



Universidade do Minho
Escola de Engenharia
Departamento de Informática

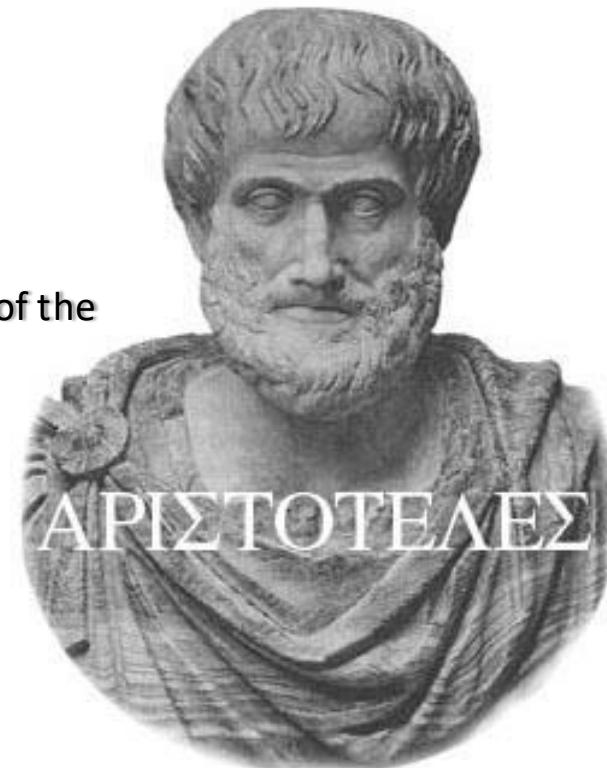
Logic

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA

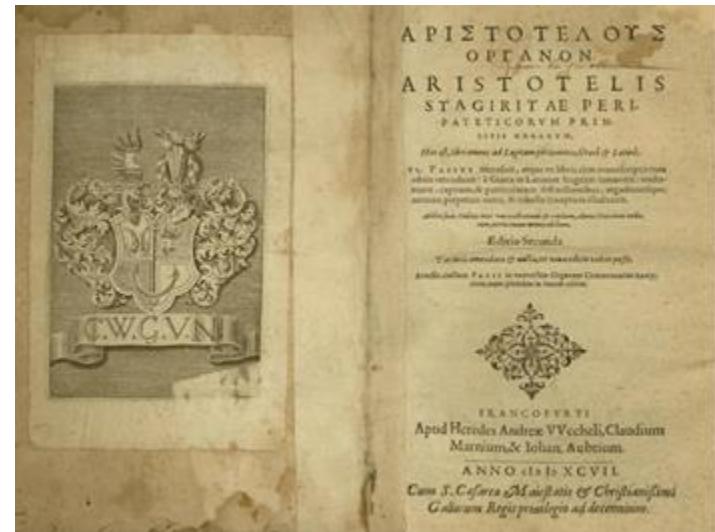
In inteligência Artificial
2025/26

- Knowledge and Reasoning;
- Logic and Logic Programming;
- Production rules;
- Standards-Driven Programming;
- Hierarchical structures:
 - Semantic networks;
 - Frames;
- Scripts;
- Knowledge-Based Systems.

- Logic:
 - From the Greek "logos" (λογική)
 - [Priberam Dictionary of the Portuguese Language](#)
 - [Infopédia - Porto Editora dictionaries](#)
- Logic's object of study is the general laws of thought (reasoning) (reasoning) and how to apply them correctly in the investigation of the truth (mechanisation of reasoning);



- Aristotle (384 BC - 322 BC)
 - Greek philosopher;
 - He systematised his knowledge of Logic in the work "Organon", a set of 6 texts;
 - "Organon": instrument; tool for correct thinking;



- He studied the way in which the chain of observations makes it possible to obtain new knowledge: reasoning;
 - Term: thing; representation of something;
 - Deadly;
 - Flag;
 - Proposition: a statement that can be evaluated logically;
 - The Portuguese flag is round;
 - Syllogism: a mechanism of interpretation that derives a conclusion from two statements:
 - Any man is mortal;
 - Socrates is a man;
 - Therefore, Socrates is mortal.
- On the basis of observations judged to be true, Logic's place is in the formulation of general chain laws that allow new truths to be discovered.

