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

- Conhecimento e Raciocínio;
- Lógica e Programação em Lógica;
- Regras de Produção;
- Programação Dirigida aos Padrões;
- Estruturas hierárquicas:
  - Redes semânticas;
  - Frames;
- Scripts;
- Sistemas Baseados em Conhecimento.

Conhecimento e Raciocínio

- O que é conhecimento?

  O **conhecimento pode ser definido como informação sobre ambiente** (que pode ser expressa na forma de proposições).

- O que é representação do conhecimento?

  Símbolos usados para representar informação sobre ambiente (as proposições).

- O que é representação e raciocínio do conhecimento?

  A manipulação de símbolos (que codificam proposições para produzir representações de novas proposições).

A questão de representar o conhecimento é uma questão fundamental na Inteligência Artificial: Como pode o conhecimento humano ser representado por uma linguagem de computador e de uma tal forma que os computadores possam usar esse conhecimento para raciocinar?

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

Delaração (*Sentence*)

- Uma asserção sobre o mundo numa linguagem de representação do conhecimento.

- Uma linguagem com regras concretas e consistentes
  - Nenhuma ambiguidade na representação;
  - Permite comunicação e processamento inequívocos;
  - Muito diferente das línguas (ex. o Português).
- Muitas maneiras de traduzir entre línguas
  - Uma declaração pode ser representada em diferentes lógicas;
  - E talvez de maneira diferente na mesma lógica;
- A expressividade de uma lógica
  - Quanto podemos dizer nesta língua?
- Não confundir com raciocínio lógico
  - Lógicas são linguagens, o raciocínio é um processo (que pode usar lógica).

Sintaxe e Semântica

- Sintaxe
  - Regras para construir sentenças admissíveis na lógica;
  - Quais símbolos que se podem usar (ex., Português: letras, sinais de pontuação);
  - Como podemos combinar esses símbolos.
- Semântica
  - Como interpretar (ler) frases (sentenças) na lógica;
  - Atribuir um significado a cada frase;
- Exemplo: “Todos os alunos têm 19 anos”
  - Uma frase válida (sintaxe);
  - Da qual podemos entender o significado (semântica);
  - Esta frase é (no entanto) falsa (existe um contraexemplos).

- **Sintaxe**
  - Proposições, por exemplo "Está a chover“;
  - Conectores : and, or, not, implies, iff (equivalent);

    $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
  - Parênteses, V (verdadeiro) e F (falso);
- **Semântica**
  - Definir como de que forma os conectores determinam a veracidade;
    - "P e Q" é verdadeiro se e somente se P for verdadeiro e Q for verdadeiro;
  - Usamos tabelas de verdade para descobrir a veracidade das declarações

Lógica de predicados

- A lógica proposicional combina átomos
  - Um átomo não contém operadores proposicionais;
  - Não possui estrutura (hoje_chove, paulo_gosta_raquel);
- Os predicados permitem-nos discursar sobre objetos
  - Propriedades: chove (hoje);
  - Relações: gosta (paulo, raquel);
  - Verdadeiro ou falso;
- Na lógica de predicados, cada átomo é um predicado
  - Lógica de primeira ordem, lógicas de ordem superior.

Lógica de Primeira Ordem

- **Constantes** são objetos: paulo, uvas;
- **Predicados** são propriedades e relações:
  - gostos (paulo, uvas)
- **Funções** transformam objetos:
  - gostos (paulo, apanha(videira))
- **Variáveis** representam qualquer objeto: gostos (X, uvas)
- **Quantificadores** quantificam os valores das variáveis
  - Verdadeiro para todos os objetos (Universal): $\forall$X. gostos(X, uvas)
  - Existe pelo menos um objeto (Existencial): $\exists$X. gostos(X, uvas)

ISLab
Synthetic Intelligence Lab

Outras vias

- Lógicas de ordem superior
  - Permitem a quantificação sobre coisas mais gerais, tais como relações entre relações.
- Lógicas multi-valor
  - Mais de dois valores de verdade (ex., Programação em Lógica Estendida)
    - ex., verdade, falso e desconhecido
  - A lógica difusa **(Fuzzy logica)** usa probabilidades, valor de verdade em [0,1]
- Lógica modal
  - Operadores modais definem modo para proposições
  - **Epistemic logics** (crença)
    - e.g. $\Box p$ (necessarily p), $\Diamond p$ (possibly p), …
  - **Temporal logics** (tempo)
    - e.g. $\Box p$ (always p), $\Diamond p$ (eventually p), …

ISLab
Synthetic Intelligence Lab

A lógica é uma excelente forma de representar

- Fácil de fazer a tradução quando isso é possível;
- Existem diversos ramos da matemática dedicados;
- Permite desenvolver o raciocínio lógico;
- Base para linguagens de programação
  - Prolog usa um subconjunto Lógica de primeira ordem.

Exercícios

Representar as seguintes frases em lógica de primeira ordem:

a) Existe um estudante que chumbou a História
b) Só um estudante chumbou a História
c) Nem todos os estudantes se inscreveram simultaneamente a Representação do Conhecimento e a Sistemas Distribuídos
d) Só um estudante chumbou a História e a Biologia
e) Eles podem enganar algumas pessoas todo o tempo ou enganar todas as pessoas algum tempo, mas eles não podem enganar todas as pessoas todo o tempo

Fonte: Luís Paulo Reis, AI: Lecture 6: Knowledge Engineering, Universidade do Porto, 2019.

