Universidade do Minho
Escola de Engenharia
Departamento de Informática

Representação do Conhecimento e Raciocínio

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

Synthetic Intelligence Lab

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

ISLab
Synthetic Intelligence Lab

Knowledge and Reasoning

- What is knowledge?

  Knowledge can be defined as information about the environment (which can be expressed in the form of propositions).

- What is knowledge representation?

  Symbols used to represent information about the environment (propositions).

- What is knowledge representation and reasoning?

  The manipulation of symbols (which encode propositions to produce representations of new propositions).

The issue of representing knowledge is a fundamental question in Artificial Intelligence: How can human knowledge be represented by a computer language and in such a way that computers can use this knowledge to reason?

Source: Grosan C., Abraham A. (2011) Knowledge Representation and Reasoning. In: Intelligent Systems. Intelligent Systems Reference Library, vol 17. Springer, Berlin, Heidelberg

Sentence

- An assertion about the world in a language of knowledge representation.

- A language with concrete and consistent rules
  - No ambiguity in the representation;
  - Allows unambiguous communication and processing;
  - Very different from languages (e.g. Portuguese).
- Many ways to translate between languages
  - A statement can be represented in different logics;
  - And perhaps in a different way in the same logic;
- The expressiveness of a logic
  - How much can we say in this language?
- Not to be confused with logical reasoning
  - Logics are languages, reasoning is a process (which can use logic).

Syntax and Semantics

- Syntax
  - Rules for constructing admissible sentences in logic;
  - Which symbols can be used (e.g. Portuguese: letters, punctuation marks);
  - How can we combine these symbols?
- Semantics
  - How to interpret (read) sentences in logic;
  - Assign a meaning to each sentence;
- Example: "All the students are 19 years old"
  - A valid sentence (syntax);
  - From which we can understand the meaning (semantics);
  - This sentence is (however) false (there is a counterexample).

Propositional logic

- **Syntax**
  - Propositions, for example "It's raining";
  - Connectors : and, or, not, implies, iff (equivalent);

    $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
  - In brackets, V (true) and F (false);
- **Semantics**
  - Define how connectors determine truthfulness;
    - "P and Q" is true if and only if P is true and Q is true;
  - We use truth tables to discover the veracity of statements.

ISLab
Synthetic Intelligence Lab

Predicate logic

- Propositional logic combines atoms
  - An atom does not contain propositional operators;
  - It has no structure (hoje_chove, paulo_gosta_raquel);
- Predicates allow us to talk about objects
  - Properties: it's raining (today);
  - Relations: likes (paulo, raquel);
  - True or false;
- In predicate logic, each atom is a predicate
  - First-order logic, higher-order logic.

Synthetic Intelligence Lab

First Order Logic

- **Constants** are objects: paulo, grapes;
- **Predicates** are properties and relationships:
    - tastes (paulo, grapes)
- **Functions** transform objects:
    - tastes (paulo, apanha(videira))
- **Variables** represent any object: tastes (X, grapes)
- **Quantifiers** quantify the values of variables
    - True for all objects (Universal): $\forall$ X. tastes(X, grapes)
    - There is at least one object (Existential): $\exists$ X. tastes(X, grapes)

Synthetic Intelligence Lab

Other routes

- Higher-order logic
  - They allow quantification of more general things, such as relationships between relationships.
- Multi-value logics
  - More than two truth values (e.g. Extended Logic Programming)
    - e.g. true, false and unknown
  - **Fuzzy** logic uses probabilities, truth value in [0,1]
- Modal logic
  - Modal operators define mode for propositions
  - **Epistemic logics** (belief)
    - e.g. ☐ p (necessarily p), ◊p (possibly p), ...
  - **Temporal logics** (time)
    - e.g. ☐ p (always p), ◊p (eventually p), ...

Logic is an excellent way of representing

- Easy to translate when possible;
- There are several dedicated branches of maths;
- Allows you to develop logical thinking;
- Basis for programming languages
  - Prolog uses a subset of first-order logic.

Exercises

Represent the following sentences in first-order logic:

a) There's a student who failed History
b) Only one student failed History
c) Not all students registered for Knowledge Representation and Distributed Systems at the same time
d) Only one student failed History and Biology
e) They can fool some of the people all of the time or fool all of the people some of the time, but they can't fool all of the people all of the time

Source: Luís Paulo Reis, AI: Lecture 6: Knowledge Engineering, University of Porto, 2019.

Solved exercises

a) There is a student who failed History

∃x(Student(x)∧Failed(x, History)

b) Only one student failed History

∃x(Student(x)∧Failed(x, History)∧∀y(Failed(y, History)⇒x=y))

c) Not all students enrolled in Knowledge-Based Systems and Distributed Systems at the same time.

∃x(Student(x)∧¬(Take(x, SBC)∧Take(x, SD)) or

∃x(Student(x)∧(¬Take(x, SBC)∨ ¬Take(x, SD)))

d) Only one student failed History and Biology

∃x(Student(x)∧Failed(x, History)∧Failed(x, Biology)∧
(∀y((Failed(y, History)∧Failed(y, Biology))⇒x=y)))

e) They can fool some of the people all of the time or fool all of the people some of the time, but they can't fool all of the people all of the time.