- Logic:
 - Relationship of propositions
- Proposition:
 - Affirmation (fact)
- Predicate:
 - Something you say about something



- **Logic:**

- Science of reasoning

in Priberam Dictionary of the Portuguese Language

- A normative discipline, traditionally linked to philosophy, which sets out to determine the conditions of truth in the different fields of knowledge.

in Infopédia - Porto Editora Dictionaries

- **Proposition:**

- Verbal statement that can be declared true or not;

in Priberam Dictionary of the Portuguese Language

- **Predicate:**

- Positive quality; virtue; merit;

In Infopédia - Porto Editora Dictionaries

- Characteristic property of something or someone (attribute);

in Priberam Dictionary of the Portuguese Language



- Mathematical logic adopts two fundamental principles of thought:
 - Principle of non-negation:
 - A proposition cannot be true and false at the same time;
 - Principle of the excluded third party:
 - Any proposition is either true or false;
 - There is always one of these cases and never a third.
- Connection operators, or logical connectives:
 - Expressions or words used to compose new propositions (and, or, not, if ... then, ... if and only if ..., etc.);
- Logical values:
 - The logical value of a proposition is true if the proposition is true;
 - The logical value of a proposition is false if the proposition is false.

Propositions

1. The sky is overcast.
2. Cristiano Ronaldo is president of the Portuguese Republic.
3. What time is it?
4. It's half past seven in the afternoon.
5. Vitória is the best club in the world.
6. Vitória is the only club that exists.
7. Which Victory?
8. Keep it down!
9. I wish I was from Vitoria.
10. Portugal is the European Football Champion.
11. Are Portugal Football World Champions?
12. Portugal are World Football Champions.

Propositions

1. The sky is overcast.
2. Cristiano Ronaldo is president of the Portuguese Republic.
- ~~3. What time is it?~~
4. It's half past seven in the afternoon.
5. Vitória is the best club in the world.
6. Vitória is the only club that exists.
- ~~7. Which Victory?~~
- ~~8. Keep it down!~~
- ~~9. I wish I was from Vitoria.~~
10. Portugal is the European Football Champion.
- ~~11. Are Portugal Football World Champions?~~
12. Portugal are World Football Champions.

- Proposition:
 - is a statement that can be classified as true or false, i.e. it has a **Truth Value**.
- Non-proposal:
 - It's a sentence (command, exclamation, question, wish, advice) that can't be classified as true or false, and therefore doesn't have a **Truth Value**.
- Simple propositions:
 - Propositions that cannot be broken down into other propositions:
 - Guimarães is a city.
- Compound propositions:
 - Propositions that can be broken down into simpler propositions:
 - Guimarães was the first capital of Portugal and Lisbon is the current capital.

- Based on the principle of the excluded third, any proposition has the logical value of true (\top) or false (\perp);
- In the case of compound propositions, their logical value depends solely on the logical values of the simple propositions that make it up;
- **Truth tables** are used to determine the logical value of compound propositions:

p
\perp
\top

p	q
\perp	\perp
\perp	\top
\top	\perp
\top	\top

p	q	r
\perp	\perp	\perp
\perp	\top	\perp
\top	\perp	\perp
\top	\top	\perp
\top	\perp	\top
\top	\top	\top
\perp	\perp	\top
\perp	\top	\top
\top	\perp	\top
\top	\top	\top