ISLab
Synthetic Intelligence Lab

Exercícios Resolvidos

a) Existe um aluno que chumbou a História

$\exists x(Student(x) \wedge Failed(x, History))$

b) Só um aluno chumbou a História

$\exists x(Student(x) \wedge Failed(x, History) \wedge \forall y(Failed(y, History) \Rightarrow x=y))$

c) Nem todos os estudantes se inscreveram simultaneamente a Sistemas Baseados em Conhecimento e Sistemas Distribuídos

$\exists x(Student(x) \wedge \neg(Take(x, SBC) \wedge Take(x, SD)))$ ou

$\exists x(Student(x) \wedge (\neg Take(x, SBC) \vee \neg Take(x, SD)))$

d) Só um aluno chumbou a História e a Biologia

$\exists x(Student(x) \wedge Failed(x, History) \wedge Failed(x, Biology) \wedge$
$(\forall y((Failed(y, History) \wedge Failed(y, Biology)) \Rightarrow x=y)))$

e) Eles podem enganar algumas pessoas todo o tempo ou enganar todas as pessoas algum tempo, mas eles não podem enganar todas as pessoas todo o tempo

$\forall ele(x) \Rightarrow ((\exists y \forall t(Tempo(t) \wedge (Pessoa(y)) \Rightarrow Enganar(x, y, t)) \vee$
$(\exists t \forall y(Tempo(t) \wedge Pessoa(y)) \Rightarrow Enganar(x, y, t)) \wedge$
$\neg \forall x \forall t(Tempo(t) \wedge Pessoa(y) \Rightarrow Enganar(x, y, t))$

Fonte: Luís Paulo Reis, AI: Lecture 6: Knowledge Engineering, Universidade do Porto, 2019.

ISLab
Synthetic Intelligence Lab

Abordagens Computacionais

- Sistemas Declarativos
  - O conhecimento sobre factos e relações no mundo devem ser codificados explicitamente de uma forma que permita a "raciocinar" sobre este mesmo conhecimento.
- Sistemas Procedimentais
  - Permitem representar o conhecimento com base em factos e regras (Se <condição> Então <ação>). O conhecimento procedimental reflete um processo incremental até chegar a um determinado objetivo.
- Sistemas híbridos
  - Combinam aspetos declarativos com procedimentais.
- Outras possíveis abordagens:
  - Sistemas conexionistas;
  - Sistemas biológicos.

Fonte: E.Costa, A.Simões, Inteligência Artificial - Fundamentos e Aplicações, 2008.

ISLab
Synthetic Intelligence Lab

Regras de Produção

- Conjunto de regras de pares <condição, ação>
- **"Se condição então ação"**
  - Modulares
  - Mais fácil expansão
  - Ativação pelo estado do sistema
  - Próximas do modelo cognitivo

Condições implicam Conclusão

Condição_1 e Condição_2 e … Condição_n $\Rightarrow$ Conclusão

- Áreas de aplicação: diagnóstico e deteção de doenças e avarias, aconselhamento e recomendação …

Exemplo:

se paciente tem febre e o paciente tem dores e
o paciente não tem infeções detetáveis
então diagnosticar gripe

ISLab
Synthetic Intelligence Lab

Vantagens das Regras de Produção

- As Regras de Produção oferecem uma forma natural de expressar conhecimento.
  - **Modularidade** - cada regra define uma parte do conhecimento, sendo independente
  - **Incrementabilidade** - novas regras podem ser acrescentadas em qualquer momento
  - **Alterabilidade** - as regras podem ser alteradas a qualquer momento
  - **Independência** do sistema de inferência utilizado
  - Facilidade em gerar **explicações** para uma dada resposta:
    - **Como** se chegou a uma dada conclusão? (questões como)
    - **Porquê** é que estamos interessados nesta informação? (questões porquê)

Utilização de Regras de Produção

- A Base de Conhecimento é feita de regras e fatos;
- Considera-se a seguinte sintaxe lógica:
  - **se** Condição **então** Conclusão
- Sendo que uma Condição pode ser:
  - um predicado lógico (e.g. Prolog)
  - uma conjunção de duas condições: Cond1 **e** Cond2
  - uma disjunção de duas condições: Cond1 **ou** Cond2
- Representar factos (dados): facto(X)
  - X é algo que atualmente é verdadeiro

ISLab
Synthetic Intelligence Lab

Exemplo
Árvore Genealógica

facto(filho(joao,jose)).
facto(filho(jose,manuel)).
facto(filho(carlos,jose)).

se filho(X,Y) entao descendente(X,Y).

se filho(X,Z) e descendente(Z,Y) entao descendente(X,Y).

se filho(FILHO,PAI)
  entao pai(PAI,FILHO).

se filho(NETO,X) e filho(X,AVO)
  entao avo(AVO,NETO).

- **Backward chaining**
  - de uma questão (hipótese) o raciocínio retrocede na cadeia de inferência até aos factos que a suportam;
  - vamos das conclusões às condições.
- **Forward chaining**
  - todas as conclusões (exaustivamente) possíveis de se provar (derivadas) são inseridas na base de conhecimento como factos;
  - com todos os factos representados na base de conhecimento, “basta” provar (confirmar) a existência da conclusão.

Backward chaining

```
demo( QUESTAO ) :-
   facto( QUESTAO ).
demo( QUESTAO ) :-
   (se CONDICAO entao QUESTAO),
   demo( CONDICAO ).
demo( QUESTAO1 e QUESTAO2 ) :-
   demo( QUESTAO1 ),
   demo( QUESTAO2 ).
demo( QUESTAO1 ou QUESTAO2 ) :-
   demo( QUESTAO1 ).
demo( QUESTAO1 ou QUESTAO2 ) :-
   demo( QUESTAO2 ).
```

