# Artificial Intelligence

# Predicate Logic

Ioan Toma

ioan.toma@sti2.at

# Where are we?

| # | Title |
|---|---|
| 1 | Introduction |
| 2 | Propositional Logic |
| **3** | **Predicate Logic** |
| 4 | Reasoning |
| 5 | Search Methods |
| 6 | CommonKADS |
| 7 | Problem-Solving Methods |
| 8 | Planning |
| 9 | Software Agents |
| 10 | Rule Learning |
| 11 | Inductive Logic Programming |
| 12 | Formal Concept Analysis |
| 13 | Neural Networks |
| 14 | Semantic Web and Services |

- Motivation
- Technical Solution
  - Syntax
  - Semantics
  - Inference
- Illustration by Larger Example
- Extensions
- Summary
- References

# MOTIVATION

- Suppose we want to capture the knowledge that

  *Anyone standing in the rain will get wet.*

  and then *use* this knowledge. For example, suppose we also learn that

  *Jan is standing in the rain.*

- We'd like to conclude that Jan will get wet. But each of these sentences would just be a represented by some proposition, say *P, Q* and *R*. What relationship is there between these propositions? We can say

  $P \wedge Q \rightarrow R$

  Then, given $P \wedge Q$, we could indeed conclude *R*. But now, suppose we were told

  *Pat is standing in the rain.*

- We'd like to be able to conclude that Pat will get wet, but nothing we have stated so far will help us do this

- The problem is that we aren't able to represent any of the details of these propositions
  - It's the **internal structure** of these propositions that make the reasoning valid.
  - But in propositional calculus we don't have anything else to talk about besides propositions!

$\Rightarrow$ A more expressive logic is needed
  - $\Rightarrow$ **Predicate logic** (occasionally referred to as **First-order logic (FOL)**)

# TECHNICAL SOLUTIONS

Syntax

- Constants
  - Names of specific objects
  - E.g. doreen, gord, william, 32

- Functions
  - Map objects to objects
  - E.g. father(doreen), age(gord), max(23,44)

- Variables
  - For statements about unidentified objects or general statements
  - E.g. x, y, z, …

- Terms represent objects
- The set of terms is inductively defined by the following rules:
  - Constants: Any constant is a term
  - Variables: Any variable is a term
  - Functions: Any expression *f(t1,…,tn)* of *n* arguments (where each argument is a term and *f* is a function of arity *n*) is a term
- Terms without variables are called ground terms
- Examples:
  - c
  - f(c)
  - g(x,x)
  - g(f(c), g(x,x))

- Predicate symbols represent relations between zero or more objects

- The number of objects define a predicate's aritiy

- Examples:
  - Likes(george, kate)
  - Likes(x,x)
  - Likes(joe, kate, susy)
  - Friends (father_of(david), father_of(andrew))

- Signature: A signature is a collection of constants, function symbols and predicate symbols with specified arities

- FOL formulas are joined together by logical operators to form more complex formulas (just like in propositional logic)

- The basic logical operators are the same as in propositional logic as well:
  - Negation: ¬p („it is not the case that p")
  - Conjunction: p ∧ q („p and q")
  - Disjunction: p ∨ q („p or q")
  - Implication: p → q („p implies q" or "q if p")
  - Equivalence: p ↔ q („p if and only if q")

- Two quantifiers: Universal (∀) and Existential (∃)
- Allow us to express properties of collections of objects instead of enumerating objects by name
  - Apply to sentence containing variable
- Universal ∀: true for **all** substitutions for the variable
  - "for all": ∀<variables> <sentence>
- Existential ∃: true for **at least one** substitution for the variable
  - "there exists": ∃<variables> <sentence>
- Examples:
  - ∃ x: Mother(art) = x
  - ∀ x ∀ y: Mother(x) = Mother(y) → Sibling(x,y)
  - ∃ y ∃ x: Mother(y) = x

- The set of formulas is inductively defined by the following rules:
  1. **Preciate symbols**: If P is an n-ary predicate symbol and $t_1,\ldots,t_n$ are terms then $P(t_1,\ldots,t_n)$ is a formula.
  2. **Negation**: If φ is a formula, then ¬φ is a formula
  3. **Binary connectives**: If φ and ψ are formulas, then (φ → ψ) is a formula. Same for other binary logical connectives.
  4. **Quantifiers**: If φ is a formula and x is a variable, then ∀xφ and ∃xφ are formulas.

