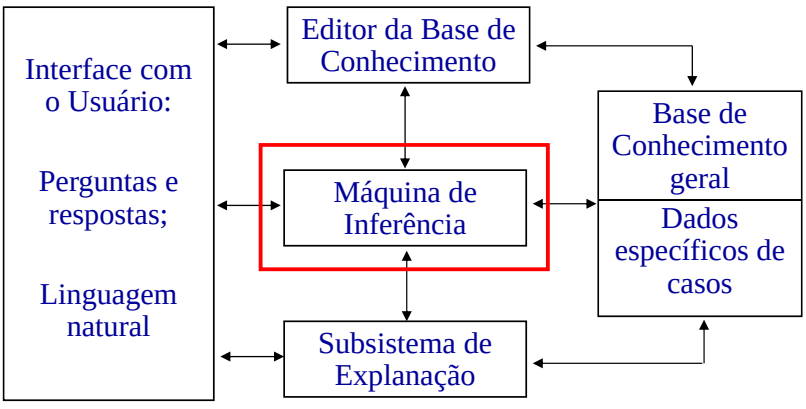
Inteligência Artificial

Sistemas Especialistas

Raciocínio e Mecanismos de Inferência

Prof. Paulo Martins Engel

Arquitetura de um SE típico



Motor de Inferência

- É um programa que utiliza a base de conhecimento como 'dado' na solução de um problema.
- É um programa que utiliza mecanismos gerais de combinação de fatos e regras.
- Funcionamento cíclico- um ciclo de base é composto por duas etapas:
  - Avaliação - procura das regras possíveis de serem ativadas, em função do estado corrente da base de fatos e escolha das regras a ativar efetivamente.
  - Execução - modificação da base de fatos e eventualmente da base de regras.
- Condição de parada:
  - Encontrada a solução para o problema
  - Não há mais regra a ser ativada

Fase de avaliação

- Compreende três etapas:
  - Seleção ou restrição
  - Filtragem
  - Resolução de conflitos
- Seleção ou restrição
  - É a primeira etapa
  - Determina um subconjunto da base de regras e da base de fatos que, a priori, merece ser submetido à etapa seguinte de filtragem.
- Exemplo:
  - Separar fatos e regras em função do domínio
  - Separar fatos reconhecidamente verdadeiros (estabelecidos) dos fatos a estabelecer (objetivos, hipóteses).

FILTRAGEM

- Compara o antecedente de cada uma das regras selecionadas na etapa anterior com o conjunto dos fatos considerados nesta etapa.
- O subconjunto destas regras que tem condições de ser ativadas é chamado conjunto de conflito.

RESOLUÇÃO DE CONFLITOS

- Nesta etapa, determina-se o subconjunto de regras que serão efetivamente ativadas:
  - Pela ordem das regras na BC (ex. Prolog)
  - Regras mais específicas antes de regras mais genéricas

FASE DE EXECUÇÃO

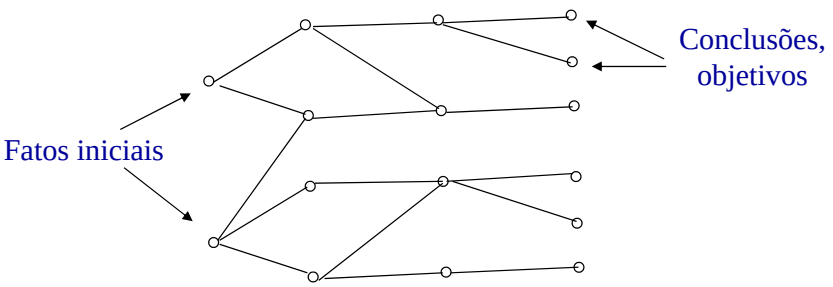
- É a segunda parte de cada ciclo.
- O mecanismo de inferência comanda a ativação das regras selecionadas na fase de avaliação.

