Universidade do Minho
Escola de Engenharia
Departamento de Informática

Programação em Lógica

LICENCIATURA EM ENGENHARIA INFORMÁTICA
MESTRADO integrado EM ENGENHARIA INFORMÁTICA
Inteligência Artificial
2025/26

From the last lesson

Limitations of problem solving problems

- Search systems are very efficient at solving problems that can be formalised by:
  - an initial state, actions and final state (or final states).
- but they are not able to solve problems that require **reasoning based on knowledge about the *world***.
  - Because their *model of the world* is poor and their thinking is limited
    - e.g. medical diagnostics, expert systems in general,...
  - even in cases that are (apparently) solvable by search (using planning), it may be necessary to add explicit knowledge.

- Axioms: initial set of logical formulae
- Theorems: formulae derived from axioms and/or theorems (semantic consequences)
- Inference Rules: set of derivation rules
  - *Modus ponens* $\{(A \text{ se } B), B\} \vdash A$ (*sound* - valid)
  - *Modus tollens* $\{(A \text{ se } B), \neg A\} \vdash \neg B$ (*sound* - valid)
  - *Modus mistakens* $\{(A \text{ se } B), A\} \vdash B$ (*unsound* – not valid)
- Inference system: union of axioms and derivation rules $\mathcal{R}$
- Proof: sequence $\langle s_1, s_2, \ldots \rangle$ of s_i that are axioms or are derivations using $\mathcal{R}$ 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 $\mathcal{R}$
  - It is said to be consistent if there is no formula s such that, in theory $\mathcal{T}$, there exists s and $\neg s$
- **None of these considerations take meaning into account!**
  Only syntactic structure!!!

- Knowledge and Reasoning;
- Logic;
- **Logic programming;**
- **Knowledge-Based Systems**.

Synthetic Intelligence Lab

Logic Programming

- 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)

Synthetic Intelligence Lab

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.

- 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

  - $\boxed{\neg q \vee p_j}$
    - where $i \geq 0$, $0 \leq j \leq 1$

ISLab
Synthetic Intelligence Lab

Definitions

- Reasoning:
  - A set of linked thoughts
  - Linking facts that lead to a conclusion
  - A mixture of the previous 2

Synthetic Intelligence Lab

Definitions

- Reasoning:
  1. …
  2. Logical train of thought
  3. …
  4. [LOGIC] Discursive operation of thought by which we conclude that one or more propositions (premises) imply the truth (…) of another proposition (conclusion).

(in Dicionário da Língua Portuguesa, Porto Editora)

- Three men, António, Belmiro and Carlos, are Dulce, Eduarda and Filipa's spouses.
- We don't know who is married to whom.
- Their backgrounds are in engineering, law and medicine.
- We don't know who does what.
- Based on the data below, find out the name of each wife and the profession of each man:
  - The doctor is married to Filipa;
  - Carlos is a lawyer;
  - Eduarda is not married to Carlos;
  - Belmiro is not a doctor.

(adapted from: www.novaconcursos.com.br/portal/dicas/questao-raciocinio-logico-para-concursos)

Synthetic Intelligence Lab

Reasoning and Inference

- In order to convince people that the answer is not "a lucky shot", you need to explain the reasons that lead you to your conclusion!
conclusion!

What does this mean?

- In order to convince people that the answer is not "a lucky shot", you need to explain the reasons that lead you to your conclusion!
conclusion!

What does this mean?

- In order for the conclusion to be accepted, it is necessary to explain the reasoning mechanism applied, i.e. the **process of inference**.

Synthetic Intelligence Lab

Definitions

- Inference:
    - It refers to the applied process that allows us to move from premises to conclusion.
    - Inference: deduction or conclusion.

(in Priberam Dictionary of the Portuguese Language)

Synthetic Intelligence Lab

Definitions

- Inference:
  - It refers to the applied process that allows us to move from premises to conclusion.
  - Inference: deduction or conclusion.

(in Priberam Dictionary of the Portuguese Language)

- Inference rules:
  - *Modus ponens*:
    - from: P , P → Q
    - it follows: Q

Synthetic Intelligence Lab

Definitions

- Inference:
  - It refers to the applied process that allows us to move from premises to conclusion.
  - Inference: deduction or conclusion.