Procedimento demo(Questão)
onde Questão é a hipótese a comprovar

ISLab
Synthetic Intelligence Lab

Forward chaining

```
demo:- derivar,
      listing(facto).

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

demo2( CONCLUSAO ) :-
   (se CONDICAO entao CONCLUSAO),
   composicao( CONDICAO ),
   nao( facto( CONCLUSAO ) ),
   assert( facto( CONCLUSAO ) ).
```

```
composicao( CONDICAO ) :-
   facto( CONDICAO ).
composicao( QUESTAO1 e QUESTAO2 ) :-
   composicao( QUESTAO1 ),
   composicao( QUESTAO2 ).
composicao( QUESTAO1 ou QUESTAO2 ) :-
   composicao( QUESTAO1 ).
composicao( QUESTAO1 ou QUESTAO2 ) :-
   composicao( QUESTAO2 ).
```

Exemplos

```
facto( 'corredor molhado' ).
facto( 'wc seco' ).
facto( 'porta fechada' ).

se 'garagem seca' e 'corredor molhado'
  entao 'fuga no wc'.
se 'corredor molhado' e 'wc seco'
  entao 'problemas na garagem'.
se 'porta fechada' ou 'nao ha chuva'
  entao 'nao ha agua do exterior'.
se 'problemas na garagem' e 'nao ha agua do exterior'
  entao 'fuga na garagem'.
```

Versão “Backward chaining”

```
| ?- demo('fuga na garagem').
yes
```

Versão "Forward Chaining:

```
facto('corredor molhado').
facto('wc seco').
facto('porta fechada').
facto('problemas na garagem').
facto('nao ha agua do exterior').
facto('fuga na garagem').
```

ISLab
Synthetic Intelligence Lab

Com geração de explicações

```
demo( Q, facto(Q)) :-
   facto( Q ).
demo( Q, regra(Q) porque Exp ) :-
   ( se C entao Q ),
   demo( C, Exp ).
demo( ( C1 e C2 ), Exp1 e Exp2 ) :-
   nonvar( C1 ), nonvar( C2 ),
   demo( C1, Exp1 ),
   demo( C2, Exp2).
demo( ( C1 ou C2 ), Exp1 ou Exp2) :-
   nonvar( C1 ), nonvar( C2 ),
   demo( C1, Exp1),
   demo( C2, Exp2).
demo( ( C1 ou C2 ), Exp1 ) :-
   nonvar( C1 ), nonvar( C2 ),
   demo( C1, Exp1 ),
   nao( demo( C2, Exp2 ) ).
demo( ( C1 ou C2 ), Exp2 ) :-
   nonvar( C1 ), nonvar( C2 ),
   nao( demo( C1, Exp1 ) ),
   demo( C2, Exp2 ).
```

Exemplo

- Com base no exemplo anterior.

```
| ?- demo('fuga na garagem', Exp).
Exp = regra('fuga na garagem')porque(regra('problemas na
garagem')porque facto('corredor molhado')e facto('wc
seco'))e(regra('nao ha agua do exterior')porque facto('porta
fechada'))
```

ISLab
Synthetic Intelligence Lab

Incerteza em Regras de Produção

- Nos sistemas previamente apresentados a informação é do tipo verdade ou falso;
- Não são considerados valores intermédios (ex., falso, pouco provável, provável, altamente provável, verdadeiro);
- No mundo real isto é irrealista;
- Os sistemas têm de saber lidar com a incerteza (ex., o grau de risco, a probabilidade, a confiança);

- Associamos a cada proposição um grau de confiança.

  Factos: facto(Proposição) :: C.

  Regras: se Condição então Ação :: C

  sendo C um número entre 0 e 1. (0-probabilidade 0%; 1:probabilidade 100%)

ISLab
Synthetic Intelligence Lab

Regras de Producao com graus de confiança

```
:- op(800,fx,se).
:- op(700,xfx,entao).
:- op(500,xfy,ou).
:- op(400,xfy,e).
:- op(900,xfx,::).
:- dynamic facto/2.
:- dynamic '::'/2.
demo( Q,G ) :-
   facto( Q ) :: G.
demo( Q,G ) :-
   ( se C entao Q ) :: Gr,
   demo( C,Gc ),
   G is Gr * Gc.
```

```
demo( ( C1 e C2 ),G ) :-
   nonvar( C1 ), nonvar( C2 ),
   demo( C1,G1 ),
   demo( C2,G2 ),
   menor( G1,G2,G ).
demo( ( C1 ou C2 ),G ) :-
   nonvar( C1 ), nonvar( C2 ),
   demo( C1,G1 ),
   demo( C2,G2 ),
   maior( G1,G2,G ).
demo( ( C1 ou C2 ),G1 ) :-
   nonvar( C1 ), nonvar( C2 ),
   demo( C1,G1 ),
   nao( demo( C2,G2 ) ).
demo( ( C1 ou C2 ),G2 ) :-
   nonvar( C1 ), nonvar( C2 ),
   nao( demo( C1, G1 ) ),
   demo( C2,G2 ).
```

ISLab
Synthetic Intelligence Lab

Exemplo

```
facto(enjoos)::1.
facto(vomitos)::0.5.

(se enjoos e vomitos
   entao 'dor de cabeca') :: 0.75.

(se vomitos e 'dor de estomago'
   entao bebedeira) :: 0.50.

(se 'dor de cabeca' ou bebedeira
   entao 'problemas intestinais') :: 0.35.

(se enjoos e 'dores lombares' e 'dor de rins'
   entao reumatismo) :: 0.20.
```

```
| ?- demo('dor de cabeca',C).
C = 0.375 ?
```

ISLab
Synthetic Intelligence Lab

Programação Dirigida aos Padrões

- A arquitetura de programação baseada em padrões de dados que ativam um ou mais módulos;
- Um programa orientado a padrões é um conjunto de módulos;
- Cada um deles definido por uma pré-condição e uma acão a ser executada sempre que os dados do problema tornarem essa pré-condição verdadeira.
- Deste modo, a execução dos módulos é ativada por padrões existentes nos dados, não havendo, como acontece nos sistemas convencionais, um esquema pré-definido de invocação.