Mecanismos de Inferência

1 - ENCADEAMENTO DE REGRAS

a) ENCADEAMENTO PROGRESSIVO (FORWARD CHAINING)

- Raciocínio para frente (forward reasoning)
- Raciocínio orientado pelos dados (data oriented reasoning)
- Os fatos da base de fatos sobre os quais é considerado o antecedente (condição) das regras representam informações cujo valor verdadeiro já foi estabelecido.



Exemplo

Base de Regras:

R_1: C → M
R_2: A, D → E
R_3: L → H
R_4: B, C → G
R_5: A, B → C
R_6: G, D → H, I
R_7: C → D
R_8: E, K → H

Base de Fatos: A, B

Objetivo: H

Regra aplicada:	Base de fatos:

Exemplo

Base de Regras:

R_1: C → M
R_2: A, D → E
R_3: L → H
R_4: B, C → G
R_5: A, B → C
R_6: G, D → H, I
R_7: C → D
R_8: E, K → H

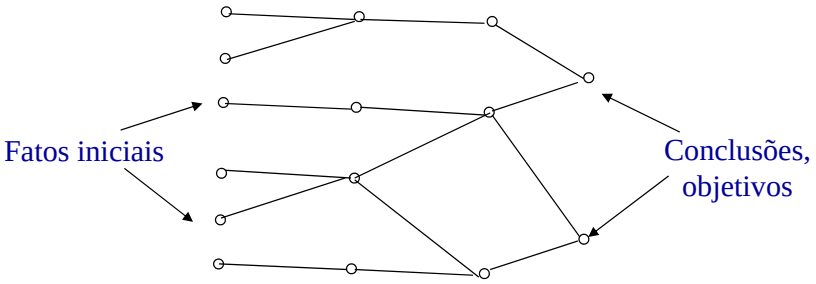
Base de Fatos: A, B

Objetivo: H

Regra aplicada:	Base de fatos:
R_5	A, B, C
R_1	A, B, C, M
R_4	A, B, C, M, G
R_7	A, B, C, M, G, D
R_2	A, B, C, M, G, D, E
R_6	A, B, C, M, G, D, E, H, I

b) ENCADEAMENTO REGRESSIVO (BACKWARD CHAINING)

- Raciocínio para trás (backward reasoning)
- Raciocínio orientado pelos objetivos (goal oriented reasoning)
- Parte-se dos objetivos, das metas e anda-se "para trás" através de subobjetivos.



Exemplo (encadeamento regressivo)

Base de Regras:

R_1: C $\rightarrow$ M
R_2: A, D $\rightarrow$ E
R_3: L $\rightarrow$ H
R_4: B, C $\rightarrow$ G
R_5: A, B $\rightarrow$ C
R_6: G, D $\rightarrow$ H, I
R_7: C $\rightarrow$ D
R_8: E, K $\rightarrow$ H

Base de Fatos: A, B

Objetivo: H

Regra	Metas	Base de fatos:
	H	A, B

Exemplo (encadeamento regressivo)

Base de Regras:

R_1: C → M
R_2: A, D → E
R_3: L → H
R_4: B, C → G
R_5: A, B → C
R_6: G, D → H, I
R_7: C → D
R_8: E, K → H

Base de Fatos: A, B

Objetivo: H

Regra	Metas	Base de fatos:
	H	A, B
R_3	L	A, B, H

Exemplo (encadeamento regressivo)

Base de Regras:

R_1: C → M
R_2: A, D → E
R_3: L → H
R_4: B, C → G
R_5: A, B → C
R_6: G, D → H, I
R_7: C → D
R_8: E, K → H

Base de Fatos: A, B

Objetivo: H

Regra	Metas	Base de fatos:
	H	A, B
R_3	L	A, B, H
	H	A, B

Impasse

Exemplo (encadeamento regressivo)

Base de Regras:

R_1: C → M
R_2: A, D → E
R_3: L → H
R_4: B, C → G
R_5: A, B → C
R_6: G, D → H, I
R_7: C → D
R_8: E, K → H

Base de Fatos: A, B

Objetivo: H

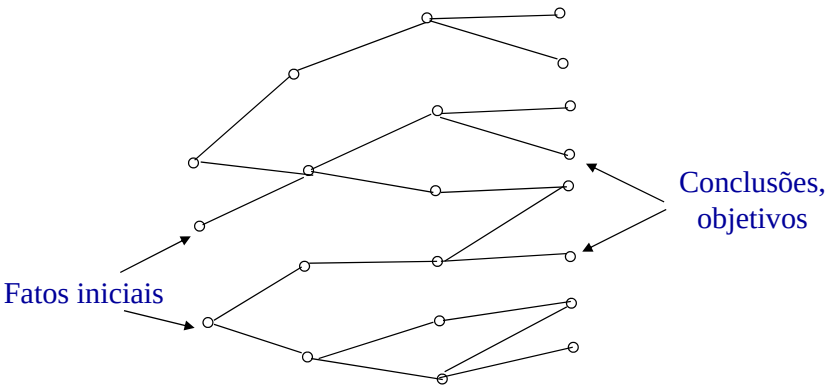
Impasse

Regra	Metas	Base de fatos:
	H	A, B
R_3	L	A, B, H
	H	A, B
R_6	G, D	A, B, H, I
R_4	D, C	A, B, H, I, G
R_7	C	A, B, H, I, G, D
R_5	—	A, B, H, I, G, D, C

Mecanismos de Inferência

2 - TIPOS DE BUSCA

Em profundidade X em largura ou amplitude



Algoritmo por Encadeamento progressivo 1: busca em profundidade

- O algoritmo acrescenta as conclusões das regras disparadas imediatamente na base de fatos.
- Isto faz com que a busca seja em profundidade.
- Com isso, a base de regras não é totalmente explorada, fazendo com que o desempenho do algoritmo (tempo de busca do objetivo) dependa da ordem das regras dentro da base de regras.

Encadeamento progressivo 1 (em profundidade)

```
procedure ESTABELECER-UM-FATO (OBJETIVO)
    if OBJETIVO pertence à BASE-DE-FATOS
            then exit "sucesso"
    exit EXECUTAR-UM-CICLO  (BASE-DE-REGRAS)

procedure EXECUTAR-UM-CICLO (AS-REGRAS)
    if AS-REGRAS é vazio
            then exit "insucesso"
    UMA-REGRA ← escolha de uma regra de AS-REGRAS
    AS-REGRAS ← AS-REGRAS diminuída de UMA-REGRA
    if todos fatos da premissa de UMA-REGRA pertencem a BASE-DE-FATOS
            then  begin
                  if a conclusão de UMA-REGRA é OBJETIVO
                          then exit "sucesso"
                  acrescentar a conclusão de UMA-REGRA à BASE-DE-FATOS
                   BASE-DE-REGRAS ← BASE-DE-REGRAS diminuída de UMA-REGRA
                  exit EXECUTAR-UM-CICLO (BASE-DE-REGRAS)
                  end
exit EXECUTAR-UM-CICLO (AS-REGRAS)
```

Encadeamento progressivo 2: busca em largura

- Ao contrário do esquema anterior, neste caso, ativa-se uma após a outra, todas as regras compatíveis com o estado da "base de fatos", antes de acrescentar suas conclusões (fatos novos) à base de fatos e as utilizar para ativar novas regras.
- Com isso, o algoritmo explora *em largura* toda a base de regra inicial.
- Neste caso, a ordem das regras dentro da base de regras tem pouca influência no desempenho (tempo de busca do objetivo) do algoritmo.

