Logic

Carlos García Martínez

DIAN
DPTO. DE INFORMATICA
Y ANALISIS NUMERICO

Contents

1. Logic in knowledge representation.
2. Propositional logic.
3. Predicate logic.
4. Extensions.
5. Conclusions.

Introduction

- ***Symbolic Logic*** can be used for knowledge representation.
- ***Facts*** are modelled in a ***mathematical language***.
- The inference mechanism is the ***Logic deductive reasoning***.

Characteristics as KBS.

- ***Huge expressiveness.***
- ***Its semantics are well stated.***
- ***Properties well known and stated***.
- ***Deductive reasoning*****.** Two versions:
  - Propositional logic.
  - Predicate Logic.
- ***Semidecidability***.

Contents

1. Logic in knowledge representation.
2. ***Propositional logic.***
3. Predicate logic.
4. Extensions.
5. Conclusions.

Propositional Logic

- Introduction
- Vocabulary
- Semantics
- Inference rules
- Automatic proof

Introduction

- To relate proposition ⇔ p (representation)
- Representation management obtains new propositions.
- ***Knowledge*** comes from the declaration of ***initial facts***.
- The ***inference*** mechanism applies ***deductive logic reasoning:***

– Logic is a representation formalism

– If logic is a representation formalism, then, propositional logic is as well another one

– ***C:*** Propositional logic is a representation formalism

Vocabulary

Constantes

- ***Propositional variables (literals):*** *p*, *q*, *r*, ...
  - They are the jump from the knowledge level to the symbolic one.
  - The opposite jump needs semantic ***tables***.
- ***Connectives***: They associate propositional variables:
  - Basic ones: "$\vee$" "$\neg$"
  - Derivate ones: "$\wedge$" "$\rightarrow$" "$\leftrightarrow$" "$\oplus$"

Connectives

- **Basic or primitive connectives**
  - Or ($\vee$)
  - Not ($\neg$)
- **Derivate connectives** (from primitive ones)
  - And ($\wedge$) $p \wedge q \equiv \neg(\neg p \vee \neg q)$
  - Conditional ($\rightarrow$) $p \rightarrow q \equiv \neg p \vee q$
  - Biconditional ($\leftrightarrow$) $p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$
  - Exclusive or ($\oplus$) $p \oplus q \equiv \neg (p \leftrightarrow q)$
- **Precedence:** $\neg \quad \wedge \quad \vee \quad \oplus \quad \rightarrow \quad \leftrightarrow$
- A ***well formed formula*** is obtained by means of connecting propositional variables.

Semantics

- In order to analyse the semantics of a formula, we construct the ***truth table***.
- Propositional logic is ***bivalued:*** 2^n interpretations.
- Formulas can be:
  - ***Valid***: They are true regardless of any interpretation.
  - ***Satisfiable***: There exists at least one interpretation that makes the formula true.
  - ***Not satisfiable***: There is not any interpretation that makes the formula true.

Truth tables for logic connectives

p	q	¬p	p∧q	p∨q	p⊕q	p→q	p↔q
V	V	F	V	V	F	V	V
V	F	F	F	V	V	F	F
F	V	V	F	V	V	V	F
F	F	V	F	F	F	V	V

Inference rules

- To obtain new valid formulas from initial valid ones, which are called axioms or premises.
- ***Proof process:*** to obtain the concluding formulae from axioms and inference rules.