∀ he(x) ⇒ ((∃y∀t(Time(t)∧(Person(y))⇒Enganar(x, y, t)) ∨
(∃t∀y(Time (t)∧Person(y))⇒ Deceive(x, y, t)) ∧
¬∀x∀t(Time (t)∧Person(y)⇒ Cheat(x, y, t))

Source: Luís Paulo Reis, AI: Lecture 6: Knowledge Engineering, University of Porto, 2019.

Computational Approaches

- Declarative Systems
  - Knowledge about facts and relationships in the world must be explicitly codified in a way that allows us to "reason" about this knowledge.
- Procedural Systems
  - They allow knowledge to be represented on the basis of facts and rules (If <condition> Then <action>). Procedural knowledge reflects an incremental process until a certain goal is reached.
- Hybrid systems
  - They combine declarative and procedural aspects.
- Other possible approaches:
  - Connectionist systems;
  - Biological systems.

Source: E.Costa, A.Simões, Artificial Intelligence - Fundamentals and Applications, 2008.

ISLab
Synthetic Intelligence Lab

Production rules

- Set of pair rules <condition, action>
- **"If condition then action"**
  - Modular
  - Easier expansion
  - Activation by system status
  - Close to the cognitive model

Conditions imply Conclusion

Condition_1 and Condition_2 and ... $\text{Condition}_n \Rightarrow$ Conclusion

- Areas of application: diagnosis and detection of diseases and malfunctions, counselling and recommendation ...

Example:

if the patient has a fever and the patient has pain and
the patient has no detectable infections
then diagnose flu

ISLab
Synthetic Intelligence Lab

Advantages of Production Rules

- Production Rules offer a natural way of expressing knowledge.
  - **Modularity** - each rule defines a part of the knowledge and is independent
  - **Incrementality** - new rules can be added at any time
  - **Changeability** - the rules can be alter**ed** at any time
  - **Independence of** the inference system used
  - Ease of generating **explanations** for a given answer:
    - **How did** you arrive at a given conclusion? (questions like)
    - **Why** are we interested in this information? (why questions)

ISLab
Synthetic Intelligence Lab

Use of Production Rules

- The Knowledge Base is made up of rules and facts;
- Consider the following logical syntax:
  - **if** Condition **then** Conclusion
- A Condition can be:
  - a logical predicate (e.g. Prolog)
  - a conjunction of two conditions: Cond1 **and** Cond2
  - a disjunction of two conditions: Cond1 **or** Cond2
- Representing facts (data): fact(X)
  - X is something that is currently true

ISLab
Synthetic Intelligence Lab

Example
Family Tree

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

if son(X,Y) then descendant(X,Y).
if son(X,Z) and descendant(Z,Y) then descendant(X,Y).

if son( SON,FATHER )
   then father( FATHER,SON ).

if son( NETO,X ) and son( X,AVO )
   then grandfather (grandfather, grandson).
```

Synthetic Intelligence Lab

- **Backward chaining**
  - From a question (hypothesis), reasoning moves backwards in the chain of inference to the facts that support it;
  - let's go from conclusions to conditions.
- **Forward chaining**
  - all (exhaustively) provable (derived) conclusions are entered into the knowledge base as facts;
  - with all the facts represented in the knowledge base, it is "enough" to prove (confirm) the existence of the conclusion.

Backward chaining

```
demo( QUESTION ) :-
    fact( QUESTION ).
demo( QUESTION ) :-
    (if CONDITION then QUESTION),
    demo( CONDITION ).
demo( QUESTION1 and QUESTION2 ) :-
    demo( QUESTION 1 ),
    demo( QUESTIONO2 ).
demo( QUESTION1 or QUESTION2 ) :-
    demo( QUESTION 1 ).
demo( QUESTION1 or QUESTION2 ) :-
    demo( QUESTIONO2 ).
```

Demo procedure(Question)
where Question is the hypothesis to be tested

Synthetic Intelligence Lab

Forward chaining

```
demo:- derive,
        listing(fact).

derive :-
    demo2( X ),
    derive.
derive.

demo2( CONCLUSION ) :-
    (if CONDITION then CONCLUSION),
    composition( CONDICAO ),
    no( fact( CONCLUSION ) ),
    assert( fact( CONCLUSION ) ).
```

```
composition( CONDITION ) :-
    fact( CONDITION ).
composition( QUESTION1 and QUESTION2 ) :-
    composition( QUESTION1 ),
    composition( QUESTIONO2 ).
composition( QUESTION1 or QUESTION2 ) :-
    composition( QUESTION1 ).
composition( QUESTION1 or QUESTION2 ) :-
    composition( QUESTIONO2 ).
```

Examples

fact('wet corridor').

fact('dry wc').

fact('door closed').

if 'dry garage' and 'wet corridor'

then 'toilet leak'.

if 'wet corridor' and 'dry toilet'

then 'problems in the garage'.

if 'door closed' or 'no rain'

so 'no water from outside'.

if 'problems in the garage' and 'no water outside'

then 'escape in the garage'.

Backward chaining" version

| ?- demo('garage leak').

yes

Forward Chaining version:

fact('wet corridor').

fact('dry toilet').

fact('door closed').

fact('problems in the garage').

fact('there is no water outside').

fact('leak in the garage').

ISLab
Synthetic Intelligence Lab

With explanation generation

```
demo( Q, facto(Q)) :-
    fact( Q ).
demo( Q, rule(Q) because Exp ) :-
    ( if C then Q ),
    demo( C, Exp ).
demo( ( C1 and C2 ), Exp1 and Exp2 ) :-
    nonvar( C1 ), nonvar( C2 ),
    demo( C1, Exp1 ),
    demo( C2, Exp2).
demo( ( C1 or C2 ), Exp1 or Exp2) :-
    nonvar( C1 ), nonvar( C2 ),
    demo( C1, Exp1),
    demo( C2, Exp2).
demo( ( C1 or C2 ), Exp1 ) :-
    nonvar( C1 ), nonvar( C2 ),
    demo( C1, Exp1 ),
    nao( demo( C2, Exp2 ) ).
demo( ( C1 or C2 ), Exp2 ) :-
    nonvar( C1 ), nonvar( C2 ),
    nao( demo( C1, Exp1 ) ),
    demo( C2, Exp2 ).
```

Example

- Based on the previous example.