Encadeamento progressivo 2 (em largura)

```
procedure ESTABELECER-UM-FATO (OBJETIVO)
    if OBJETIVO pertence à BASE-DE-FATOS
    then exit "sucesso"
    FATOS-NOVOS ← lista vazia
exit EXECUTAR-UM-CICLO  (BASE-DE-REGRAS, FATOS-NOVOS )

procedure EXECUTAR-UM-CICLO (AS-REGRAS, FATOS-NOVOS )
    if AS-REGRAS é vazio
     then begin          % no caso de já ter explorado toda a base de regras
            if FATOS-NOVOS é vazio
                then exit "insucesso"
            BASE-DE-FATOS ← BASE-DE-FATOS acrescida de FATOS-NOVOS
            FATOS-NOVOS ← lista vazia
            exit EXECUTAR-UM-CICLO  (BASE-DE-REGRAS, FATOS-NOVOS )
          end
```

Encadeamento progressivo 2 (em largura)

```
    % (else) se AS-REGRAS não é vazio
    UMA-REGRA ← escolha de uma regra de AS-REGRAS
    AS-REGRAS ← AS-REGRAS diminuída de UMA-REGRA
    if todos fatos da premissa de UMA-REGRA pertencem a BASE-DE-FATOS
            then begin
                    if a conclusão de UMA-REGRA é OBJETIVO
                            then exit "sucesso"
                    acrescentar a conclusão de UMA-REGRA à FATOS-NOVOS
                     BASE-DE-REGRAS ← BASE-DE-REGRAS diminuída de UMA-REGRA
                end
exit EXECUTAR-UM-CICLO (AS-REGRAS, FATOS-NOVOS)
```


Encadeamento regressivo

```
procedure ESTABELECER-UM-FATO (OBJETIVO)
    if OBJETIVO pertence à BASE-DE-FATOS
    then exit "sucesso"
exit ESTABELECER1  (BASE-DE-REGRAS)

procedure ESTABELECER1 (AS-REGRAS)
    if AS-REGRAS é vazio
            then exit "insucesso"
    UMA-REGRA ← escolha de uma regra de AS-REGRAS
    AS-REGRAS ← AS-REGRAS diminuída de UMA-REGRA
    if UMA-REGRA tem OBJETIVO em conclusão
            then if  ESTABELECER2 (UMA-REGRA) = "sucesso"
                    then exit "sucesso"
    exit ESTABELECER1 (AS-REGRAS)

procedure ESTABELECER2 (A-REGRA)
    OS-OBJETIVOS ← todos os fatos que compõem a premissa de A-REGRA
    exit ESTABELECER-CONJUNÇÃO-DE-FATOS (OS-OBJETIVOS)
```

Encadeamento regressivo

```
procedure ESTABELECER-CONJUNÇÃO-DE-FATOS (AS-METAS)
    if AS-METAS é vazio
            then exit "sucesso"
    UMA-META ← escolha de um elemento de AS-METAS
    AS-METAS ← AS-METAS diminuído de UMA-META
    if ESTABELECER-UM-FATO (UMA-META) = "insucesso"
            then exit "insucesso"
    exit ESTABELECER-CONJUNÇÃO-DE-FATOS (AS-METAS)
```


Exercício

- A partir da base de regras e da base de fatos abaixo, aplicar os três algoritmos estudados e determine a seqüência de disparo das regras. Considere que Q seja o objetivo buscado.

Base de Regras:

R_1: K, L, M $\rightarrow$ I
R_2: I, L, J $\rightarrow$ Q
R_3: C, D, E $\rightarrow$ B
R_4: A, B $\rightarrow$ Q
R_5: L, N, O, P $\rightarrow$ Q
R_6: C, H $\rightarrow$ R
R_7: R, J, M $\rightarrow$ S
R_8: F, H $\rightarrow$ B
R_9: G $\rightarrow$ F