- Negation (\neg): the negation of a proposition p is the proposition represented by $\neg p$, whose logical value is truth (⊤) when p is false and falsity (⊥) when p is true;

p	$\neg p$
⊤	⊥
⊥	⊤

- **Conjunction (\wedge):** the conjunction of two propositions p and q is the proposition represented by $p \wedge q$, whose logical value is true (⊤) when both propositions p and q are true and false (⊥) in other cases;

p	q	$p \wedge q$
⊤	⊤	⊤
⊤	⊥	⊥
⊥	⊤	⊥
⊥	⊥	⊥

- Disjunction (\vee): the disjunction of two propositions p and q is called the proposition represented by $p \vee q$, whose logical value is truth (⊤) when at least one of the propositions is true and falsity (⊥) when both propositions are false;

p	q	$p \vee q$
⊤	⊤	⊤
⊤	⊥	⊤
⊥	⊤	⊤
⊥	⊥	⊥

- Exclusive disjunction (\vee'): the exclusive disjunction of two propositions p and q is the proposition represented by $p \vee' q$, whose logical value is true (\top) when only one of the propositions is true and false (\perp) when both propositions are false or both are true;

p	q	$p \vee' q$
\perp	\perp	\perp
\top	\perp	\top
\perp	\top	\top
\top	\top	\perp

- Implication (\rightarrow): the proposition represented by $p \rightarrow q$ (read "if p then q "), whose logical value is false (\perp) when p is true and q is false and true (\top) in other cases, is called implication;

p	q	$p \rightarrow q$
\perp	\perp	\perp
\perp	\top	\top
\top	\perp	\top
\top	\top	\top

- Equivalence (\leftrightarrow): equivalence is the proposition represented by $p \leftrightarrow q$ (read "p is equivalent to q"), whose logical value is true (⊤) when both propositions are true or false and false (⊥) in other cases;

p	q	$p \leftrightarrow q$
⊤	⊤	⊤
⊤	⊥	⊥
⊥	⊤	⊥
⊥	⊥	⊤

- Assume that:
 - p: Lisbon is the capital of Brazil; q: Carlos is a lawyer; r: It's raining;
1. $\neg p$: Lisbon is not the capital of Brazil
 2. $\neg q$: Carlos is not a lawyer
 3. $p \wedge q$: Lisbon is the capital of Brazil and Carlos is a lawyer
 4. $p \vee r$: _____
 5. $\neg q \wedge \neg r$: _____
 6. $p \vee q \wedge r$: _____
 7. $(p \vee q) \wedge r$: _____
 8. $\neg(\neg r \wedge \neg q) \vee p$: _____

- Assume that:
 - p: Lisbon is the capital of Brazil; q: Carlos is a lawyer; r: It's raining;
1. $\neg p$: Lisbon is not the capital of Brazil
 2. $\neg q$: Carlos is not a lawyer
 3. $p \wedge q$: Lisbon is the capital of Brazil and Carlos is a lawyer
 4. $p \vee r$: Lisbon is the capital of Brazil or it's raining
 5. $\neg q \wedge \neg r$: Carlos is not a lawyer and it's not raining
 6. $p \vee q \wedge r$: Lisbon is the capital of Brazil or Carlos is a lawyer and it's raining
 7. $(p \vee q) \wedge r$: Lisbon is the capital of Brazil and it's raining or Carlos is a lawyer and it's raining
 8. $\neg(\neg r \wedge \neg q) \vee p$: Lisbon is the capital of Brazil or it's not true that it's not raining and Carlos is not a lawyer

- How can logic and computerisation be used to automate reasoning procedures?
- There are two approaches to developing Logic as a computational (programming) mechanism:
- The Theory of Models
 - Hodges, Wilfrid - Model theory. Cambridge University Press, Cambridge 1997.
 - (<https://doi.org/10.1017/CBO9780511551574>)
- The Theory of Evidence
 - A. S. Troelstra, H. Schwichtenberg (1996). Basic Proof Theory. In series Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, ISBN 0-521-77911-1.
 - (<https://doi.org/10.2307/2586674>)