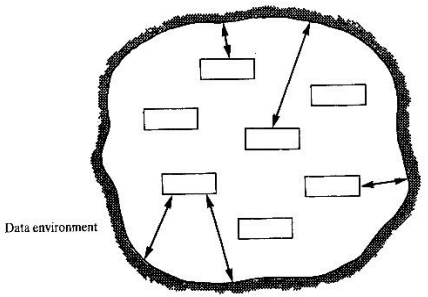
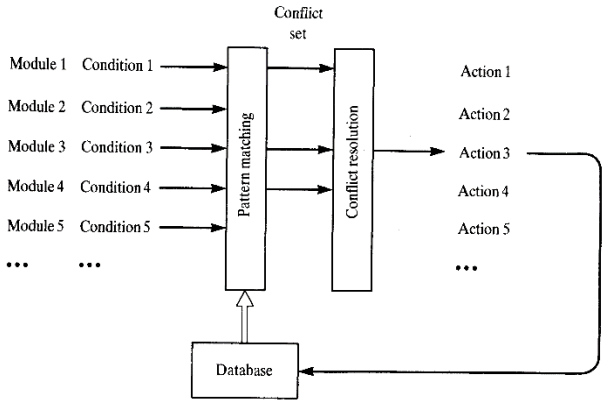


Figure 16.1 A pattern-directe



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

Interpretador

```
demo :-
   Condicao ==> Accao,
   verificar( Condicao ),
   executar( Accao ).
demo.
verificar( [] ).
verificar( [Condicao|Resto] ) :-
   call( Condicao ),
   verificar( Resto ).
executar( [parar] ).
executar( [] ) :-
   demo.
executar( [Accao|Resto] ) :-
   call( Accao ),
   executar( Resto ).
```

Exemplo:
Máximo divisor Comum

```
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 ).
```

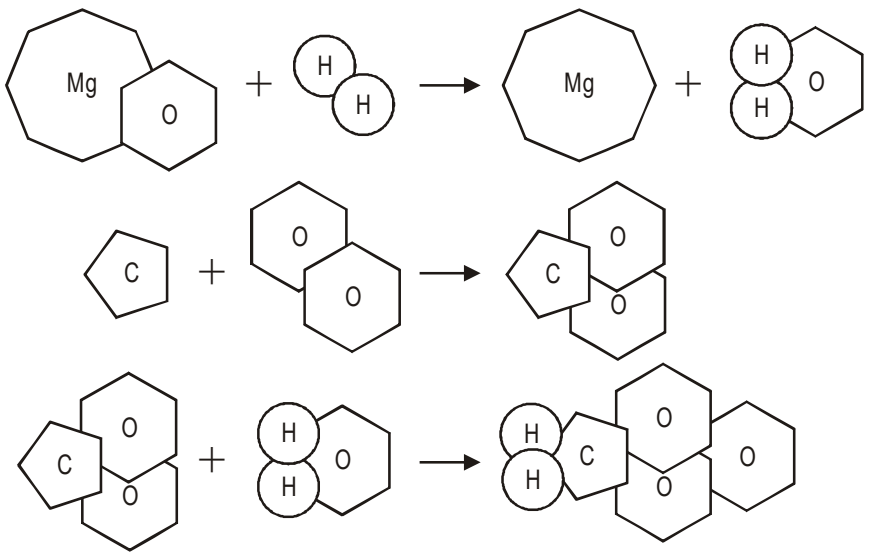
Como padrões:

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

ISLab
Synthetic Intelligence Lab

Exemplo (2)

Reações químicas envolvidas na produção do composto H2CO3.



[mol(mgo), mol(h2)] ==>
[consumir(mol(mgo)), consumir(mol(h2)),
produzir(mol(mg)), produzir(mol(h2o))].

[mol(c), mol(o2)] ==>
[consumir(mol(c)), consumir(mol(o2)),
produzir(mol(co2))].

[mol(co2), mol(h2o)] ==>
[consumir(mol(co2)), consumir(mol(h2o)),
produzir(mol(h2co3))].

Estruturas Hierárquicas

O objetivo é o de fazer uma compactação dos factos a representar num dado sistema;

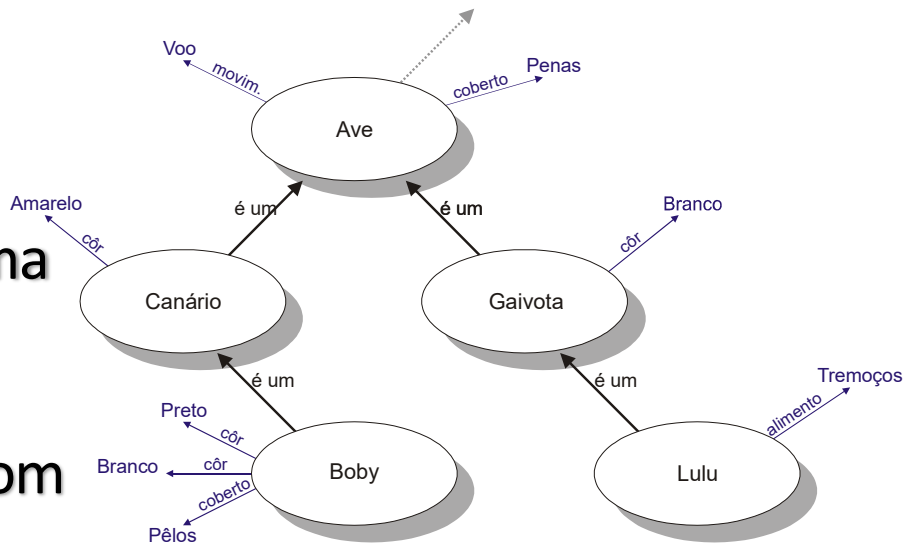
As entidades podem-se agrupar em classes, partilhando valores para os mesmos atributos;

Os factos associados a um dado objeto poderão não estar representados ao seu nível, mas antes serem reconstruídos através de um processo de inferência por herança, sobre as classes superiores.