- Atomic formulas are formulas obtained only using the first rule
- Example: If f is a unary function symbol, P a unary predicate symbol, and Q a ternary predicate symbol, then the following is a formula:

$$\forall x \forall y (P(f(x)) \rightarrow \neg (P(x)) \rightarrow Q(f(y), x, x)))$$

- Any occurrence of a variable in a formulate not in the scope of a quantifier is said to be a free occurrence
- Otherwise it is called a bound occurrence
- Thus, if x is a free variable in φ, it is bound in ∀xφ and ∃xφ
- A formula with no free variables is called a closed formula
- Example: x and y are bound variables, z is a free variable

$$\forall x \forall y (P(f(x)) \rightarrow \neg(P(x)) \rightarrow Q(f(y), x, z)))$$

S := <Sentence>
<Sentence> :=      <AtomicSentence>
                  | <Sentence> <Connective> <Sentence>
                  | <Quantifier> <Variable>,... <Sentence>
                  | ¬ <Sentence>
                  | ( <Sentence> )
<AtomicSentence> := <Predicate> ( <Term>, ... )
<Term> :=          <Function> ( <Term>, ... )
                  | <Constant>
                  | <Variable>
<Connective> :=  ∧ | ∨ | → | ↔
<Quantifier> := ∃ | ∀
<Constant> := "c" | "x1" | "john" | ...
<Variable> := "a" | "x" | "s" | ...
<Predicate> := "before" | "hasColor" | "raining" | ...
<Function> := "mother" | "leftLegOf" | ...

# TECHNICAL SOLUTIONS

Semantics

- Interpretations
- Models and Satisfiability
- Logical Consequence (Entailment)

- Interpretation – Maps symbols of the formal language (predicates, functions, variables, constants) onto objects, relations, and functions of the "world" (formally: Domain, relational Structure, or Universe)

- Valuation – Assigns domain objects to variables
  - The Valuation function can be used for describing value assignments and constraints in case of nested quantifiers.
  - The Valuation function otherwise determines the satisfaction of a formula only in case of open formulae.

- Constructive Semantics – Determines the semantics of complex expressions inductively, starting with basic expressions

Domain, relational Structure, Universe

| D | finite set of Objects | $d_1, d_2, ... , d_n$ |
|---|---|---|
| R,... | Relations over D | $R \subseteq D^n$ |
| F,... | Functions over D | $F: D^n \rightarrow D$ |

Basic Interpretation Mapping

| constant | $I$ [c] = d | Object |
|---|---|---|
| function | $I$ [f] = F | Function |
| predicate | $I$ [P] = R | Relation |

Valuation V

variable $V(x) = d \in D$

Next, determine the semantics for complex terms and formulae constructively, based on the basic interpretation mapping and the valuation function above.

Terms with variables

$I[f(t_1,...,t_n)) = I[f](I[t_1],..., I[t_n]) = F(I[t_1],..., I[t_n]) \in D$

where   $I[t_i] = V(t_i)$   if $t_i$ is a variable

Atomic Formula

$I[P(t_1,...,t_n)]$         true if $(I[t_1],..., I[t_n]) \in I[P] = R$

Negated Formula

$I[\neg\alpha]$         true if $I[\alpha]$ is not true

Complex Formula

$I[\alpha\vee\beta]$         true if $I[\alpha]$ or $I[\beta]$ true

$I[\alpha\wedge\beta]$         true if $I[\alpha]$ and $I[\beta]$ true

$I[\alpha\rightarrow\beta]$         if $I[\alpha]$ not true or $I[\beta]$ true

Quantified Formula          (relative to Valuation function)

$I$ [$\exists$x:$\alpha$]                    true if $\alpha$ is true with V'(x)=d for some d$\in$D
                          where V' is otherwise identical to the prior V.

$I$ [$\forall$x:$\alpha$]                    true if $\alpha$ is true with V'(x)=d for all d$\in$D and

                          where V' is otherwise identical to the prior V.