Inference rules

→ - introduction $$\dfrac{\begin{array}{l}[p]\\ q\end{array}}{p \rightarrow q}$$	**→ - removal** (modus ponens) $$\dfrac{\begin{array}{l}p \rightarrow q\\ p\end{array}}{q}$$ (modus tollens) $$\dfrac{\begin{array}{l}p \rightarrow q\\ \neg q\end{array}}{\neg p}$$
↔ - introduction $$\dfrac{\begin{array}{l}p \rightarrow q\\ q \rightarrow p\end{array}}{p \leftrightarrow q}$$	**↔ - removal** $$\dfrac{p \leftrightarrow q}{p \rightarrow q} \qquad \dfrac{p \leftrightarrow q}{q \rightarrow p}$$
¬ - introduction $$\dfrac{\begin{array}{l}[p]\\ \text{falso}\end{array}}{\neg p}$$	**¬ - removal** $$\dfrac{\begin{array}{l}p\\ \neg p\end{array}}{\text{falso}} \qquad \dfrac{\text{falso}}{p}$$
∧ - introduction $$\dfrac{\begin{array}{l}p\\ q\end{array}}{p \wedge q}$$	**∧ - removal** $$\dfrac{p \wedge q}{p} \qquad \dfrac{p \wedge q}{q}$$
∨ - introduction $$\dfrac{p}{p \vee q} \qquad \dfrac{q}{p \vee q}$$	**∨ - removal** $$\dfrac{p \vee q \qquad \begin{array}{c}[p]\\ r\end{array} \qquad \begin{array}{c}[q]\\ r\end{array}}{r}$$

Resolution principle

Given two ***clauses***, which are formulas with only logical ORs and NOts, containing ***ONLY ONE*** complementary literal, we can construct a new clause containing all the literals except the complementary one (***p*** and ***¬p***):

$$q_1 \vee q_2 \vee q_3 \vee \ldots \vee q_i \vee \boldsymbol{p} \vee q_{i+1} \vee \ldots \vee q_m$$

$$r_1 \vee r_2 \vee r_3 \vee \ldots \vee r_k \vee \boldsymbol{\neg p} \vee r_{k+1} \vee \ldots \vee r_n$$

$$q_1 \vee q_2 \vee \ldots \vee q_i \vee q_{i+1} \vee \ldots \vee q_m \vee r_1 \vee r_2 \vee \ldots \vee r_k \vee r_{k+1} \vee \ldots \vee r_n$$

Procedure to obtain the Conjunctive normal form

- Replace derivate connectives by logical *nots*, *ors* and *ands*.
- Move *nots* inwards, ***De Morgan's Laws:***

  $\neg (p \wedge q) \equiv \neg p \vee \neg q$

  $\neg (p \vee q) \equiv \neg p \wedge \neg q)$

  + double negative law: $\neg \neg p \equiv p$
- ***Distribute ORs over ANDs***:
  - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  - $(p \wedge q) \vee r \equiv (p \vee r) \wedge (q \vee r)$
- Remove tautologies

Example

- $[(p \rightarrow q) \wedge (q \rightarrow r)] \rightarrow (p \rightarrow r)$

Automatic proof

- Given a set of ***premises*** ($p_1 \land p_2 \land \ldots \land p_n$), how does a system test a ***conclusion c?***

- There are three ways:
  - Truth tables.
  - Inference rules.
  - Proof by contradiction (reductio ad absurdum).

Proof by truth tables

We build the truth table associated to

$$(p_1 \wedge p_2 \wedge p_3 \wedge \ldots \wedge p_n \rightarrow \boldsymbol{c})$$

and test if it is a tautology.

Example: Can we obtain *q* from *p* and *p* → *q?* Is the following formula a tautology?

$((p \rightarrow q) \wedge p) \rightarrow q$

Proof by inference rules

Combine the premises p_i by means of the inference rules until c is obtained.

Proof by contradiction

To test if the following formula is false

$$p_1 \wedge p_2 \wedge p_3 \wedge \ldots \wedge p_n \wedge \neg \mathbf{c}$$

by means of:

- Truth tables
- Inference rules
- Resolution principle

Contents

1. Logic in knowledge representation.
2. Propositional logic.
3. ***Predicate logic.***
4. Extensions.
5. Conclusions.

Predicate Logic

- Introduction
- Vocabulary
- Semantics
- Inference rules
- Automatic proof.

Introduction

- It ***improves*** the lack of expressiveness of the propositional logic.
- Characteristics:
  - It relates ***predicates*** instead of propositions.
  - It can express ideas that propositional logic can not:
    - Socrates is a man
    - Every man is mortal

    - ??????????????????????