```
| ?- demo('garage escape', Exp).
Exp = rule('leak in garage')because(rule('problems in garage')because
fact('corridor wet')and fact('toilet dry'))and(rule('no water
outside')because fact('door closed'))
```

Uncertainty in Production Rules

- In the systems presented above, the information is either true or false;
- Intermediate values (e.g. false, unlikely, probable, highly probable, true) are not considered;
- In the real world, this is unrealistic;
- Systems need to be able to deal with uncertainty (e.g. the degree of risk, probability, confidence);

- We associate each proposition with a degree of confidence.

  Facts: fact(Proposition) :: C.

  Rules: if Condition then Action :: C

  where C is a number between 0 and 1. (0-probability 0%; 1: probability 100%)

Synthetic Intelligence Lab

Production rules with degrees of confidence

```
:- op(800,fx,se).
:- op(700,xfx,then).
:- op(500,xfy,or).
:- op(400,xfy,e).
:- op(900,xfx,::).
:- dynamic facto/2.
:- dynamic '::'/2.

demo( Q,G ) :-
    fact( Q ) :: G.
demo( Q,G ) :-
    ( if C then Q ) :: Gr,
    demo( C,Gc ),
    G is Gr * Gc.

demo( ( C1 and C2 ),G ) :-
    nonvar( C1 ), nonvar( C2 ),
    demo( C1,G1 ),
    demo( C2,G2 ),
    smaller( G1,G2,G ).
demo( ( C1 or C2 ),G ) :-
    nonvar( C1 ), nonvar( C2 ),
    demo( C1,G1 ),
    demo( C2,G2 ),
    greater( G1,G2,G ).
demo( ( C1 or C2 ),G1 ) :-
    nonvar( C1 ), nonvar( C2 ),
    demo( C1,G1 ),
    no( demo( C2,G2 ) ).
demo( ( C1 or C2 ),G2 ) :-
    nonvar( C1 ), nonvar( C2 ),
    no( demo( C1, G1 ) ),
    demo( C2,G2 ).
```

Synthetic Intelligence Lab

Example

```
fact(s)::1.
fact(vomit)::0.5.

(if nausea and vomiting
   then 'headache') :: 0.75.

(if vomiting and 'stomach ache'
   then drunk) :: 0.50.

(if 'headache' or drunkenness
   then 'intestinal problems') :: 0.35.

(if nausea and 'lower back pain' and 'kidney pain'
   then rheumatism) :: 0.20.
```

```
| ?- demo('headache',C).
C = 0.375 ?
```

ISLab
Synthetic Intelligence Lab

Pattern-Driven Programming

- Programming architecture based on data patterns that activate one or more modules;
- A standards-oriented programme is a set of modules;
- Each one is defined by a precondition and an action to be carried out whenever the problem data makes this precondition true.
- In this way, the execution of the modules is triggered by patterns in the data and, as in conventional systems, there is no pre-defined invocation scheme.

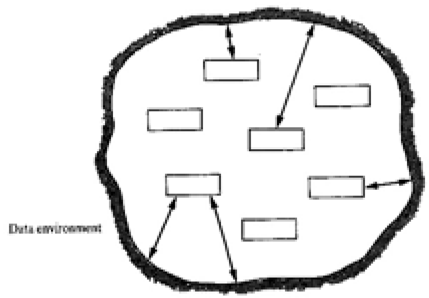
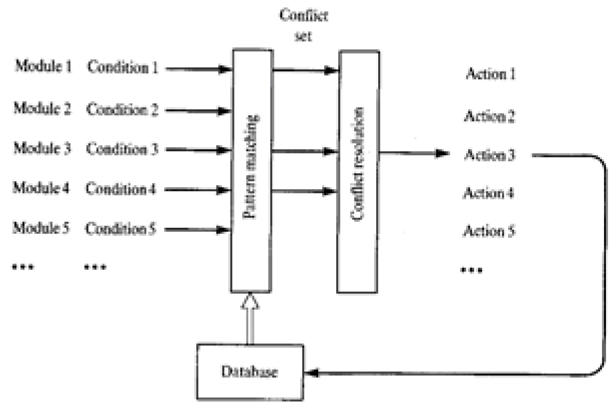


Figure 16.1 A pattern-directe



Source: Ivan Bratko, "PROLOG: Programming for Artificial Intelligence", 3rd Edition, Addison-Wesley Longman Publishing Co., Inc., 2000

Synthetic Intelligence Lab

Interpreter