**Note:** $\forall$x$\exists$y:$\alpha$ is different from  $\exists$y$\forall$x:$\alpha$

In the first case $\forall$x$\exists$y:$\alpha$ , we go through all value assignments V'(x), and for each value assignment V'(x) of x, we have to find a suitable value V'(y) for y.

In the second case $\exists$y$\forall$x:$\alpha$, we have to find one value V'(y) for y, such that all value assignments V'(x) make $\alpha$ true.

Given is an interpretation $I$ into a domain D with a valuation V, and a formula $\varphi$.

We say that:

$\varphi$ is **satisfied** in this interpretation      or

this interpretation is a **model** of $\varphi$      iff

$I\,[\varphi]$ is true.

That means the interpretation function $I$ into the domain D (with valuation V) makes the formula $\varphi$ true.

Given a set of formulae $\Phi$ and a formula **α.**

   **α** is a **logical consequence** of $\Phi$     iff

   **α** is true in every model in which $\Phi$ is true.

Notation:

$$\Phi \vDash \boldsymbol{\alpha}$$

That means that for every model (interpretation into a domain) in which $\Phi$ is true, **α** must also be true.

# TECHNICAL SOLUTIONS

Inference

- ## Entailment: KB ⊨ Q

  - Entailment is a relation that is concerned with the semantics of statements

  - Q is entailed by KB (a set of premises or assumptions) if and only if there is no logically possible world in which Q is false while all the premises in KB are true

  - Stated positively: Q is entailed by KB if and only if the conclusion is true in every possible world in which all the premises in KB are true

- ## Provability: KB ⊢ Q

  - Provability is a syntactic relation

  - We can derive Q from KB if there is a proof consisting of a sequence of valid inference steps starting from the premises in KB and resulting in Q

- Soundness: If KB $\vdash$ Q then KB $\vDash$ Q
  - If Q is derived from a set of sentences KB using a set of inference rules, then Q is entailed by KB
  - Hence, inference produces only real entailments, or any sentence that follows deductively from the premises is valid
  - Only sentences that logically follow will be derived

- Completeness: If KB $\vDash$ Q then KB $\vdash$ Q
  - If Q is entailed by a set of sentences KB, then Q can also be derived from KB using the rules of inference
  - Hence, inference produces all entailments, or all valid senteces can be proved from the premises
  - All the sentences that logically follow can be derived

- Inference rules for propositional logic apply to predicate logic as well
  - Modus Ponens, Modus Tollens etc.
- New (sound) inference rules for use with quantifiers:
  - Modus Ponens, Modus Tollens
  - Universal elimination
  - Existential elimination
  - Existential introduction
  - Generalized Modus Ponens (GMP)

- **Modus Ponens (Law of Detachment )**
  - Based on claiming 1.) that P is true, and 2.) the implication $P \rightarrow Q$, we can conclude that Q is true.
  - If P, then Q. P, therefore, Q

  $$\frac{P \rightarrow Q, P}{Q}.$$

  - Example:

    RegularlyAttends(joe,  lecture),

    RegularlyAttends(joe, lecture) $\rightarrow$ Pass(joe, lecture)

    Allow us to conclude:

    Pass(joe, lecture)

STI · INNSBRUCK

- **Modus Tollens (Denying the consequent)**
  - We again make two claims. 1.) The implication $P \to Q$, and 2.) that $Q$ is false. We can conclude that P is false as well.
  - If $P$, then $Q$. $\neg Q$ Therefore, $\neg P$

$$\frac{P \to Q \,, \quad \neg Q}{\neg P}$$

  - Example:

    Detects(alarm, intruder) $\to$ GoesOff(alarm),

    ¬GoesOff(alarm)

    Allows us to conclude:

    ¬Detects(alarm, intruder)

- ## Universal elimination
  - If $\forall x\ P(x)$ is true, then $P(c)$ is true, where c is *any* constant in the domain of x
  - Variable symbol can be replaced by any ground term

- ## Example:

  $\forall x\ \ \text{RegularlyAttends}(x, \text{lecture}) \rightarrow \text{Pass}(x, \text{lecture})$

  Allow us to conclude (assuming Joe is in the domain of x):