- **Redes semânticas**
  - correspondem a um grafo, onde os nodos definem as entidades (objetos, classes) do sistema e os ramos definem relações entre as mesmas. Algumas destas relações são chamadas relação é_um (isa) e permitem a herança do conhecimento definido numa dada entidade.
- **Enquadramentos** (Frames)
  - *frames* definem objetos (ou classes), cada um deles com uma designação e um conjunto de *slots*, correspondentes a atributos, onde é colocado um valor. Alguns destes atributos são utilizados para representar as relações entre um objeto e uma classe à qual este pertence e as relações entre classes e superclasses, relações estas que possibilitam a herança de um valor de um dado *slot* não preenchido

Redes semânticas

- Rede de entidades e de relacionamento entre elas;
- Um grafo:
  - cada nodo corresponde a uma entidade;
  - os ramos corresponde às relações e são etiquetados com o nome da relação
- Tipos de Relações: é_um, desloca-se, coberto, etc…



ISLab
Synthetic Intelligence Lab

Interpretador

```
demo( Agente,Questao ) :-
   agente( Agente,Teoria ),
   prova( Questao,Teoria ).
demo( Agente,Questao ) :-
   e_um( Agente,Entidade ),
   demo( Entidade,Questao ).

prova( Questao,[Questao|Teoria] ).
prova( Questao,[X|Teoria] ) :-
   prova( Questao,Teoria ).
```

Implementação de Demo baseado em grafos de agentes;
Cada agente possui um conjunto de propriedades, sendo representado por um nodo;
As relações existentes definem a hierarquia do sistema, sendo que todos os ramos definem relações *é_um*.
Duas formas distintas de responder a uma questão:

- diretamente através do conhecimento representado no agente;
- através de inferência por herança.

ISLab
Synthetic Intelligence Lab

Exemplo

```
agente( ave,
        [ coberto( penas ),
          movimento( voo ) ] ).
agente( canario,
        [ cor( amarelo ),
          som( chilro ) ] ).
agente( papagaio,
        [ comida( pao ),
          som( fala ),
          cor( verde ),
          cor( vermelho )] ).

agente( boby,
        [ cor( preto ),
          cor( branco ) ] ).
agente( lulu,
        [ alimento( tremocos
),
          coberto( pelos ) ] ).

e_um( canario,ave ).
e_um( papagaio,ave ).
e_um( boby,canario ).
e_um( lulu,papagaio ).
```

Diretamente através do conhecimento representado
no agente;

```
| ?- demo(papagaio, comida(X)).
X = pao ?
```

Através de inferência por herança.

```
| ?- demo(papagaio, coberto(X)).
X = penas ?
```

ISLab
Synthetic Intelligence Lab

Frames

- Técnica de representação do conhecimento que tenta organizar conceitos de uma forma que explora inter-relações e crenças comuns;
- Análogo à programação orientada a objetos;
- Estrutura de dados cujos componentes se chamam slots;
- Os Slots são identificados por nomes e denotam informação de vários tipos: valores simples, referências de outras frames, procedimentos, …
- Is_a: relação de entre Classe e Superclasse;
- Instance_of: relação de membro de uma classe;
- Representação: frame(Frame, Slot, Valor).

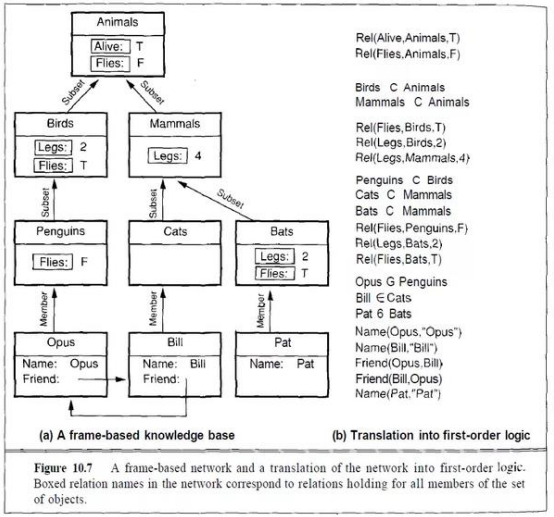


Figure 10.7 A frame-based network and a translation of the network into first-order logic. Boxed relation names in the network correspond to relations holding for all members of the set of objects.

Scripts

- Um script é uma estrutura de dados usada para representar uma sequência de eventos
- Os scripts são usados para interpretar histórias.
- Exemplos populares foram sistemas orientados a scripts que podem interpretar e extrair factos de revistas.

- Uma script é caracteriza-se:
  1) Uma cena
  2) Adereços (objetos manipulados no script)
  3) Os atores (agentes que podem mudar o estado do mundo).
  4) Eventos
  5) Atos: conjunto de ações dos atores.

Em cada cena, um ou mais atores realizam ações. Os atores agem com os adereços. O script pode ser representado como uma árvore ou rede de estados, impulsionada por eventos.

Assim como os *Frames*, os scripts orientam a interpretação, dizendo ao sistema o que procurar e onde procurar. O script pode prever eventos.

ISLab
Synthetic Intelligence Lab

Exemplo

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.