```
demo :-
    Condition ==> Action,
    check( Condition ),
    execute( Action ).
demo.
check( [] ).
check( [Condition|Rest] ) :-
    call( Condition ),
    check( Remainder ).
execute( [stop] ).
execute( [] ) :-
    demo.
execute( [Action|Resto] ) :-
    call( Action ),
    execute( Resto ).
```

Example:
Maximum common divisor

```
mdc( X,Y,R ) :-
    X > Y,
    X1 is X-Y,
    mdc( X1,Y,R ).

mdc( X,Y,R ) :-
    Y > X,
    Y1 is Y-X,
    mdc( X,Y1,R ).

mdc( X,X,X ).
```

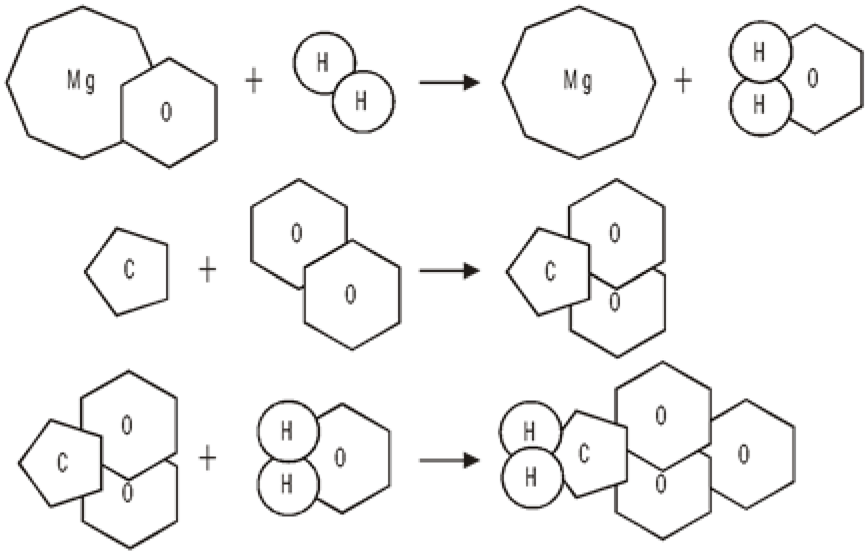
As Pattern:

```
[ num( X ), num( Y ), X > Y ] ==>
    [ N is X-Y, swap( num(X), num(N) ) ].
[ num( X ) ] ==>
    [ write( X ), stop ].
```

Synthetic Intelligence Lab

Example (2)

Chemical reactions involved in the production of the compound H2CO3.



[mol(mgo), mol(h2)] ==>
[consume(mol(mgo)), consume(mol(h2)),
produce(mol(mg)), produce(mol(h2o))].

[mol(c), mol(o2)] ==>
[consume(mol(c)), consume(mol(o2)),
produce(mol(co2))].

[mol(co2), mol(h2o)] ==>
[consume(mol(co2)), consume(mol(h2o)),
produce(mol(h2co3))].

ISLab
Synthetic Intelligence Lab

Hierarchical structures

The aim is to summarise the facts to be represented in a given system;

Entities can be grouped into classes, sharing values for the same attributes;

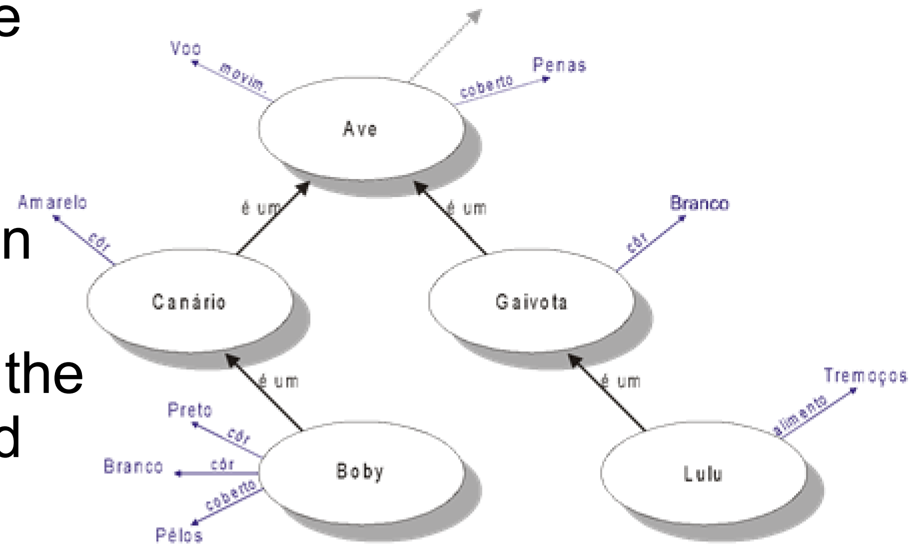
The facts associated with a given object may not be represented at its level, but rather be reconstructed through a process of inference by inheritance on higher classes.

- **Semantic networks**
  - correspond to a graph, where the nodes define the entities (objects, classes) of the system and the branches define relationships between them. Some of these relationships are called is_a relationships and allow for the inheritance of the knowledge defined in a given entity.
- **Frames**
  - *frames* define objects (or classes), each with a designation and a set of *slots*, corresponding to attributes, where a value is placed. Some of these attributes are used to represent the relationships between an object and a class to which it belongs and the relationships between classes and superclasses, which make it possible to inherit a value from a given unfilled *slot*.

ISLab
Synthetic Intelligence Lab

Semantic networks

- Network of organisations and the relationships between them;
- A graph:
  - each node corresponds to an entity;
  - the branches correspond to the relationships and are labelled with the name of the relationship
- Types of Relations: is_one, moves, covered, etc...



ISLab
Synthetic Intelligence Lab

Interpreter