- Model Theory

- Examines the relationships between logical formulae, when interpreted
- Associate them with value domains
- Assign them truth values
- It uses terms such as:
 - True
 - False
 - Interpretation
 - Satisfaction
 - Model
 - Implications
 - Semantic consequence

- Model Theory

- Examines the relationships between logical formulae, when interpreted
- Associate them with value domains
- Assign them truth values
- It uses terms such as:
 - True
 - False
 - Interpretation
 - Satisfaction
 - Model
 - Implications
 - Semantic consequence

- Theory of Evidence

- Examines the relationships between logical formulae through derivability from other formulae.
- Uses rules that act only on the structure of formulas
- It uses terms such as:
 - Axiom
 - Inference rule
 - Theorem
 - Proof
 - Consistency
 - Syntactic consequence

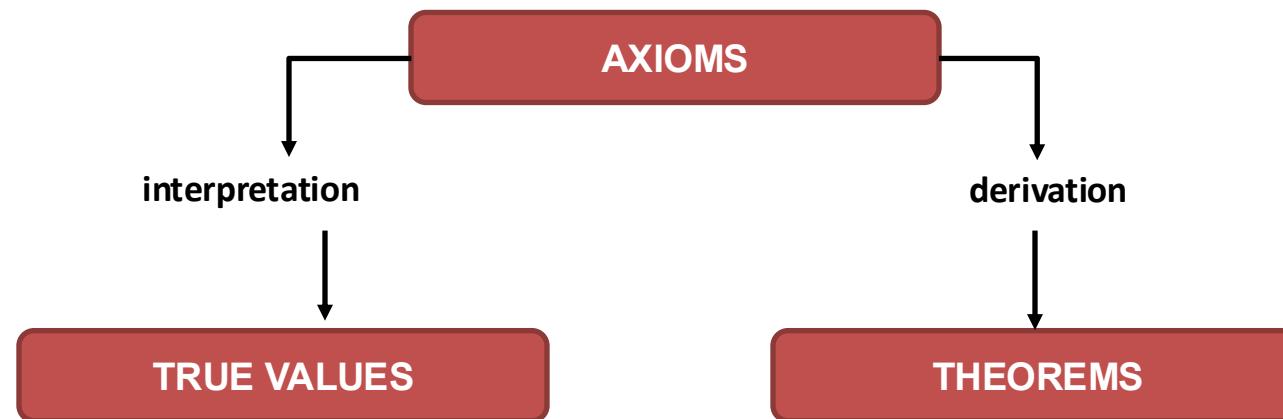
- Model Theory
 - What we intend to be true
 - The answers a programme implies
- Theory of Evidence
 - What we can prove
 - The answers that are computationally obtained from a programme

- Model Theory

- What we intend to be true
- The answers a programme implies

- Theory of Evidence

- What we can prove
- The answers that are computationally obtained from a programme



- Model Theory considers the attribution of meaning to logical formulae
- The relationship of Logical Implication:
 - It's a relation on a language L, such that it is:
 - L a first-order (predicative) logical language
 - S is a subset of L
 - s a member of L
 - It follows that:
 - $\models = \{ \langle S, s \rangle \text{ where any model for } S \text{ is a model for } s \}$
 - $S \models s$
 - Using the notion of truth values about L

- Proof Theory considers the generation of logical formulae from other logical formulae;
- Proof Theory uses the notion of Derivability of a formula by applying a set of derivation rules;
- The Derivability relation:
 - It's a relation on a language L, such that it is:
 - L a first-order (predicative) logical language
 - S is a subset of L
 - R a set of derivation rules
 - s a member of L
 - It follows that:
 - $\vdash = \{ \langle S, s \rangle \mid s \text{ is derivable from } S \text{ using } R \}$
 - $S \vdash s$