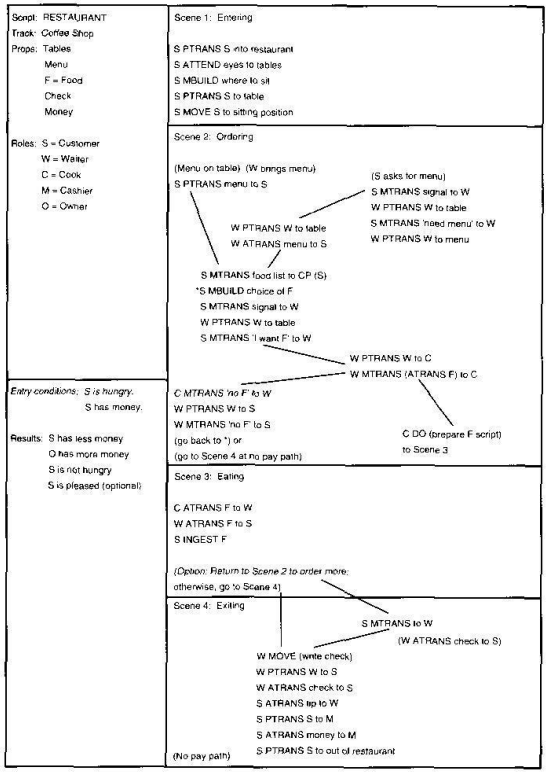
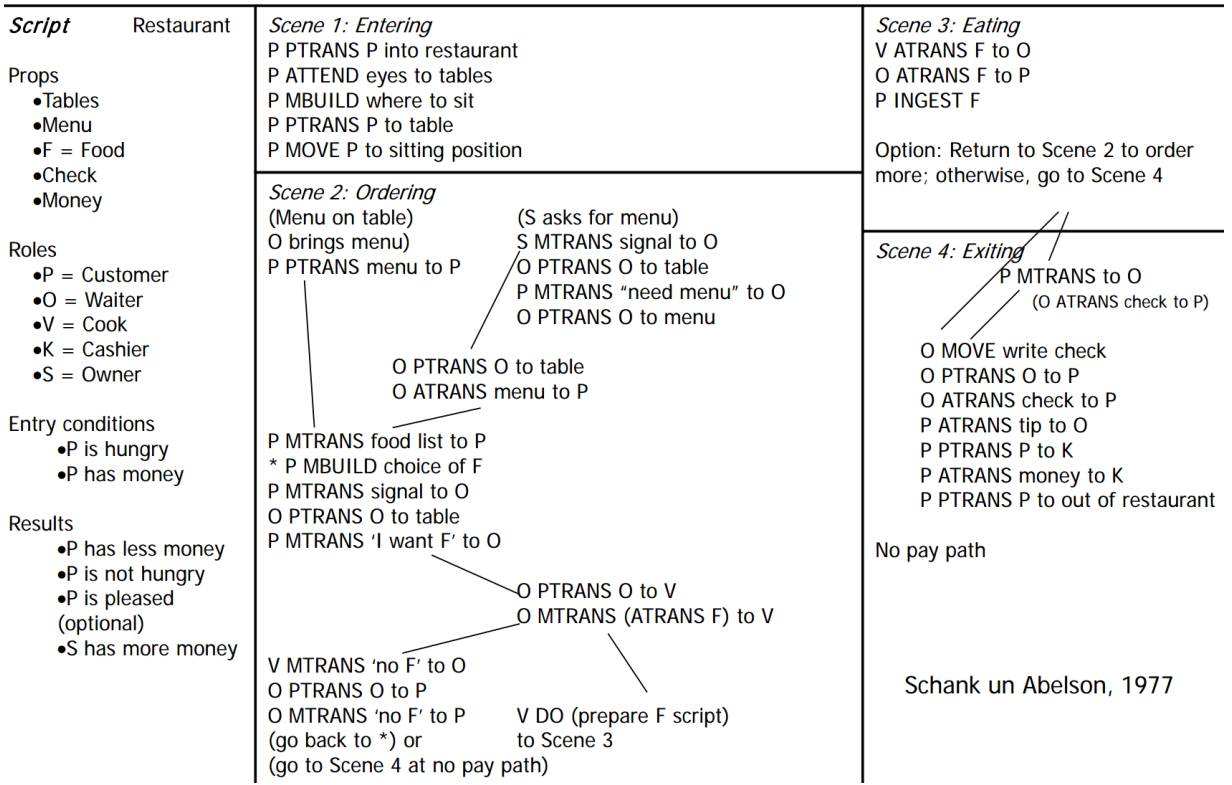


Figure 9.26 A restaurant script (Schank 1977).

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

ISLab
Synthetic Intelligence Lab

Exemplo Continuação



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

- Sistemas Baseados em Conhecimento
  - Programas de computador que utilizam o **conhecimento representado explicitamente** para resolver problemas;
  - Manipulam conhecimento e informação de forma inteligente;
  - São desenvolvidos para resolverem problemas que requerem grandes porções de conhecimento humano e especialização (perícia).

- CONHECIMENTO + RACIOCÍNIO = RESOLUÇÃO PROBLEMA

Definição

- Perspectiva do Conhecimento processável pelo homem
  - A análise e modelação do método de resolução do problema.
- Perspectiva Simbólica processável pelo computador
  - A atividade de representar este método através de um formalismo computacionalmente eficiente.
- Capacidade de Raciocínio/Inferência
  - É a capacidade de definir um conjunto de passos para a resolução eficiente e rápida de um problema;
  - O próprio mecanismo de inferência é conhecimento.