```
demo( Agent,Question ) :-
    agent( Agent, Theory ),
    proof( question, theory).
demo( Agent,Question ) :-
    e_one( Agent,Entity ),
    demo( Entity,Question ).

proof( Question,[Question | Theory] ).
proof( Question,[X | Theory] ) :-
    proof( question, theory).
```

Demo implementation based on agent graphs;
Each agent has a set of properties and is represented by a node;
The existing relationships define the hierarchy of the system, with all branches defining *is_one* relationships.
Two different ways of answering a question:

- directly through the knowledge represented in the agent;
- through inference by inheritance.

Synthetic Intelligence Lab

Example

```
agent( bird,
        [ covered( feathers ),
          movement( flight ) ] ).
agent( canario,
        [ colour( yellow ),
          sound( chirp ) ] ).
agent( parrot,
        (bread),
          sound (speech),
          colour (green),
          colour( red )] ).
```

```
agent( boby,
        [ colour ( black ),
          colour( white ) ] ).
agent( lulu,
        (food),
          covered( by ) ] ).

e_one( canary,bird ).
e_one( parrot,bird ).
e_um( boby,canario ).
e_um( lulu,parrot ).
```

Directly through represented knowledge in the agent;

```
| ?- demo(parrot, food(X)).
X = bread ?
```

Through inference by inheritance.