Base de Fatos

A, C, D, E, G, H, K

Métodos de Solução de Problemas

- Componente dinâmico do conhecimento
- Modelo abstrato da inferência aplicável àquela classe de problemas
- NÃO correspondem aos métodos de inferência por busca, como raciocínio progressivo ou regressivo
- Generalização de um *padrão* de raciocínio específico, mas *não* é um raciocínio genérico

Modelo da Tarefa Diagnóstico

TASK Diagnóstico;

ROLES:

INPUT:

reclamação:“Queixa do cliente”;

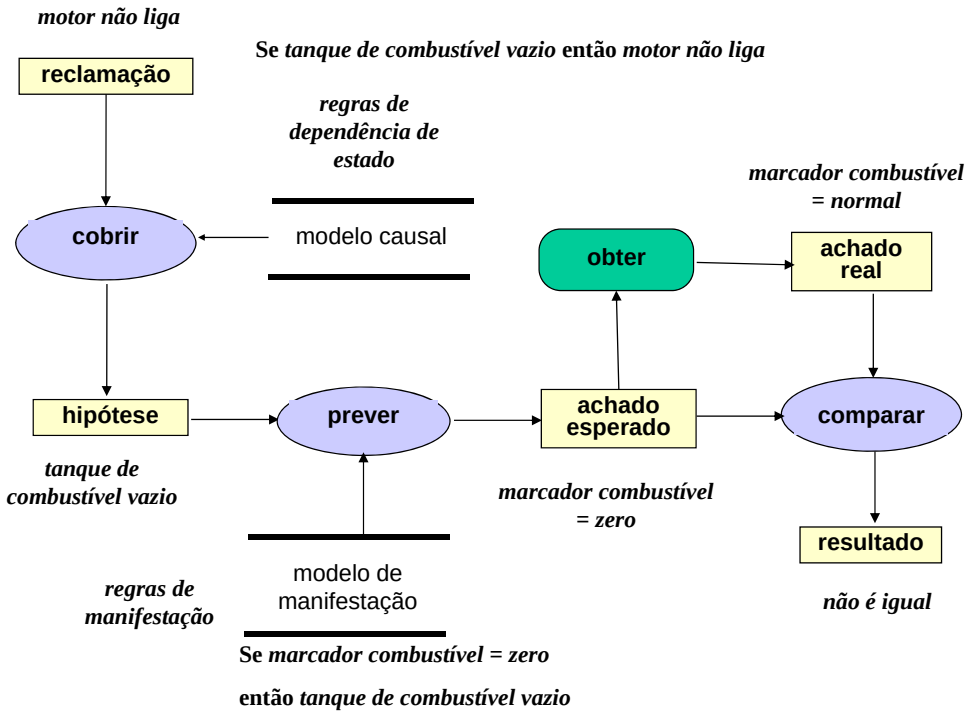
OUTPUT:

Falhas: “As falhas que causaram reclamações”;

Evidência: “As evidências reunidas durante o diagnóstico”;

END TASK Diagnóstico;

Diagnóstico



TASK-METHOD diagnóstico-por-gerar-e-testar;
REALIZES: diagnóstico-de-carro;
DECOMPOSITION:
INFERENCES: cobrir, prever, comparar*;*
TRANSFER-FUNCTIONS: obter;
ROLES:
INTERMEDIATE:
hipótese*: "uma solução candidata";*
achado-esperada: *"O achado previsto, caso a hipótese seja verdadeira";*
achado-real: *"O achado realmente observado";*
resultado: *"O resultado da comparação";*
CONTROL-STRUCTURE:
WHILE NEW-SOLUTION cobrir(reclamação -> hipótese) DO
prever(hipótese -> achado-esperado);
obter(achado-esperado -> achado-real);
evidência := evidência ADD achado-real;
comparar(achado-esperado + achado-real -> resultado);
IF resultado == equal
THEN "interromper o laço";
END IF
END WHILE
IF result == equal
THEN categoria-falha := hipótese;
ELSE "não foi encontrada uma solução";
END IF
END WHILE