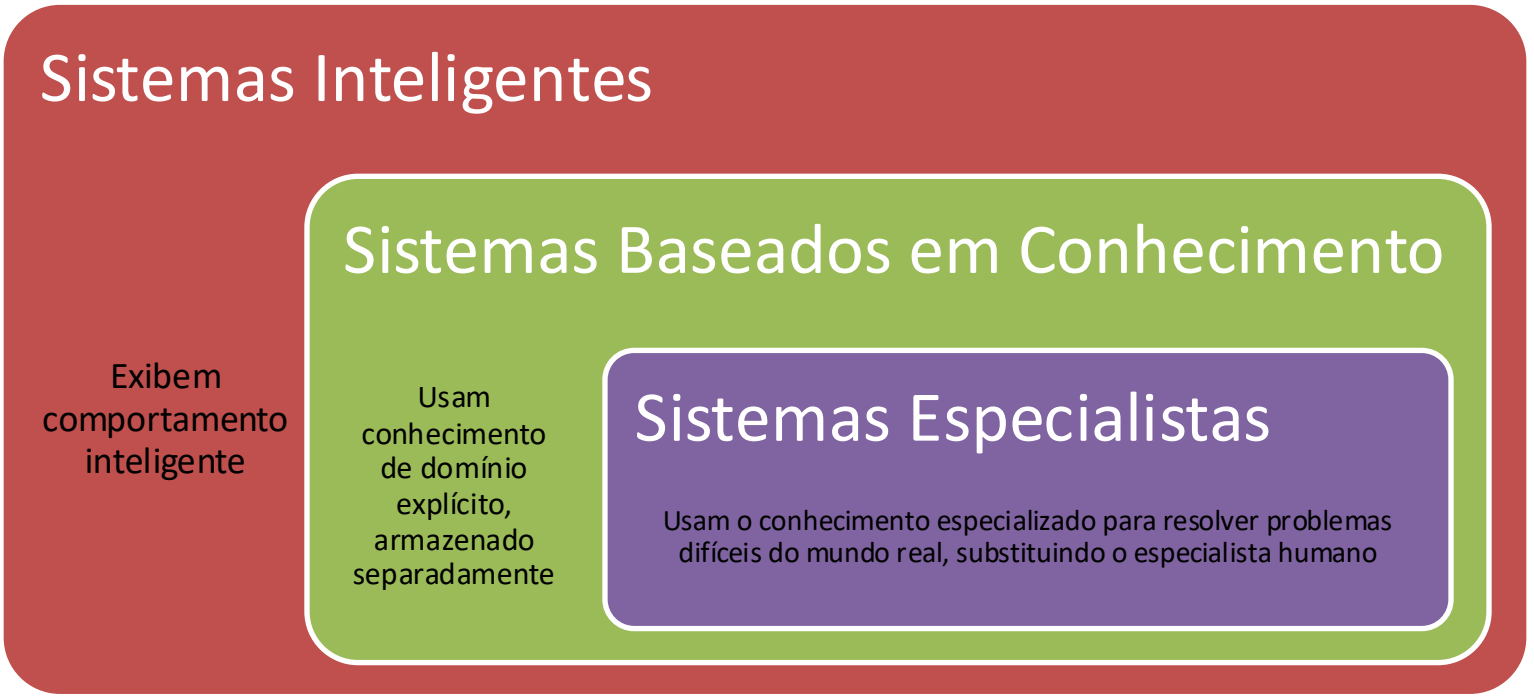
ISLab
Synthetic Intelligence Lab

Diferenças para outros Sistemas

Sistemas Convencionais	Sistemas Baseados em Conhecimento
Estrutura de Dados	Representação de Conhecimento
Dados e Relações entre os Dados	Conceitos, Relações entre Conceitos e Regras
Usam Algoritmos Determinísticos	Pesquisa com Heurística
Conhecimento embebido no código do programa	Conhecimento representado explicitamente e separado do programa que o manipula e interpreta
Explicação do raciocínio é difícil	Podem e devem explicar o seu raciocínio