```
| ?- demo(parrot, covered(X)).
X = feathers ?
```

ISLab
Synthetic Intelligence Lab

Frames

- A knowledge representation technique that attempts to organise concepts in a way that exploits interrelationships and common beliefs;
- Analogous to object-oriented programming;
- Data structure whose components are called slots;
- Slots are identified by names and denote information of various kinds: simple values, references to other frames, procedures, ...
- Is_a: relationship between Class and Superclass;
- Instance_of: member relation of a class;
- Representation: frame(Frame, Slot, Value).

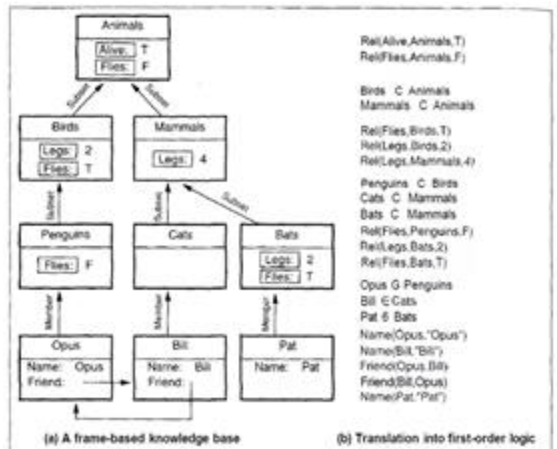


Figure 10.7 A frame-based network and a translation of the network into first-order logic. Boxed relation names in the network correspond to relations holding for all members of the set of objects.

Scripts

- A script is a data structure used to represent a sequence of events
- Scripts are used to interpret stories.
- Popular examples have been script-orientated systems that can interpret and extract facts from magazines.

- A script is characterised:
  1) A scene
  2) Props (objects manipulated in the script)
  3) Actors (agents who can change the state of the world).
  4) Events
  5) Acts: a set of actions by the actors.

In each scene, one or more actors perform actions. The actors act with the props. The script can be represented as a tree or network of states, driven by events.

Like *Frames*, scripts guide interpretation, telling the system what to look for and where to look. The script can predict events.

Example

The classic example is the restaurant script:

- Scene: A restaurant with an entrance and tables.
- Actors: The diners, servers, chef and Maitre d'Hotel.
- Props: The table setting, menu, table, chair.
- Acts: Entry, Seating, Ordering a meal, Serving a meal, Eating the meal, requesting the check, paying, leaving.

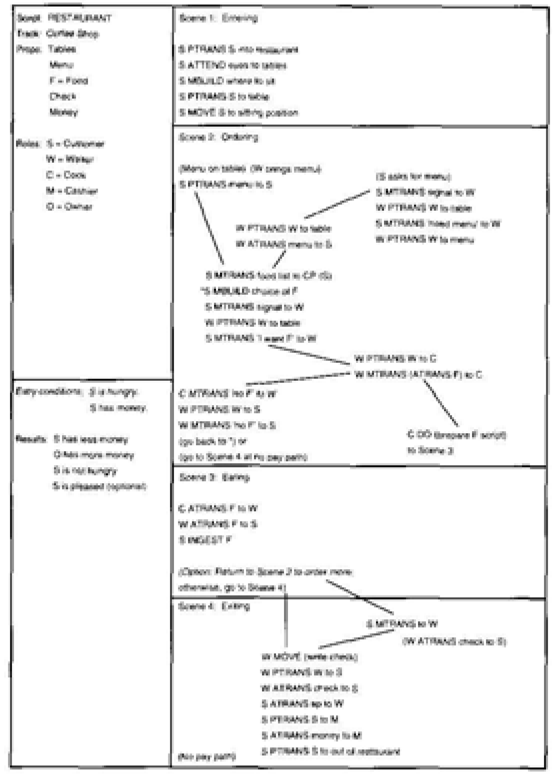
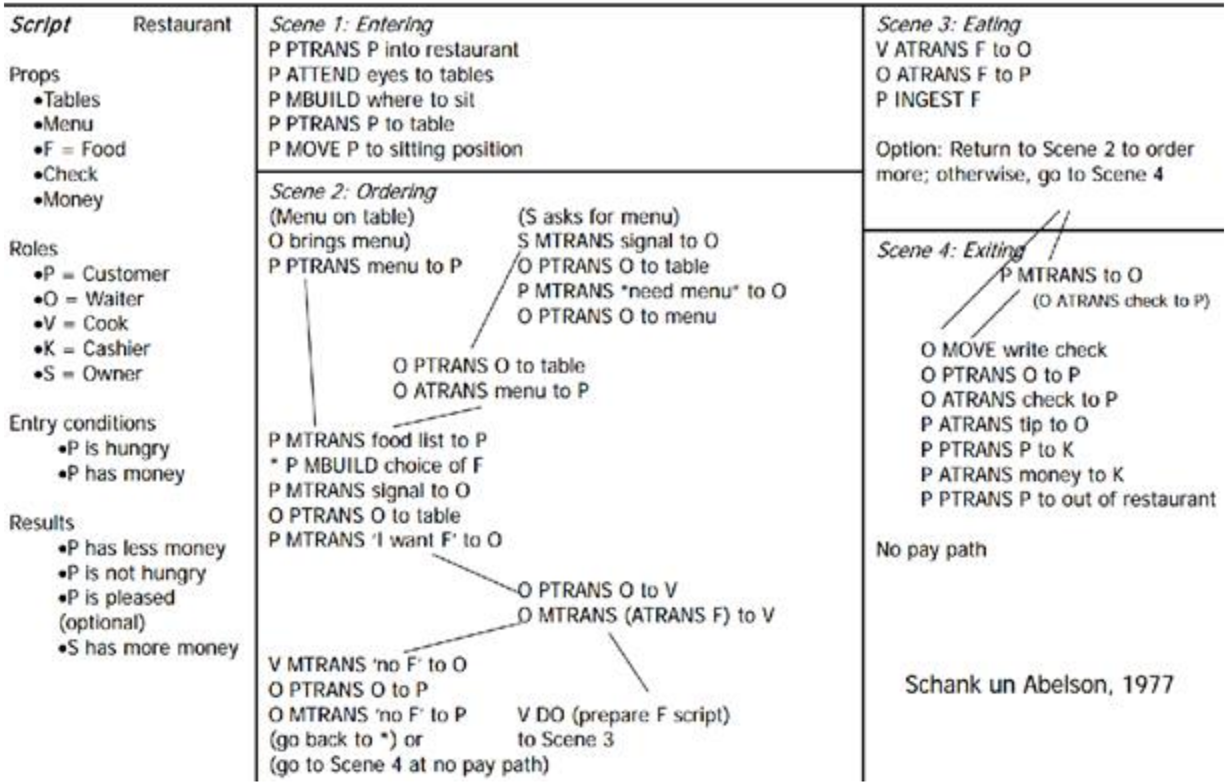


Figure 9.26 A restaurant script (Schank 1977).

Source: Philipp Koehn, Knowledge Representation, The Johns Hopkins University, 2019.

Synthetic Intelligence Lab

Example Continued



Source: Philipp Koehn, Knowledge Representation, The Johns Hopkins University, 2019.

ISLab
Synthetic Intelligence Lab

- Knowledge-Based Systems
  - Computer programmes that use **explicitly represented knowledge** to solve problems;
  - They manipulate knowledge and information intelligently;
  - They are designed to solve problems that require large amounts of human knowledge and specialisation (expertise).