(in Priberam Dictionary of the Portuguese Language)

- Inference rules:
  - *Modus ponens*: (*modus ponendo ponens Latin meaning "the way that affirms by affirming"*)
    - from: P , P → Q
    - it follows: Q

P	Q	P → Q
$\mathbb{V}$	$\mathbb{V}$	$\mathbb{V}$
$\mathbb{V}$	$\mathbb{F}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{V}$	$\mathbb{V}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{V}$

ISLab
Synthetic Intelligence Lab

Definitions

- Inference:
  - It refers to the applied process that allows us to move from premises to conclusion.
  - Inference: deduction or conclusion.

(in Priberam Dictionary of the Portuguese Language)

- Inference rules:
  - *Modus ponens*:
    - from: P , $P \rightarrow Q$
    - it follows: Q
  - *Modus tollens*:
    - from: $P \rightarrow Q$, $\neg Q$
    - inferred: $\neg P$

ISLab
Synthetic Intelligence Lab

Definitions

- Inference:
    - It refers to the applied process that allows us to move from premises to conclusion.
    - Inference: deduction or conclusion.

(in Priberam Dictionary of the Portuguese Language)

- Inference rules:
    - *Modus ponens*:
        - from: P , $P \rightarrow Q$
        - it follows: Q
    - *Modus tollens* (Latin: way that negates by negation):
        - from: $P \rightarrow Q$, $\neg Q$
        - inferred: $\neg P$

P	Q	$\neg Q$	$P \rightarrow Q$
$\mathbb{V}$	$\mathbb{V}$	$\mathbb{F}$	$\mathbb{V}$
$\mathbb{V}$	$\mathbb{F}$	$\mathbb{V}$	$\mathbb{F}$
$\mathbb{F}$	$\mathbb{V}$	$\mathbb{F}$	$\mathbb{V}$
$\mathbb{F}$	$\mathbb{F}$	$\mathbb{V}$	$\mathbb{V}$

Resolution Algorithm

- The *modus ponens* inference rule makes it possible to derive the conclusion of a clause as true by proving its conditions:

$$\{ \ (\ A \text{ if } B\ ),\ B \ \} \vdash A$$

Resolution Algorithm

- The *modus ponens* inference rule makes it possible to derive the conclusion of a clause as true by proving its conditions:

$$\{ (\ A \text{ if } B\ ),\ B\ \} \vdash A$$

- Applying the *modus tollens* rule of inference makes it possible to direct the search for proof to a specific point in the reasoning process:

$$\{ (\ A \text{ if } B\ ),\ \neg A\ \} \vdash \neg B$$

- which makes it possible to develop a **mechanism of proof by contradiction**.

Resolution Algorithm

- Suppose we have a programme □ in which we want to determine the derivability of a question A:
  i. Let's admit the negation of A:
  $$\neg A$$
  ii. Insert the negation of A into the programme □:
  $$\mathcal{P} \cup \neg A$$
- If a contradiction occurs
  $$\{ A, \neg A \} \equiv \square$$
  it means that the initial question A is derivable from $\mathcal{P}$.

Synthetic Intelligence Lab

Resolution Algorithm

- Suppose we have a programme □ in which we want to determine the derivability of a question A:
  - i. Let's admit the negation of A:

    $$\neg A$$

  - ii. Insert the negation of A into the programme □:

    $$\mathcal{P} \cup \neg A$$

- If a contradiction occurs

  $$\{ A, \neg A \} \equiv \Box$$

  it means that the initial question A is derivable from $\mathcal{P}$.

- Modus tollens inference rule, with B = true

$$\{ (A \text{ se } B), \neg A \} \vdash \neg B$$

$$\{ (A \text{ se } \mathbb{V}), \neg A \} \vdash \neg \mathbb{V}$$

$$\{ A, \neg A \} \vdash \mathbb{F}$$

$$\{ A, \neg A \} \vdash \Box$$

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example

```
% son: Son,Father → {𝕍,𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍,𝔽}

father( P,F ) :- son( F,P ).
```

Consider the programme $\mathcal{P}$ shown opposite

Synthetic Intelligence Lab

Resolution Algorithm
Application example (I)

- Is João José's son?

```
% son: Son,Father → {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

Let's take the question posed above

Synthetic Intelligence Lab

Resolution Algorithm
Application example (I)

- Is João José's son?

  ¬son(john,jose)

```
% son: Son,Father → {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

In logical terms, the question is described as follows

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example (I)

- Is João José's son?