  $\text{RegularlyAttends}(\text{joe}, \text{lecture}) \rightarrow \text{Pass}(\text{joe}, \text{lecture})$

- Existential elimination
  - From ∃x P(x) infer P(c)
  - Note that the variable is replaced by a **brand-new constant** not occurring in this or any other sentence in the KB
  - Also known as skolemization; constant is a **skolem constant**
  - In other words, we don't want to accidentally draw other inferences by introducing the constant
  - Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

- Example:

  ∃x Pass(x)

  Allows us to conclude:

  Pass(someStudent) , where someStudent is a new constant

- If P(c) is true, then ∃x P(x) is inferred.
- The inverse of existential elimination
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

- Example:
    Eats(Ziggy, IceCream)
    ∃x Eats(Ziggy,x)

- Apply modus ponens reasoning to generalized rules

- Combines Universal-Elimination, and Modus Ponens
  - E.g, from *P(c)* and *Q(c)* and $\forall x\ (P(x) \wedge Q(x)) \rightarrow R(x)$ derive *R(c)*

- GMP requires substitutions for variable symbols
  - subst(θ, α) denotes the result of applying a set of substitutions defined by θ to the sentence α
  - A substitution list θ = $\{v_1/t_1, v_2/t_2, ..., v_n/t_n\}$ means to replace all occurrences of variable symbol $v_i$ by term $t_i$
  - Substitutions are made in left-to-right order in the list
  - subst({x/IceCream, y/Ziggy}, eats(y,x)) = eats(Ziggy, IceCream)

- ## General definition: Given
  - **atomic sentences** $P_1, P_2, ..., P_N$
  - **implication sentence** $(Q_1 \wedge Q_2 \wedge ... \wedge Q_N) \rightarrow R$
    - $Q_1, ..., Q_N$ and R are atomic sentences
  - **substitution** $subst(\theta, P_i) = subst(\theta, Q_i)$ for i=1,...,N
  - **Derive new sentence: $subst(\theta, R)$**

- GMP is usually written formally as following:

$$\frac{\vdash A(a)}{\vdash \forall x\, A(x)}$$

- A clause is a disjunction of literals
  - Example: $C_1 \vee \ldots \vee C_n$
- A definite clause is a clause containing exactly one positive literal
  - Example: $\neg C_1 \vee \ldots \vee \neg C_n \vee H$

- GMP is used with a collection of definite clauses

- Example:

  $\dfrac{\text{Person(John)}, \text{Rich(x)}, (\text{Person(x)} \wedge \text{Rich(x)} \rightarrow \text{Popular(x)})}{\text{Popular(John)}}$

  With the substitution $\theta = \{x/\text{John}, y/\text{John}\}$, and
  Rich $\theta$ = Popular $\theta$

- *Completeness*: If a knowledge-base KB entails a statement *S*, we can prove *S*

- *Gödel Completeness Theorem*: There exists a complete proof system for FOL

  - KB $\models$ Q $\leftrightarrow$ KB $\vdash$ Q

  - Gödel proved that there exists a complete proof system for FOL.

  - Gödel did not come up with such a concrete proof system.

- *Robinson's Completeness Theorem*: Resolution is such a concrete complete proof system for FOL

- *FOL is only semi-decidable*: If a conclusion follows from premises, then a complete proof system (like resolution) will find a proof.
  - If there's a proof, we'll halt with it (eventually)
  - However, If there is **no** proof (i.e. a statement does not follow from a set of premises), the attempt to prove it may never halt

- From a practical point of view this is problematic
  - We cannot distinguish between the non-existence of a proof or the failure of an implementation to simply find a proof in reasonable time.
  - Theoretical completeness of an inference procedure does not make a difference in this cases
    - Does a proof simply take too long or will the computation never halt anyway?

# ILLUSTRATION BY LARGER EXAMPLE

- Consider the following english text:

  "Anyone passing his Intelligent System exam and winning the lottery is happy. But any student who studies for an exam or is lucky can pass all his exams.  John did not study but John is lucky.  Anyone who is lucky wins the lottery. Mary did not win the lottery, however Mary passed her IS exam. Gary won the lottery. Gary, John, and Mary are all students."