- Axioms: initial set of logical formulae
- Theorems: formulae derived from axioms and/or theorems (semantic consequences)
- Inference Rules: set of derivation rules
 - *Modus ponens* { (A if B), B } ⊢ A (*sound - valid*)
 - *Modus tollens* { (A if B), ¬A } ⊢ ¬B (*sound - valid*)
 - *Modus mistakens* { (A if B), A } ⊢ B (*unsound - not valid*)
- Inference system: union of axioms and derivation rules R
- Proof: sequence $\langle s_1, s_2, \dots \rangle$ of s_i that are axioms or are derivations using R and a subset of the members of the sequence that precede s_i
 - The sequence is a proof for s_n (derivation or deduction)
- Theory: union of the axioms and all the theorems derived using R
 - It is said to be consistent if there is no formula s such that, in theory T, there exists s and $\neg s$
- **None of these considerations take meaning into account!**
Only syntactic structure!!!

The proof of theorems in Logic

- It is intended that:
 - s is logically implied by S
- In other words, that correct answers are obtained in any interpretation that uses valid rules of inference (*modus ponens* or *modus tollens*)
 - $\exists S, s : S \models s \text{ se } S \vdash s$ (derivability is a subset of logical implication)
- The aim is also to:
 - s is derivable from S , whenever S implies s
- So that there are no correct answers that cannot be obtained through a test
 - $\exists S, s : S \vdash s \text{ se } S \models s$ (logical implication is a subset of derivability)
- This happens when L (language):
 - It's Propositional Logic
 - It's first-order Predicative Logic
 - Follows clausal notation

- Logic Programming is a computer formalism that combines 2 basic principles:
 1. Uses logic to represent knowledge
(representation of assumptions and conclusions)
 1. Use Inference to manipulate knowledge
(establish logical relationships between assumptions and conclusions)
(mechanising proof procedures; reasoning)

Characterisation of Logic Programming PROLOG

- A programme in PROLOG is created by adding formulas called clauses
- The clauses can be of 3 types:
 - Facts: express something that is always true

p. son(xico,quim).
 - Rules: express something that is true, depending on the veracity of the conditions

p if q. father(josé,joão) if son(joão,josé).
 - Questions: express something that is true, depending on the veracity of the conditions

?q. ? father(josé,joão).
¬q. ¬father (Joseph, John).

Characterisation of Logic Programming PROLOG

- A programme in PROLOG is created by adding formulas called clauses

- The clauses can be of 3 types:

- Facts: express something that is always true

p. son(xico,quim).

- Rules: express something that is true, depending on the veracity of the conditions

p if q. father(josé,joão) if son(joão,josé).

- Questions: express something that is true, depending on the veracity of the conditions

?q. ? father(josé,joão).

¬q. ¬father (Joseph, John).

p if q.

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i. e,

p :- q

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i.e.

o or, assuming the expansion of q

[red box]

p :- q

p :- q₁ \wedge q₂ \wedge ... \wedge q_n

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i.e.

o or, assuming the expansion of q

o using connective logical implication (\Rightarrow)

p \Rightarrow q

p \Rightarrow q₁ \wedge q₂ \wedge ... \wedge q_n

q₁ \wedge q₂ \wedge ... \wedge q_n \Rightarrow p

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i.e.

p ↗ q

o or, assuming the expansion of q

p ↗ q₁ ∧ q₂ ∧ ... ∧ q_n

o using connective logical implication (↗)

q₁ ∧ q₂ ∧ ... ∧ q_n ↗ p

o applying the equivalence rule of introducing the implication

A ↗ B ↗ ¬A ∨ B

o we now have

¬(q₁ ∧ q₂ ∧ ... ∧ q_n) ∨ p

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i.e.

- or, assuming the expansion of q
- using connective logical implication (\Rightarrow)
- applying the equivalence rule of introducing the implication
- we now have
- applying Morgan's Law
- we'll have

p \Rightarrow q

p \Rightarrow q₁ \wedge q₂ \wedge ... \wedge q_n

q₁ \wedge q₂ \wedge ... \wedge q_n \Rightarrow p

A \Rightarrow B \Leftrightarrow $\neg A \vee B$

$\neg(q_1 \wedge q_2 \wedge \dots \wedge q_n) \vee p$

$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$

$\neg q_1 \vee \neg q_2 \vee \dots \vee \neg q_n \vee p$

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i.e.