  ¬son(john,jose)

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

Of all the clauses in the programme $\mathcal{P}$, only those that offer conclusions about the predicate father/2 will contribute to the development of the test

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example (I)

- Is João José's son?

  ¬son(john,jose)

```
% son: Son,Father → {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

Of the three clauses listed, the first will be used in the first step of developing the (tree of) evidence

Resolution Algorithm
Application example (I)

- Is João José's son?

¬son(john,jose)

joao | joao, jose | jose

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

The word joao unites with joao; the word jose unites with jose;

Synthetic Intelligence Lab

Resolution Algorithm
Application example (I)

- Is João José's son?

¬son(john,jose)

joao | joao, jose | jose

¬(true)

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

Under the conditions established by the unifications, which used the first clause of $\mathcal{P}$, the initial question is reduced to the current one, because it is a fact

Synthetic Intelligence Lab

Resolution Algorithm
Application example (I)

- Is João José's son?

¬son(john,jose)

joao | joao, jose | jose

¬(true)

–|–

□

% son: Son,Father → {𝕍 , 𝔽}

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → {𝕍 , 𝔽}

father(P,F) :- son(F,P).

From which a contradiction can be drawn □ (regardless of any unifications)

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

```
% son: Son,Father → {𝕍,𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍,𝔽}

father( P,F ) :- son( F,P ).
```

You want to know if José is João's father

Resolution Algorithm
Application example (II)

- Is José João's father?

  ¬father(jose,joao)

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

In logical terms, the question is described as follows

Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

  ¬father(jose,joao)

```
% son: Son,Father → {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

Of all the clauses in the $\mathcal{P}$ programme, only those that offer conclusions about the father/2 predicate will contribute to the development of the test

Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

  ¬father(jose,joao)

```
% son: Son,Father →  {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son →  {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

The only clause of the father/2 predicate that has been ticked will be used in the first step of developing the (proof) tree

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

  ¬father(jose,joao)

  jose | P, joao | F

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

The term jose unifies with the variable P; the term joao unifies with the variable F;

Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

¬father(jose,joao)

jose | P, joao | F

¬child(john,jose)

% son: Son,Father • {?,•}

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son • {?,•}

father(P,F) :- son(F,P).

To continue the (tree of) proof, the initial question is reduced to proof of the conditions of the clause used to make the unifications

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

¬father(jose,joao)

jose | P, joao | F

¬son(john,jose)

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

Of all the clauses in the programme $\mathcal{P}$, only those that offer conclusions about the predicate son/2 will contribute to the development of the test

Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

¬father(jose,joao)

jose | P, joao | F

¬son(john,jose)

```
% son: Son,Father → {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

Of the three clauses listed, the first will be used in the first step of developing the (tree of) evidence

ISLab
Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

¬father(jose,joao)

jose | P, joao | F

¬son(john,jose)

joao | joao, jose | jose

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

The word joao unites with joao; the word jose unites with jose;

Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

¬father(jose,joao)

jose | P, joao | F

¬son(john,jose)

joao | joao, jose | jose

¬(true)

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

Under the conditions established by the unifications, which used the first clause of $\mathcal{P}$, the previous question is reduced to the current one, because it is a fact

Synthetic Intelligence Lab

Resolution Algorithm
Application example (II)

- Is José João's father?

¬father(jose,joao)

jose | P, joao | F

¬child(john,jose)

joao | joao, jose | jose

¬(true)

–|–

□

% son: Son,Father → {𝕍 , 𝔽}

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → {𝕍 , 𝔽}

father(P,F) :- son(F,P).

From which a contradiction can be drawn □ (regardless of any unifications)

Resolution Algorithm
Application example (III)

- Is José John's son?

```
% son: Son,Father → {𝕍 , 𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍 , 𝔽}

father( P,F ) :- son( F,P ).
```

Synthetic Intelligence Lab

Resolution Algorithm
Application example (III)

- Is José John's son?

```
% son: Son,Father → {𝕍,𝔽}

son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).

% father: Father,Son → {𝕍,𝔽}

father( P,F ) :- son( F,P ).
```

Let's take the question posed above

Resolution Algorithm
Application example (III)

- Is José John's son?

  ¬son(jose,joao)

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

In logical terms, the question is described as follows; of the three clauses in the predicate son/2, the first will be used to develop the (tree of) evidence

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose | ~~joao~~, ...