- Is John happy?
- Is Mary happy? Is she lucky?
- Did every student pass the IS exam?

1. Identify the task
   – Knowledge to be captured, questions to be answered

2. Assemble the relevant knowledge
   – Relavant: Relation between exams, lottery jackpots, passing of an exam information about individuals, etc.
   – Irrelevant: Dates of exams, teacher of the exam, amount of money in the lottery pot, etc.

3. Decide on a vocabulary
   – Predicates
   – Constants symbols
   – E.g. Win(x, Lottery) or Win(x, y) ∧ Lottery(y)?

## 4. Encode general knowledge of the domain

- Anyone passing the IS exams and winning the lottery is happy
  $\forall$x Pass(x, IS) ∧ Win(x, Lottery) $\Rightarrow$ Happy(x)

- Any student who studies or is lucky can pass all his exams
  $\forall$x $\forall$y Student(x) ∧ Exam(y) ∧ (StudiesFor(x, y) ∨ Lucky(x)) $\Rightarrow$ Pass(x,y)

- Anyone who is lucky wins the lottery
  $\forall$x Lucky(x) $\Rightarrow$ Win(x, Lottery)

5. Encode the specific problem instance

  – John did not study, but John is lucky

  ¬Study(John) ∧ Lucky(John)

  – Mary did not win the lottery, however Mary passed her IS exam

  ¬ Win(Mary, Lottery)

  Pass(Mary,IS)

  – Gary won the lottery

  Win(Gary, Lottery)

  – Gary, John, and Mary are all students

  Student(Gary), Student(John), Student(Mary)

## 6. Pose queries to the inference procedure

– Is John happy?

Happy(John) ?

– Is Mary happy? Is she lucky?

Happy(Mary), Lucky(Mary)?

– Did every student pass the IS exam?

$\exists$x Student(x) $\wedge$ Pass(x, IS) ?

• Specific inference procedures (e.g. *Resolution*) can be used to systematically answer those queries (see next lecture)

# EXTENSIONS

- Ordinary first-order interpretations have a single domain of discourse over which all quantifiers range; many-sorted first-order logic allows variables to have different *sorts*, which have different domains

- The characteristic feature of first-order logic is that individuals can be quantified, but not predicates; second-order logic extends first-order logic by adding the latter type of quantification

- Intuitionistic first-order logic uses intuitionistic rather than classical propositional calculus; for example, ¬¬φ need not be equivalent to φ

- Infinitary logic allows infinitely long sentences; for example, one may allow a conjunction or disjunction of infinitely many formulas, or quantification over infinitely many variables

- First-order modal logic has extra *modal operators* with meanings which can be characterised informally as, for example "it is necessary that φ" and "it is possible that φ"

# SUMMARY

- Predicate logic differentiates from propositional logic by its use of quantifiers
  - Each interpretation of predicate logic includes a domain of discourse over which the quantifiers range.

- There are many deductive systems for predicate logic that are sound (only deriving correct results) and complete (able to derive any logically valid implication)

- The logical consequence relation in predicate logic is only semidecidable

- This lecture focused on three core aspects of the predicate logic: Syntax, Semantics, and Inference

# REFERENCES

- Mandatory Reading:
  - M. Fitting: First-Order Logic and Automated Theorem Proving, 1996 Springer-Verlag New York (Chapter 5)
- Further Reading:
  - Michael Huth and Mark Ryan, Logic in Computer Science(second edition) , 2004, Cambridge University Press
  - http://plato.stanford.edu/entries/model-theory/
- Wikipedia links:
  - http://en.wikipedia.org/wiki/First-order_logic

# Next Lecture

| # | Title |
|---|---|
| 1 | Introduction |
| 2 | Propositional Logic |
| 3 | Predicate Logic |
| **4** | **Reasoning** |
| 5 | Search Methods |
| 6 | CommonKADS |
| 7 | Problem-Solving Methods |
| 8 | Planning |
| 9 | Software Agents |
| 10 | Rule Learning |
| 11 | Inductive Logic Programming |
| 12 | Formal Concept Analysis |
| 13 | Neural Networks |
| 14 | Semantic Web and Services |

# Questions?