightarrow (\text{wheel}(w) \land \text{part}(x, w))))]

A definition is a notational short-cut and it corresponds to a subroutine or a macro in programming languages.

\(x\) — the free variable in the defining formula — is called the formal parameter.

While expanding a definition we use \(\alpha\)-conversion: renaming local variables to unique ones.
Even simple sentences of a natural language convey much hidden information.

Giving all the details leads to difficulties in finding the key information while reading.

The overflow of the details is partially caused by making all the variables explicit.

The level of detail depends on the choice of predicates, i.e. the choice of an ontology.

The predicates in an ontology may be divided in two classes:

- the domain-dependent predicates: car, wheel(x)
- the domain-independent predicates: part(x, y), set(s), count(s, n), member(x, s)
Special languages

- Subsets of logic with their own notation and built-in ontology.
- Examples: musical notation, timetables, maps, plans, schemes.
- Shorter and more readable than logic (and natural language).
- Logic is an ontologically neutral notation that can be adapted to any subject by adding one or more domain-dependent predicates.
- Special languages may be translated to logical formulas in order to automatically process the content they express.
Example: A map
Example: A map

- Streets may be represented as segments of straight lines and arcs.
- Streets intersect at cross-roads and make a graph.
- Street names are edge labels in the graph.
- There are also points such as restaurants, bus stops etc.
- A structural implementation of the map would be done by creating a graph of streets etc. Adding a new component to the map requires some modifications to be introduced to the data structure, so the code must be rewritten.
- The elements of the map could be associated with object classes divided into point-like, line-like and so on. Addition of a new component may be then achieved by adding a new subclass, but it can be impossible, for example for street names.
- We may as well express the content of the map using logic and add new components by introducing new predicates.
Existential-Conjunctive Logic

- Allows to convey facts about particular objects.
- Suffices as the representation for most of the special languages.
- Represents the information stored in both relational and object-oriented database systems.
- It cannot represent any generalizations, negations, implications, or alternatives.
- An example of generalization: The rule *A bus stop must be located on a street.*
Table of contents

1. Knowledge Representation in logic

2. Varieties of Logic

3. Names, Types, and Measures
Varieties of Logic

Logics can differ along the following dimensions

- **Syntax**: Influences readability, doesn’t change the expressive power
- **Subset**: Possible operators and combinations of operators, e.g. the logic $\exists \land$ or propositional calculus
- **Proof theory**: Restrictions on the permissible proofs (e.g. intuitionistic logic, linear logic), or extensions (e.g. non-monotonic logics); In that cases the semantics follows from the proof theory — opposite to FOL.
- **Model theory**: Modifications of the notion of truth, e.g. three-valued logic or fuzzy logic.
- **Ontology**: the set of predicates and axioms we take for basic and domain-independent, e.g. $\equiv$, set theory, time theory
- **Metalanguage**: The language that is used to talk about another language (e.g. FOL, context-free grammar)
We label variables with types:

\[(\forall x : t)P(x) \equiv (\forall x)(t(x) \Rightarrow P(x))\]

\[(\exists x : t)P(x) \equiv (\exists x)(t(x) \land P(x))\]

For example:

*Every car has 4 wheels.*

\[(\forall x : \text{car})\text{fourWheeler}(x)\]

*Some Corvettes are cars.*

\[(\exists x : \text{Corvette})\text{car}(x)\]

*Therefore, some Corvettes have 4 wheels.*

\[(\exists x : \text{Corvette})\text{fourWheeler}(x)\]
We can also use special notation for domain-independent predicates.

The sentence

\[(\forall x)(\text{car}(x) \Rightarrow (\exists s)(\text{set}(s) \land \text{count}(s, 4) \land (\forall w)(\text{member}(w, s) \Rightarrow (\text{wheel}(w) \land \text{part}(x, w))))))\]

can be written as

\[(\forall x : \text{car})(\exists s : \text{set})(s@4 \land (\forall w \in s)(\text{wheel}(w) \land \text{part}(x, w))))\]
Defining new functions and function closures

Instead of

\[
\text{fourWheeler}(x) = (\exists s : \text{set})(s@4 \land (\forall w \in s)(\text{wheel}(w) \land \text{part}(x, w)))
\]

we can write

\[
\text{fourWheeler} = \lambda x ((\exists s : \text{set})(s@4 \land (\forall w \in s)(\text{wheel}(w) \land \text{part}(x, w))))
\]

The notation \( \lambda x \Psi(x) \) is a definition of a function closure, where \( x \) is the formal parameter and \( \Psi \) is the formula that defines the function.