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

The term jose does not unify with joao because they are two different constants, so the proof procedure cannot evolve through this branch of proof

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose | ~~joao~~, ...

% son: Son,Father $\rightarrow \{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son $\rightarrow \{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

Abandoning the first clause, the second is taken as an alternative to continue the development of the (tree of) proof;

ISLab

Synthetic Intelligence Lab

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose | ~~joao~~, ...

jose | jose, joao | ~~manuel~~

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
<u>son(jose,manuel).</u>
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

The term jose unifies with jose; the term joao does not unify with manuel because they are two different constants, so the proof procedure cannot evolve through this branch of proof

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose | ~~joao~~, ...

jose | jose, joao | ~~manuel~~

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
<u>son(carlos,jose).</u>

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

Also abandoning the second clause, the third (and last) is taken as an alternative to give continuity to the development of the (tree of) proof

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose ~~| joao~~, ...

jose | jose, joao ~~| manuel~~

jose ~~| carlos~~, ...

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

father(P,F) :- son(F,P).

The term jose does not unify with carlos because they are two different constants, so the proof procedure cannot evolve through this branch of proof

Synthetic Intelligence Lab

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose | ~~joao~~, ...

jose | jose, joao | ~~manuel~~

jose | ~~carlos~~, ...

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

```
son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).
```

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

```
father( P,F ) :- son( F,P ).
```

As there are no more clauses in the son/2 predicate that can be considered as alternatives in the (proof) tree, the procedure of applying the resolution algorithm ends without reaching any contradiction

ISLab

Synthetic Intelligence Lab

Resolution Algorithm
Application example (III)

- Is José John's son?

¬son(jose,joao)

jose ~~| joao~~, ...

jose | jose, joao ~~| manuel~~

jose ~~| carlos~~, ...

~~Is José John's son?~~

% son: Son,Father → $\{\mathbb{V},\mathbb{F}\}$

son(joao,jose).
son(jose,manuel).
son(carlos,jose).

% father: Father,Son → $\{\mathbb{V},\mathbb{F}\}$

father(P,F) :- son(F,P).

This means that the initial question doesn't generate any contradiction at $\mathcal{P}$, so it's false.

Synthetic Intelligence Lab

Resolution Algorithm
Application example (IV)

¬ son(X,jose)

What is the meaning of this question?

Develop the evidence tree for this question

% son: Son,Father → $\{\mathbb{V}, \mathbb{F}\}$

```
son( joao,jose ).
son( jose,manuel ).
son( carlos,jose ).
```

% father: Father,Son → $\{\mathbb{V}, \mathbb{F}\}$

```
father( P,F ) :- son( F,P ).
```

ISLab
Synthetic Intelligence Lab

factorial(0,1):-!
factorial(N,F):-N1 is N-1,factorial(N1,F1),F is N*F1.

Let's see what happens when you call ?- factorial(3,F).

F=6

factorial(0,1) failure

factorial(3,F):-N1 ⟵ 3-1,factorial(2,F1) , F is 3*2. success (w/ F ⟵ 6)

factorial(0,1) failure

factorial(2,F):-N1 ⟵ 2-1,factorial(1,F1) , F is 2*1. success (w/ F ⟵ 2)

factorial(0,1) failure

factorial(1,F):-N1 ⟵ 1-1,factorial(0,F1) , F is 1*1. success (w/ F ⟵ 1)

factorial(0,1):-!. success

- These are systems that
  - They use explicitly represented knowledge to solve complex problems;
  - They manipulate knowledge and information;
  - They have "encrusted" the ability to reason:
    - The ability to define a set of steps for efficiently solving a problem;
    - The very mechanism of inference is knowledge.