- KNOWLEDGE + REASONING = PROBLEM SOLVING

ISLab
Synthetic Intelligence Lab

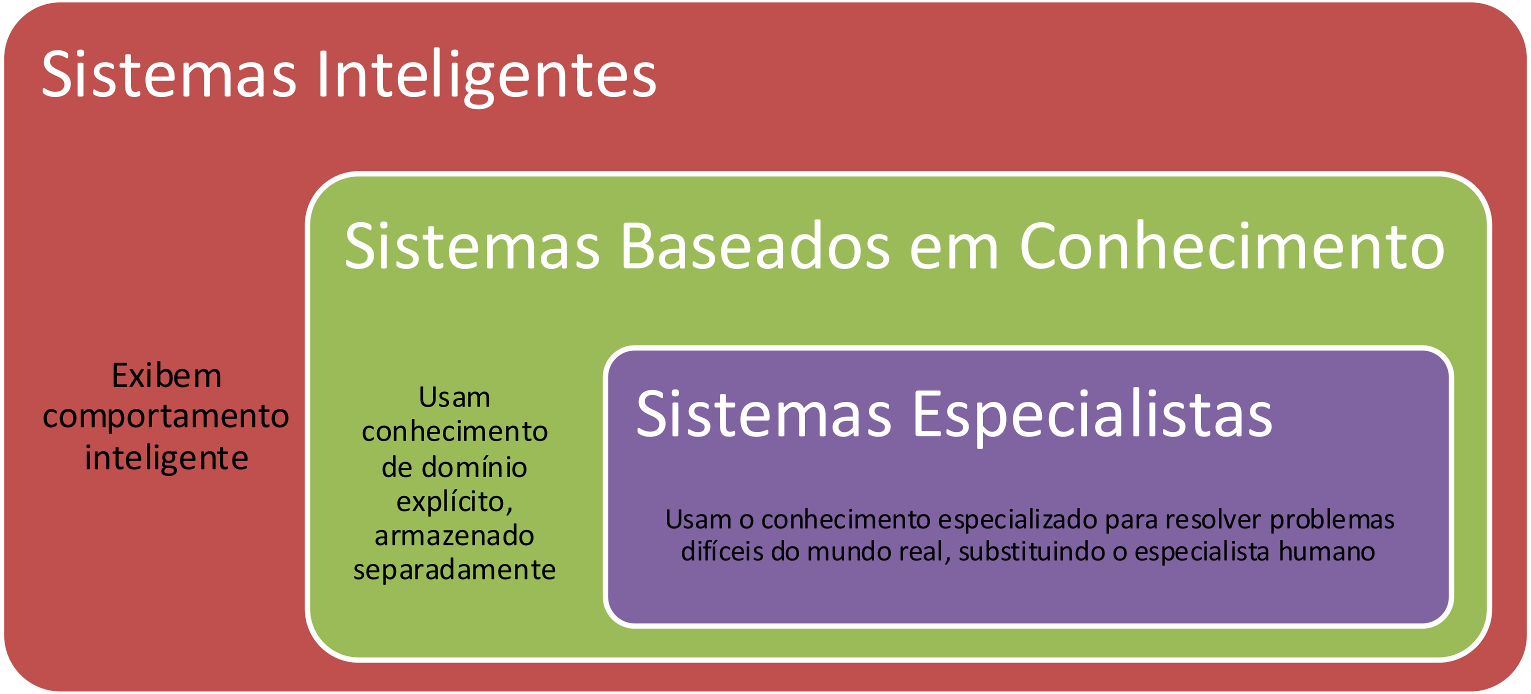
Definition

- Humanly Processable Knowledge Perspective
  - Analysis and modelling of the problem-solving method.
- Computer-processable symbolic perspective
  - The activity of representing this method through a computationally efficient formalism.
- Reasoning skills/Inference
  - It's the ability to define a set of steps to solve a problem efficiently and quickly;
  - The very mechanism of inference is knowledge.

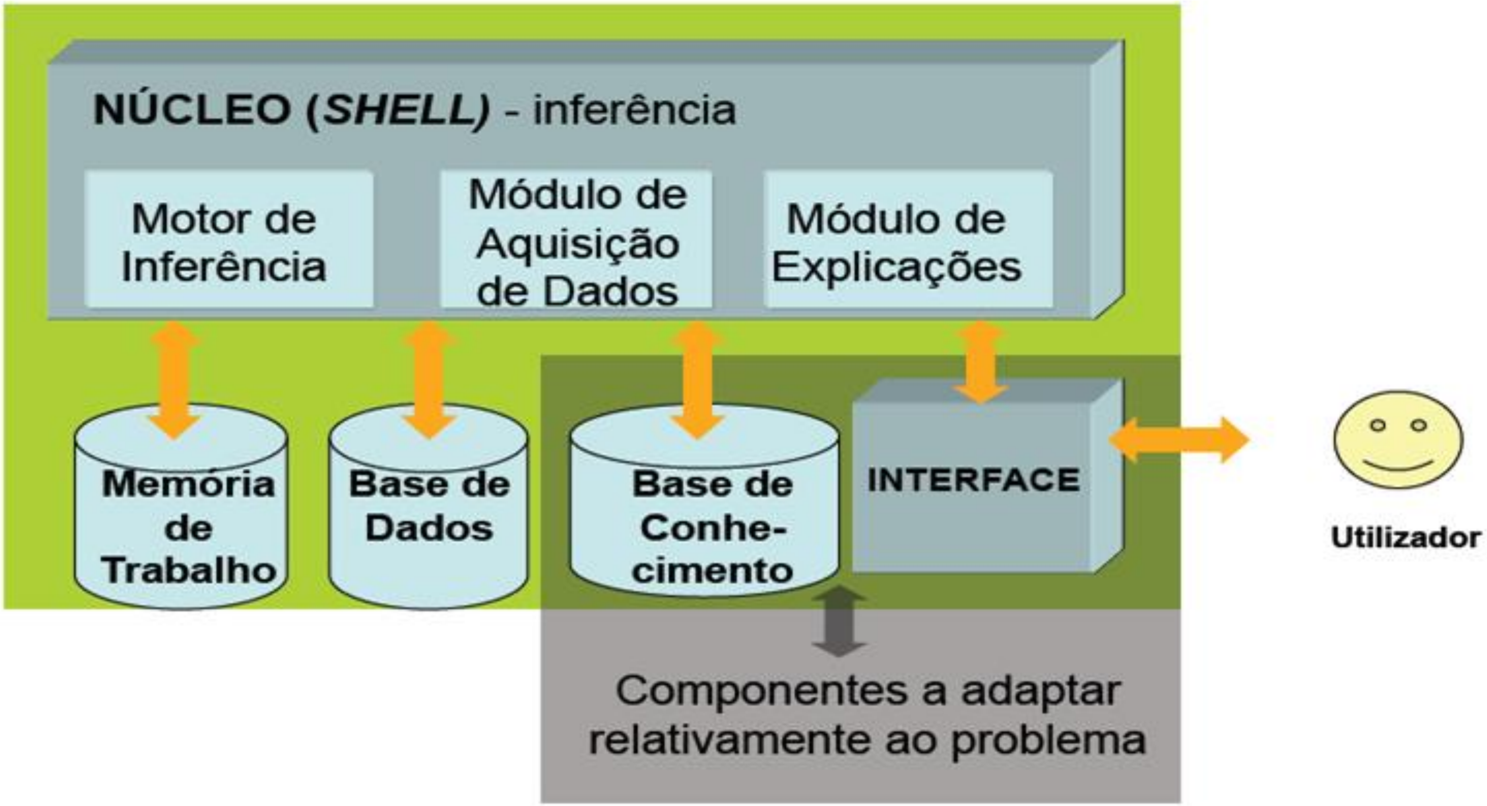
ISLab
Synthetic Intelligence Lab

Differences to other systems

Conventional systems	Knowledge-Based Systems
Data Structure	Representation of Knowledge
Data and relationships between Data	Concepts, Relationships between Concepts and Rules
Use Algorithms Deterministic	Search with Heuristics
Knowledge embedded in programme code	Knowledge represented explicitly and separately from programme that handles it and interprets
Explaining the reasoning is difficult	They can and should explain their reasoning


Sistemas Inteligentes
Exibem comportamento inteligente
Sistemas Baseados em Conhecimento
Usam conhecimento de domínio explícito, armazenado separadamente
Sistemas Especialistas
Usam o conhecimento especializado para resolver problemas difíceis do mundo real, substituindo o especialista humano

Structure



Source: Luís Paulo Reis, Artificial Intelligence (2019), University of Porto

ISLab
Synthetic Intelligence Lab

Structure

- **Shell**
  - Control of user interaction;
  - Knowledge inference;
  - Explanation of conclusions.
- **INFERENCE ENGINE/SYSTEM**
  - Development of reasoning based on the information obtained by the Data Acquisition Engine and the knowledge represented in the Knowledge Base.
  - For example, for production rules:
    - *Forward chaining*:
    - *Backward chaining*.

ISLab
Synthetic Intelligence Lab

Structure

- **DATA ACQUISITION MODULE**
  - Interaction with the user;
  - Obtaining information about the problem (questions to the user);
  - Checking the validity of the answers.
- **EXPLANATION MODULE**

  Justification of the conclusions reached:
  - Why - why MAD asked the user the question;
  - How - way of reasoning to reach conclusions;
  - Scenario study - What happens if some information provided by the user is changed (*what if);*
  - Why not - explain why a certain conclusion was not reached.

- **KNOWLEDGE BASE**