Vocabulary

- Predicates, (with variables).
- Quantifiers:
  - Universal ($\forall$)
  - Existential ($\exists$)
- Functions.
- Connectives: from propositional logic

Predicates

$$Name(Arg_1, Arg_2, ..., Arg_N)$$

- Arg_i can be a variable or atom.
- ***Variables:*** represented by the last characters from the alphabet
- ***Atoms***: they are concrete values, constants.
- The ***number of arguments*** must be constant.
- The truth of a predicate depends on the values of its arguments.

Quantifiers

- They appear next to variables
- They are related to a ***domain,*** use parentheses.
- ***Universal Quantifier:*** any value makes the expression true.
- ***Existential Quantifier:*** there is, at least, one value that makes the expresión true.

Quantifier equivalence

$$\forall x \ \ p(x) \equiv \neg (\exists x \ \ \neg p(x))$$

$$\exists x \ \ p(x) \equiv \neg (\forall x \ \ \neg p(x))$$

$$\neg (\forall x \ \ p(x)) \equiv \exists x \ \ \neg p(x)$$

Functions

$$Name(Arg_1, Arg_2, \ldots, Arg_N)$$

- Its syntax is equivalent to that of predicates, but its semantic is different:
  - Predicates may be true or false.
  - Functions return values of any type (numbers, strings, structs,...)

Grammar

argument := variable | constant | function(argument_list)

argument_list := argument | argument "," argument_list

atomic_formula := predicate(argument_list)

operator := "∧" | "∨" | "→" | "↔"

quantifier := "∀" | "∃"

formula := V | F | atomic_formula | ¬ formula | formula operator formula | quantifier variable formula | (formula)

Semantics

- **Atomic Formula,** only one predicate. Its value depends on the values of its arguments.
- To obtain the value of a formula, every variable must be quantified. The value depends on:
  - The universally quantified variables
  - The existentially quantified variables.
  - The functions.
  - Predicates.
  - Connectives.

Inference rules in predicate logic

- Propositional rules plus the following ones:
  - Rule 1: Existential quantifier introduction:
    $P(a) \rightarrow \exists x\, P(x)$
  - Rule 2: Existential quantifier removal:
    $\exists x\, P(x) \rightarrow P(a)$
  - Rule 3: Universal quantifier introduction:
    For all constant P is true $\rightarrow \forall x\, P(x)$
  - Rule 4. Universal quantifier removal:
    $\forall x\, P(x) \rightarrow$ For any constant P is true

Automatic Proof. Resolution Principle

- Steps:
  - To obtain the conjunctive normal form
  - Unification and resolution principle

Conjunctive normal form

- Also called ***Skolem normal form***.
- Properties:
  - There are only ***universal quantifiers*** at the beginning of the formula (every variable is universally quantified).
  - The remainder of the formula consists of conjunctions of ***clauses,*** disjunctions of literals.
    - Conjunction of disjunction of literals

Conjunctive normal form (2). Procedure

- Replace derivative connectives with primitive (and $\wedge$) ones.
- Move NOTs inward by means of:
  - De Morgan's laws
  - Double negation law
  - Quantifier equivalence laws.
- Make quantified variables independent.
- Remove existential quantifiers:
  - when it is out of domain of every universal quantifier.
  - when it is in the domain of a universal quantifier.
- Move universal quantifiers to the beginning of the formula.
- Remove universal quantifiers.
- Distribute ORs over ANDs.
- Make a clause for every disjunction.
- Change the variables' names.