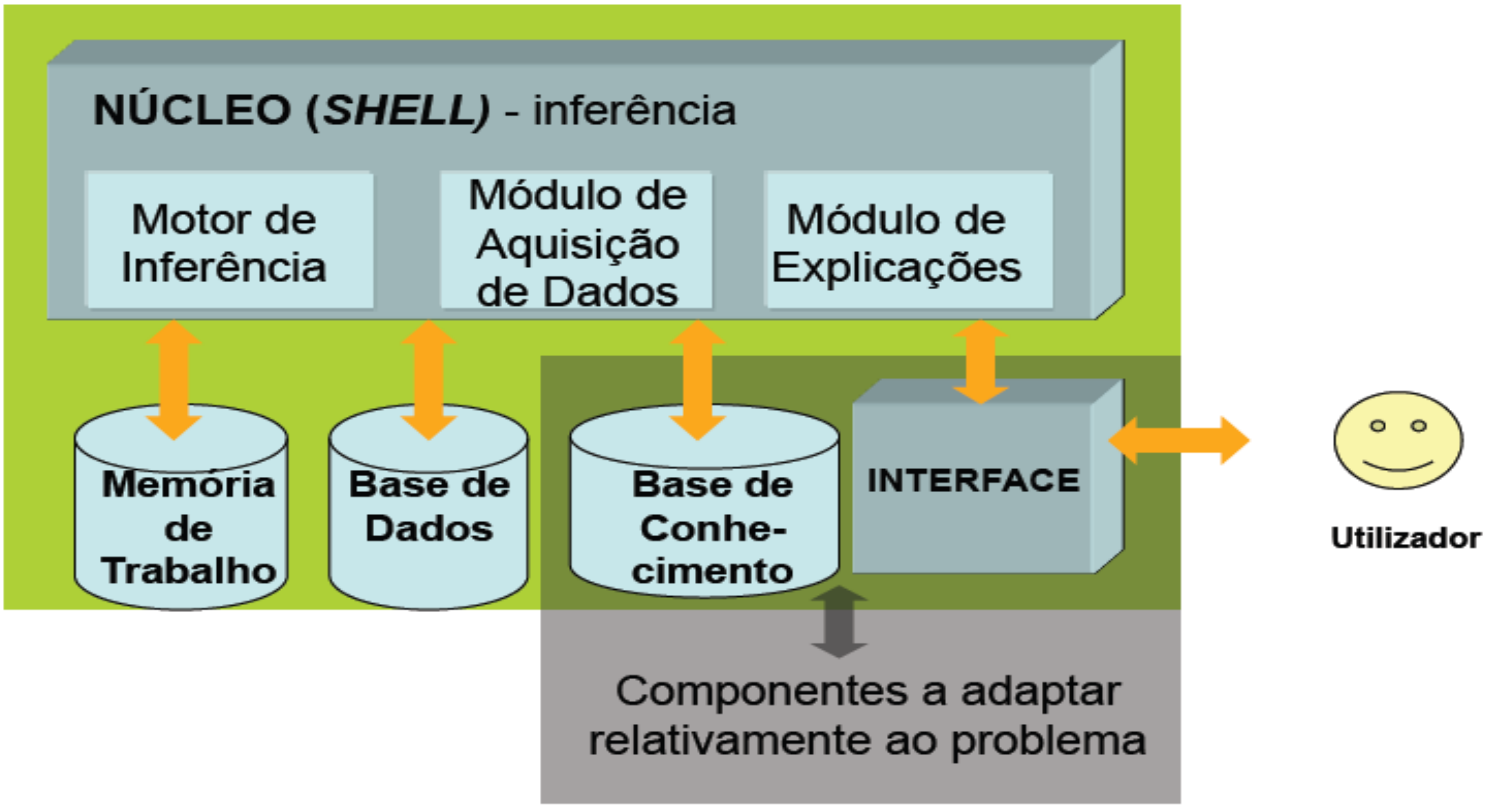
ISLab
Synthetic Intelligence Lab

Sistemas Baseados em Conhecimento



ISLab
Synthetic Intelligence Lab

Estrutura



Fonte: Luís Paulo Reis, Artificial Intelligence (2019), Universidade do Porto

Estrutura

- **Núcleo (Shell)**
  - Controlo da interação com o utilizador;
  - Inferência do conhecimento;
  - Explicação das conclusões.
- **MOTOR/SISTEMA DE INFERÊNCIA**
  - Desenvolvimento do raciocínio baseado nas informações obtidas pelo Motor de Aquisição de Dados e no conhecimento representado na Base de Conhecimento.
  - Por exemplo, para regras de produção:
    - Encadeamento para a frente (*forward chaining*):
    - Encadeamento para trás (*backward chaining*).

Estrutura

- **MÓDULO DE AQUISIÇÃO DE DADOS**
  - Interação com o utilizador;
  - Obtenção de informações sobre o problema (perguntas ao utilizador);
  - Verificação da validade das respostas.
- **MÓDULO DE EXPLICAÇÃO**

  Justificação das conclusões obtidas:
  - Porquê – porque é que o MAD fez a pergunta ao utilizador;
  - Como – caminho de raciocínio para chegar às conclusões;
  - Estudo de cenários - O que acontece se alguma informação fornecida pelo utilizador for alterada (*what if);*
  - Porque não – explicar porque uma determinada conclusão não foi obtida.

Base Conhecimento

- **BASE DE CONHECIMENTO**
  - Descrição do CONHECIMENTO necessário para a resolução do problema;
  - Conjunto de representações de ações e acontecimentos do mundo;
  - SENTENÇAS expressas numa determinada LINGUAGEM DE REPRESENTAÇÃO DE CONHECIMENTO:
    - Regras de Produção;
    - Redes Semânticas;
    - *Frames (Enquadramentos);*
    - Orientado ao objecto;
    - Lógica;
    - Baseado em Casos (*Case Based Learning);*
    - Híbridas, …

ISLab
Synthetic Intelligence Lab

Base Conhecimento

Exemplo de uma frase do tipo CAUSA-EFEITO (regra de produção)

- SE TEMP-PACIENTE > 37.5 º C ENTÃO PACIENTE-TEM-FEBRE

Exemplo de META-CONHECIMENTO – conduz a procura da solução:

- SE O DOENTE É ALCOÓLICO ENTÃO PROCURAR PRIMEIRO DOENÇAS HEPÁTICAS
- PROCURAR A SOLUÇÃO PRIMEIRO EM CAMINHOS ONDE EXISTEM POUCAS POSSIBILIDADES (heurística)

Atender a problemas de:

- CONFLITOS / INCONSISTÊNCIA;
- INCOMPLETUDE / INCERTEZA.

Estrutura

- **BASE DE DADOS**
  - Destina-se a conter os dados/informações que caracterizam o problema (fatos).

- **MEMÓRIA DE TRABALHO**
  - Permite armazenar e fornecer a linha de raciocínio;
  - Armazena respostas do utilizador (evita perguntas repetidas);
  - Armazenamento de conclusões intermédias (evita repetição de inferências).

INTERFACE

- Interação entre o SBC e o utilizador;
- Linguagem difere da utilizada para representar o conhecimento:
  - Linguagem natural;
  - Linguagens Visuais;
  - Linguagens Diagramáticas;
  - Multimédia.
- Princípios oriundos das teorias cognitivas e semióticas (*Human-Computer Interaction*):
  - Eficiência;
  - Dinamismo;
  - Desenvolvimento em tempo útil.

FERRAMENTAS DE SUPORTE À CONSTRUÇÃO DE UM SBC:

- Utilização de linguagens de programação como LISP e PROLOG
- Ferramentas de Apoio – diversos esquemas de representação de conhecimento, motores de inferência, interfaces, etc.
  - ART, Babylon, KEE, Knowledge Craft, Loops, Flex, Elements Environment, ...
- Shells - a interface e estratégia de resolução de problemas é prédefinida:
  - Insight, KES, MED2, M.1, Personal Consultant, S.1, Timm
  - EXSYS CORVID, CLIPS, JESS (Clips em Java), Jlisa (Clips for Java),

Fonte: Luís Paulo Reis, Artificial Intelligence (2019), Universidade do Porto

ISLab
Synthetic Intelligence Lab

Bibliografia Recomendada

- 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