  - Description of the KNOWLEDGE needed to solve the problem;
  - A set of representations of actions and events in the world;
  - SENTENCES expressed in a particular LANGUAGE OF REPRESENTATION OF KNOWLEDGE:
    - Production rules;
    - Semantic Networks;
    - *Frames;*
    - Object-orientated;
    - Logic;
    - *Case Based Learning;*
    - Hybrids, ...

ISLab
Synthetic Intelligence Lab

Knowledge Base

Example of a CAUSE-EFFECT sentence (production rule)

- IF TEMP-PATIENT > 37.5 º C THEN PATIENT HAS FEVER

Example of GOAL-KNOWLEDGE - leads to the search for a solution:

- IF THE PATIENT IS AN ALCOHOLIC THEN LOOK FOR LIVER DISEASE FIRST
- SEEK THE SOLUTION FIRST IN WAYS WHERE THERE ARE FEW POSSIBILITIES (heuristics)

Attend to problems of:

- CONFLICTS / INCONSISTENCIES;
- INCOMPLETENESS / UNCERTAINTY.

Synthetic Intelligence Lab

Structure

- **DATABASE**
  - It is intended to contain the data/information that characterises the problem (facts).

- **WORKING MEMORY**
  - It allows the line of reasoning to be stored and provided;
  - Stores user responses (avoids repeated questions);
  - Storage of intermediate conclusions (avoids repetition of inferences).

Synthetic Intelligence Lab

INTERFACE

- Interaction between the SBC and the user;
- Language differs from that used to represent knowledge:
  - Natural language;
  - Visual Languages;
  - Diagrammatic languages;
  - Multimedia.
- Principles derived from cognitive and semiotic theories (*Human-Computer Interaction*):
  - Efficiency;
  - Dynamism;
  - Timely development.

TOOLS TO SUPPORT THE CONSTRUCTION OF A SBC:

- Use of programming languages such as LISP and PROLOG
- Support tools - various knowledge representation schemes, inference engines, interfaces, etc.
  - ART, Babylon, KEE, Knowledge Craft, Loops, Flex, Elements Environment, ...
- Shells - the interface and problem-solving strategy is predefined:
  - Insight, KES, MED2, M.1, Personal Consultant, S.1, Timm
  - EXSYS CORVID, CLIPS, JESS (Clips in Java), Jlisa (Clips for Java),

Source: Luís Paulo Reis, Artificial Intelligence (2019), University of Porto

Synthetic Intelligence Lab

Recommended Bibliography

- Stuart Russell and Peter Norvig, Artificial Intelligence - A Modern Approach, 4rd edition, ISBN: 978-0134610993, 2020.
- Inteligência Artificial-Fundamentos e Aplicações, E.Costa, A.Simões; FCA, ISBN: 978-972-722-340-4, 2008.
- 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

Representação do Conhecimento e Raciocínio

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