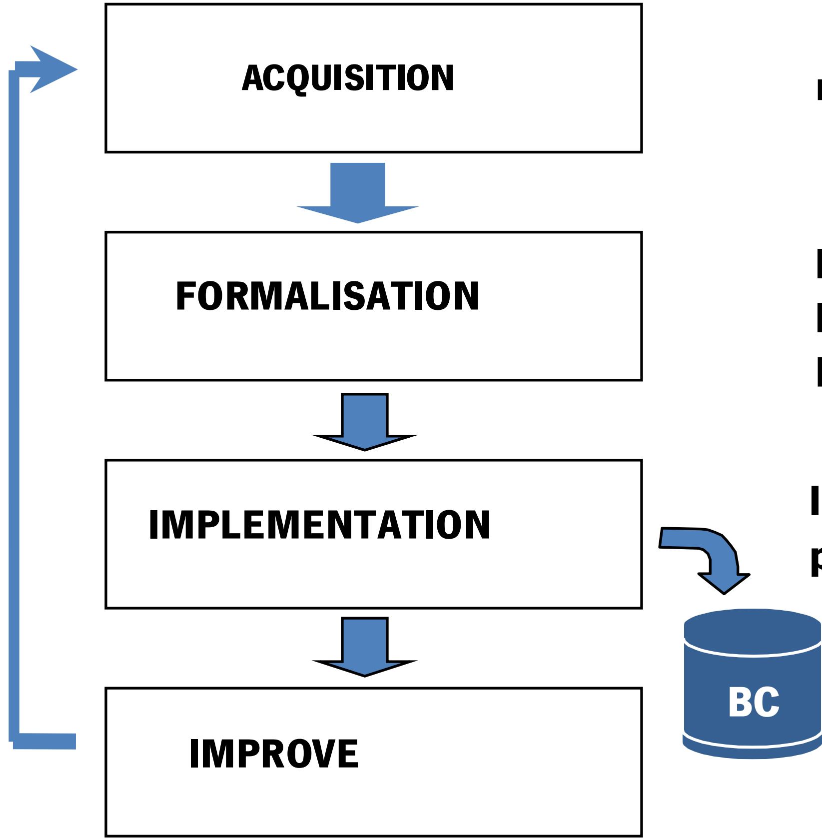
Synthetic Intelligence Lab

Knowledge-Based Systems

Level of Expertise

Logic Level

Level of Implementation



natural language

language of language knowledge

languages programme

ISLab
Synthetic Intelligence Lab

Types of Reasoning

- **Deduction**
  - facts + inference rules => new facts
  - cause → effect
    - If there is fire (cause), there is smoke (effect). There's fire here, so there's smoke here (new fact)
  - It's the only type of inference that holds true
    - *truth-preserving*
- **Abduction**
  - reverse deduction: from effect to cause
    - If there's smoke, there's fire. I saw smoke (effect), so there's fire here (cause)
  - This type of inference maintains the falsehood
  - We have generated a possible explanation that will have to be proven

ISLab
Synthetic Intelligence Lab

Types of Reasoning

- **Induction**
  - part of the facts to generate (new) rules
    - fact1 + fact2 + fact3 → rule!
    - ex. Mr Joaquim, like Mrs Isabel, has the flu and a headache, so everyone who has the flu has a headache
  - It transforms facts (knowledge in extension) into knowledge in the form of a hypothesis!
- **Analogue**
  - facts + similarities + adaptation rules
  - from facts, from the similarity between them, solves the problem without generating rules
    - e.g.: In the previous case of flu, I took an aspirin and it didn't work, so I'm not going to take aspirin in this similar case

ISLab
Synthetic Intelligence Lab

Reasoning in practice

- Deduction and Abduction
  - They are used in knowledge-based agents
- Induction and Analogy
  - They are used in machine learning

- Deduction: there are two large groups
  - Logic and the like
  - Treatment of uncertainty (e.g. probabilistic or *fuzzy*)

Prolog implementations

SWI-Prolog - A Free Software Prolog environment, licensed under the Lesser GNU public licence. This popular interpreter was developed by Jan Wielemaker. This is the interpreter we used while developing this book.
http://www.swi-prolog.org/

SICStus Prolog - Industrial strength Prolog environment from the Swedish Institute of Computer Science. http://www.sics.se/sicstus/

GNU Prolog - Another more widely used free Prolog compiler developed by Daniel Diaz. http://www.gprolog.org

YAP Prolog - A Prolog compiler developed at the University of Porto and Federal University of Rio de Janeiro. Free for use in academic environments.
http://www.ncc.up.pt/~vsc/Yap/

Synthetic Intelligence Lab

Recommended Bibliography

- Ivan Bratko, PROLOG: Programming for Artificial Intelligence, 4th Edition, Addison-Wesley Longman Publishing Co., Inc., ISBN-13: 978-0321417466, 2011.

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Logic Programming

BACHELOR'S DEGREE IN COMPUTER ENGINEERING
integrated master's degree in computer engineering
Artificial Intelligence
2025/26