The application \( (\lambda x \Psi(x))(a) \) generates the formula \( \Psi(a) \).

\( \lambda \) will be used among others for defining the function closures.

\[
(\exists x : \text{fourWheeler}) \text{car}(x)
\]

\[
(\exists x : (\lambda x)((\exists s : \text{set})(s@4 \land (\forall w \in s)(\text{wheel}(w) \land \text{part}(x, w)))))) \text{car}(x)
\]
Every car has 4 wheels.

- A notation for logic that eliminates variables
- The boxes are called concepts and circles are called conceptual relations.
- Inside the box is the concept description (a label or a definition) and a colon-separated description of reference which consists of either a label, a quantifier $\forall$ or a specification of quantity.
- The relations show how the denotations are associated.
Conceptual Graphs Semantics

- Every box is labeled with a unique natural number.
- A FOL formula is composed step-by-step from the labelled boxes in increasing order:
  - If you see $\text{Concept}_i$, append $(\exists x_i : \text{concept})$ to the formula.
  - $\text{Concept}_i : \forall$ translates to $(\forall x_i : \text{concept})$
  - $\text{Concept}_i : \{\star\}@n$ translates to $(\exists s_i : \text{set})s_i@n \land (\forall x_i \in s_i)\text{concept}(x_i)$

where $i$ is the label of the box.

- Process the circles in any order. For the relations of the form

  ![Diagram of relations]

  generate $\land\text{relation}(x_i, x_j)$ and append to the formula (omit „∧” at the first occurrence)
If we’re restricted to Existential-Conjunctive Logic, no order of predicates and quantifiers does matter, so the translation from a graph to a FOL formula is unambiguous.

If $\forall$ occurs, the translation is ambiguous.

For example

\[
\text{Boy : } \forall \xrightarrow{\text{Like}} \text{Girl}
\]

can be interpreted as

\[
(\forall y : \text{boy})(\exists x : \text{girl})\text{like}(y, x)
\]

or as

\[
(\exists x : \text{girl})(\forall y : \text{boy})\text{like}(y, x)
\]

The same phenomenon occurs in natural languages (*Quantifier Scope Ambiguity*).
Every car ma 4 wheels.

\((\forall x : \text{car}) \text{fourWheeler}(x)\)

- In the example above two concepts have the same referent.
- We can express it in a graph as:

  ![Graph](#)

  Which corresponds to the formula:
  
  \((\forall x : \text{car})(\exists y : \text{fourWheeler}) x = y\)

  We can also use the shorthand notation:

  ![Shorthand Notation](#)
Knowledge Interchange Format (KIF)

- Typed logic with Lisp-style syntax and restricted to ASCII symbols.
- Example
  
  ```
  (forall (?x car)
    (exists (?s set)
      (and (count ?s 4)
        (forall (?w in ?s) (and (wheel ?w)
          (part ?x ?w))))))
  ```
Modal Logics

- Logic with operators that express modality.
- $\Diamond \varphi$ means that $\varphi$ is possible.
- $\Box \varphi$ means that $\varphi$ is necessary.

Modal operators semantics can be summarized as follows:
- we have a set of possible worlds one of which is designated as the actual world;
- $\varphi$ means that $\varphi$ is true in the actual world;
- $\Diamond \varphi$ means that there exists a world where $\varphi$ is true;
- $\Box \varphi$ means that $\varphi$ is true in every possible world.

- Different axiomatizations of modality differ in their expressive power.
- In database theory
  - any statement that is stored in the database is assumed true in the actual world
  - constraints are taken to be necessary true.
Higher order logics

- First order logic allows for quantification over individuals.
- Second order logic allows for quantification over relations on individuals.
- Third order logic allows for quantification over relations on relations on individuals etc. ...
- For example, the axiom of induction for natural numbers:

\[(\forall P : \text{Predicate})((P(0) \land (\forall n \in \mathbb{N})(P(n) \Rightarrow P(n+1))) \Rightarrow (\forall n \in \mathbb{N})P(n))\]

- Higher order logics may be defined as FOL with an ontology for relations (eg. ZFC)
Metalanguage

- A language used to talk about a language.
- Example: Natural language when used to define first order logic.
- Let us consider the sentence: *The sentence „It is true that John is tall” means the same as „John is tall”*
- Quotation marks indicate the use of metalanguage, the word „true” as well.
- Thus, whole the utterance belongs to the metametalanguage and the slide that treats about it - to the metametametalanguage.
- We can introduce a special predicate into a logic to play the role of quotation marks and to connect a language with it’s metalanguage.
The expression *Fluffy is a cat* may be represented as

\[
\text{cat(Fluffy)}
\]

This representation is simple but fallible, as
- the name is ambiguous, e.g. many cats bear the name „Fluffy”.
- one entity bears many names, e.g. a cat called „Fluffy” by one family is known as „Mittens” to another.