Example

- $\forall x(big(x) \wedge house(x) \rightarrow work(x) \vee (\exists y\ clean(y,x) \wedge \neg \exists y\ garden(y,x)))$

Unification

- At the resolution principle, arguments of predicates are important. For instance:

$$Man(Socrates) \vee \neg Man(Socrates)$$

is a tautology, whereas

$$Man(Socrates) \vee \neg Man(dog(Socrates)), \text{ no.}$$

- In order to be able to apply the resolution principle, we must make literals equal. For instance:

$$Man(x) \vee \neg Man(Socrates)$$

are equal if variable x gets the value “Socrates”

Unification(2). Concept

- Process that performs variable substitutions in order to make literals equal.
- Substitution ***s*** of variables $\boldsymbol{x_1, x_2, \ldots, x_n}$ by terms (functions, variables, constants) $\boldsymbol{t_1, t_2, \ldots, t_n}$ consists in replacing every variable $\boldsymbol{x_i}$ ***apparition*** by the corresponding term $\boldsymbol{t_i}$, in the clause. It is denoted by the set $\boldsymbol{\{t_1/x_1, t_2/x_2, \ldots, t_n/x_n\}}$.

Substitution Properties

- There exists an ***empty substitution*** {} that does not modify the clause.
- Substitution can be composed**:**

$$Ls_1s_2 = (Ls_1)s_2$$

- Substitution composition is ***associative***:

$$(s_1s_2)\,s_3 = s_1\,(s_2s_3)$$

- Substitution composition is ***not commutative***:

$$s_1s_2 \neq s_2s_1$$

Unification algorithm

1. Look for the first difference between two comparable predicates:
   - If they are constant, unification is impossible.
   - If one of them is a constant “a” and the another is a variable “x”, then, apply substitution “a/x”.
   - If both are variables “x” and “y”, then, apply substitution “x/y” (or “y/x”.
   - If one of them is a function (with one or more variables) and the another one is a variable, substitute the variable by the function with the same variables.
   - If both are the same function, then, invoque recursively the unification algorithm for the argument lists.
2. Go to step 1 while there are differences.

Resolution principle

- The procedure starts with a set of clauses, whose variables use different names.
- Find two clauses containing the same predicate, where it is negated in one clause but not in the other.
- Perform unification to make the arguments of the complementary literal equal.
- Combine both clauses by the logical OR connective, discarding the unified predicates

Resolution principle (2)

1. Select a ***pair of literals*** in two different clauses.
2. Perform ***unification.***
3. If unification successes, include the combination of both clauses, discarding the unified literals.
4. If the result is the ***empty clause***, there was a contradiction in the initial clauses (success).
5. If there is no pair of literals to be unified, the method stops.
6. If there are more pairs of literals to be unified, go to step 1.

Exercise

1) man(Marco)
2) pompeian(Marco)
3) $\forall x$ (pompeian(x) $\rightarrow$ roman(x))
4) governor(Cesar)
5) $\forall x$ (roman(x) $\rightarrow$ faithful(x,César) $\vee$ hate(x,César))
6) $\forall x \exists y$ faithful(x,y)
7) $\forall x \forall y$ (person(x) $\wedge$ governor(y) $\wedge$ try_to_kill(x,y) $\rightarrow$ $\neg$faithful(x,y))
8) try_to_kill(Marco,César)
9) $\forall x$ (man(x) $\rightarrow$ person(x))
10) ¿¿$\neg$faithful(Marco,Cesar)??

Obtain the Conjunctive normal form

Prove by contradiction with resolution principle

Exercise

- Prove that “someone passes AI” by contradiction and the resolution principle, given that:
  - “If someone solves the exercises by himself (he does not copy the results), then, he passes AI”
  - “If someone copies the answers, then, another one solves them by himself”.
  - “Pepe copies the answers”.

Contents

1. Logic in knowledge representation.
2. Propositional logic.
3. Predicate logic.
4. ***Extensions.***
5. Conclusions.

Extensions

- Modal logic
- Predicate logic with identity
- Classes and relations logic
- Superior order predicate logic
- Multivalued logic
- Fuzzy logic
- No monotonous logic

Conclusions

- Huge expressiveness
- Inference ability
- Propositional logic
  - Decidability
  - Limited
- Predicated logic
  - Improve propositional logic
  - Semidecidability
- Drawbacks:
  - unstructured knowledge
  - There is some kind of information that is inexpressible logic.