- or, assuming the expansion of q
- using connective logical implication (\Rightarrow)
- applying the equivalence rule of introducing the implication
- we now have
- applying Morgan's Law
- we'll have
- that we can reduce to
- where

$i \neq 0$

$0 \leq j \leq 1$

$p \Rightarrow q$

$p \Rightarrow q_1 \wedge q_2 \wedge \dots \wedge q_n$

$q_1 \wedge q_2 \wedge \dots \wedge q_n \Rightarrow p$

$A \Rightarrow B \Leftrightarrow \neg A \vee B$

$\neg(q_1 \wedge q_2 \wedge \dots \wedge q_n) \vee p$

$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$

$\neg q_1 \vee \neg q_2 \vee \dots \vee \neg q_n \vee p$

$\neg q_i \vee p_j$

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

o p if q

i.e.

- or, assuming the expansion of q
- using connective logical implication (\Rightarrow)
- applying the equivalence rule of introducing the implication

o we now have

o applying Morgan's Law

o we'll have

o that we can reduce to

o where

$i \neq 0$

$0 \leq j \leq 1$

o i.e.

p \Rightarrow q

$p \Rightarrow q_1 \wedge q_2 \wedge \dots \wedge q_n$

$q_1 \wedge q_2 \wedge \dots \wedge q_n \Rightarrow p$

$A \Rightarrow B \Leftrightarrow \neg A \vee B$

$\neg(q_1 \wedge q_2 \wedge \dots \wedge q_n) \vee p$

$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$

$\neg q_1 \vee \neg q_2 \vee \dots \vee \neg q_n \vee p$

$\neg q_i \vee p_j$

$q_i \Rightarrow p_j$

Characterisation of Logic Programming PROLOG

- Using the language of Logic to represent knowledge, we use rules like:

- $p \text{ if } q$

i.e.

- or, assuming the expansion of q
 - using connective logical implication (\Rightarrow)
 - applying the equivalence rule of introducing the implication
 - we now have
 - applying Morgan's Law
 - we'll have
 - that we can reduce to

- where

Horn clauses

$i \geq 0$

$0 \leq j \leq 1$

- i.e.

- $p \sqsupseteq q$

$p \sqsupseteq q_1 \wedge q_2 \wedge \dots \wedge q_n$

$q_1 \wedge q_2 \wedge \dots \wedge q_n \sqsupseteq p$

$A \sqsupseteq B \Leftrightarrow \neg A \vee B$

$\neg(q_1 \wedge q_2 \wedge \dots \wedge q_n) \vee p$

$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$

$\neg q_1 \vee \neg q_2 \vee \dots \vee \neg q_n \vee p$

$\neg q_i \vee p_j$

$q_i \sqsupseteq p_j$

- Horn's Clause (clausal notation of first-order logic)
 - It's a restricted version of Predicative Calculus
 - It's a well-formed formula
 - All formulas are universally quantified
 - All formulas are closed
 - Logical formulae only allow 1 term in the positive disjunction of literals

- Horn's Clause (clausal notation of first-order logic)
 - It's a restricted version of Predicative Calculus
 - It's a well-formed formula
 - All formulas are universally quantified
 - All formulas are closed
 - Logical formulae only allow 1 term in the positive disjunction of literals

Horn clauses

- $\neg q_i \vee p_j$
- where $i \geq 0$
 $0 \leq j \leq 1$

Recommended Bibliography

- Stuart Russell and Peter Norvig, Artificial Intelligence - A Modern Approach, 4rd edition, ISBN: 978-0134610993, 2020, Chapter 7.
- Ivan Bratko, PROLOG: Programming for Artificial Intelligence, 3rd Edition, Addison-Wesley Longman Publishing Co., Inc., 2000.



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Logic

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA

In inteligência Artificial
2025/26