The solution are **surrogates**, i.e. unique identifiers for individuals. Surrogates for Polish citizens are their PESEL numbers.

Proper names become plain attributes of objects:

\[
\text{Name(„Fluffy”) } \land (\exists x : \text{Cat}) \text{hasName}(x, „Fluffy”)
\]
Unique Name Assumption

- If two constants $a$ and $b$ have different names then $a \neq b$.
- KIF has two types of constants with uniqueness assumed for one of the types but not for the other.
- The constants in conceptual graphs are not assumed unique but symbols of the form #number are. They serve as surrogates.
- A node of the form `Concept : Name` is represented in logic as

  \[ \text{Concept(Name)} \]

  and means that an object named "Name" is of type "Concept".
- `Name : "Fluffy"` is the name "Fluffy".
- `Cat : #2563` is a cat under the record number #2563.
Skolemization

If we introduce a surrogate for each individual of which we know then we can discard all $\exists$ and substitute for all the occurrences of bound variables their surrogates.

The process is an existential instantiation, also called skolemization.

Skolemization changes the meaning slightly, because an existentially quantified variable doesn’t uniquely indicate an object.

- If we know that

$$(\exists x)\text{cat}(x)$$

then we can answer the question „What is the color of this cat’s fur?” by giving the color of any existing cat.

- But if our knowledge is that

$$\text{cat}(\#2563)$$

then the question „What is the color of this cat’s fur?” must be answered with the color of the fur of the object indicated by $\#2563$. 

Wojciech Jaworski (MIM UW)
The sentence *Cats like fish* can be represented in logic by

\[(\forall x : \text{Cat})(\forall y : \text{Fish})\text{like}(x, y)\]

And *A cat likes a fish* can be represented as

\[(\exists x : \text{Cat})(\exists y : \text{Fish})\text{like}(x, y)\]

The correlates of common names are types or predicates.
First order logic allows to quantify over individuals only. In order to quantify over types we introduce the type „Type”. The variables of type „Type” are types. „Type” is a type of second order.

*Cats are mammals. The cat is a mammal.*

\[(\forall x : \text{Cat})\text{mammal}(x)\]

*Cat is a specie.*

\[(\exists x : \text{Specie})(\text{type(Cat)} \land x = \text{Cat})\]

*Fluffy is a cat.*

\[(\exists x : \text{Cat})x = \text{Fluffy}\]

*Fluffy the cat.*

\[\text{cat(Fluffy)}\]
We introduce the predicate \( \text{kind}(x, t) \) which means that object \( x \) is of type \( t \).

Using the predicate „kind” to give types is a process called reification:

\[
(\forall x)(\forall y)(\text{kind}(x, \text{Cat}) \land \text{kind}(y, \text{Fish})) \Rightarrow \text{like}(x, y)
\]

The names of types may have aliases just as the names of individuals can, so in practice surrogates are used for the names of types as well.

Now we can define non-empty types by stating that Every non-empty type contains at least one individual:

\[
(\forall t : \text{Non-EmptyType})(\exists x : \text{Individual})\text{kind}(x, t)
\]
Representing Measures

Tom and Sue each earn a salary whose amount is $30.000.

\[(\exists x, y : \text{Earn})(\exists z, w : \text{Salary})\]

\[\text{(Person}(\text{Tom}) \land \text{Person}(\text{Sue}) \land \]
\[\text{agnt}(x, \text{Tom}) \land \text{thme}(x, z) \land \text{amount}(z, 30000\$) \land \]
\[\text{agnt}(y, \text{Sue}) \land \text{thme}(y, w) \land \text{amount}(w, 30000\$))\]

- The formula doesn’t make it clear that the two salaries are distinct, as indicated by the word „each”.
- This can be conveyed in the following ways:
  - append the condition \(z \neq w\) to the formula
  - introduce surrogates for the salaries and establish the unique naming convention.