Conhecimento e Raciocínio

Aula 7
Aquisição e Representação do Conhecimento

Viriato A.P. Marinho Marques
DEIS - ISEC
2020 / 2021

2.1 Tipos de Conhecimento

2.1 Tipos de Conhecimento

Conhecimento

• **A priori**: Os factos e regras conhecidos acerca de um domínio antes de ser realizada uma sessão como SP (p.e. factos regras na *knowledge base*)

• **Inferido**: Os factos e regras gerados pelo SP durante uma sessão (nota: só são criadas novas regras para SP com capacidade de aprendizagem)

Conhecimento

• **Profundo (*deep knowledge*)**: Obtido através de aprendizagem formal (estudo, treino). Inclui modelos, relações causais, senso comum, etc.

(Exemplo: compreender como funciona um frigorífico)

• **Superficial (*shallow knowledge*)**: Obtido à custa da experiência; essencialmente regras práticas, heurísticas; ligações causa-efeito ou entradas-saídas de um sistema.

(Exemplo: Se não há electricidade, um frigorífico não produz frio)

2.1 Tipos de Conhecimento

Exemplo

Na análise de um processo de empréstimo bancário, X analisa uma proposta do cliente Y. Dados como o seu nome, residência, quanto ganha, etc. serão colocados na **memória de trabalho** de Y e posteriormente esquecidos.

Conhecimento profundo: é neste caso representado pelas políticas do banco, i.e. princípios conhecidos de todos os funcionários: *"se a prestação for acima de x% do vencimento, negar empréstimo"*.

Mas neste exemplo, o caso de Y é marginal, i.e., encontra-se no limiar aprovar / negar empréstimo. É em situações deste tipo que um perito tem a maior importância. O caso requer toda a atenção de X.

Analisando a morada, descobre que Y vive numa zona em declínio e trabalha na empresa Z conhecida por proporcionar empregos pouco estáveis.

Conhecimento superficial: Porém, em vez de negar o empréstimo imediatamente, X aprendeu, pela sua experiência, que em casos destes é aconselhável entrevistar o candidato. Descobre então, na entrevista, que Y é casado com a filha do presidente da empresa Z, uma pessoa de vida estável e de óptimas referências locais. E o empréstimo é concedido.

2.1 Tipos de Conhecimento

Conhecimento

• **Procedimental**: Como o nome indica, descreve os procedimentos a observar para realizar uma inferência. Descreve como proceder
• **Declarativo**: Declara factos, definições, princípios. O processo de raciocínio (como proceder) não é explicitamente visível
• **Meta-Conhecimento**: Definido como conhecimento sobre o próprio conhecimento. Funciona como um "supervisor" das inferências em curso

Exemplos

```
// Procedimental (C) (Jackson,P.)
char c;
c=fly("pinguim");
...
// Function: Se é pinguim não voa
char fly (char *s)
{
   char answer = "y";
   if (strcmp (s, "penguin")==0)
      {answer='n';
   return answer;
}
```

```
; Declarativo (CLIPS) (Jackson,P.)
; Se for pinguim não voa
(defrule
   (bird (type ?X))
   => (assert (yes)))
(defrule
   (bird (type pinguim))
   => (assert (no)))

* Meta-Conhecimento (Diag. CBR)
* Se sintoma x ausente de síndroma y
* Gerar contribuição negativa
*       para diag. d em consideração
```

2.1 Tipos de Conhecimento

Já (Turban & Aronson) definem:

- **Regras Declarativas** (*knowledge rules*): Declaram os factos e relações relativas a um dado problema
- **Regras de Inferência ou Meta-Regras** (*procedural rules, inference rules, meta-rules*): Guiam a resolução de um problema sendo conhecidos previamente factos e relações. São regras a respeito de regras.

*** Regras Declarativas**

```
SE
ocorrer um conflito internacional
ENTÃO
o ouro desce

SE
o conflito for no Médio Oriente
ENTÃO
comprar ouro
```

*** Meta-Regras**

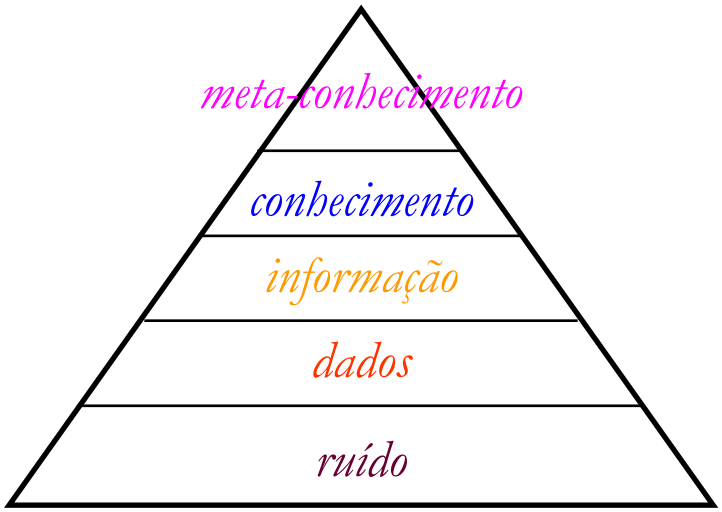
```
SE
os dados necessários não estão
disponíveis
ENTÃO
pedi-los ao utilizador

SE
mais que uma regra está activada
ENTÃO
desactivar as regras que não geram
factos novos
```

2.1 Tipos de Conhecimento

• **Documentado**: O que provém de manuais, livros, filmes, bases de dados, diagramas, sensores, comportamento observado, canções, etc.
• **Não Documentado:** O que reside no cérebro das pessoas, em particular no dos peritos

Hierarquia do Conhecimento



Segundo Wirth's

Algoritmos + Estruturas de Dados = Programas

Paralelismo com Wirth's

Conhecimento + Inferência = Sistema Pericial

2.2 Representação do Conhecimento

O modo exacto como o cérebro humano representa o conhecimento ou como funciona não é fundamental nem objecto de estudo dos SP. Para os SP importa:

- Utilizar um formato compatível com um sistema informático
- Garantir uma correspondência entre este formato e os factos e regras que os peritos efectivamente utilizam
- Ter uma representação que permita acesso e manutenção fáceis

Vamos considerar os seguintes métodos de representação do conhecimento:

- Ternos OAV
- Redes Semânticas
- *Frames* (enquadramentos)
- Proposições Lógicas
- Redes Neuronais (estudo já iniciado em SI)
- Regras (a analisar em secção própria, mais adiante)
- *Scripts* (a analisar na secção de CBR)
- Árvores de Decisão (a analisar na secção de CBR)

2.2 Representação do Conhecimento

2.2.1 Ternos OAV

O termo Terno OAV identifica um conjunto de 3 elementos: Objecto, Atributo, Valor *(Object-Attribute-Value)*

Além de serem um método de representação em si, são também um elemento constitutivo de virtualmente qualquer outro método de representação.

Os objectos são entidades abstractas ou do mundo real. Os atributos são características dos objectos. Os valores são as quantidades ou descrições associadas a cada atributo num dado domínio ou situação.

Exemplos

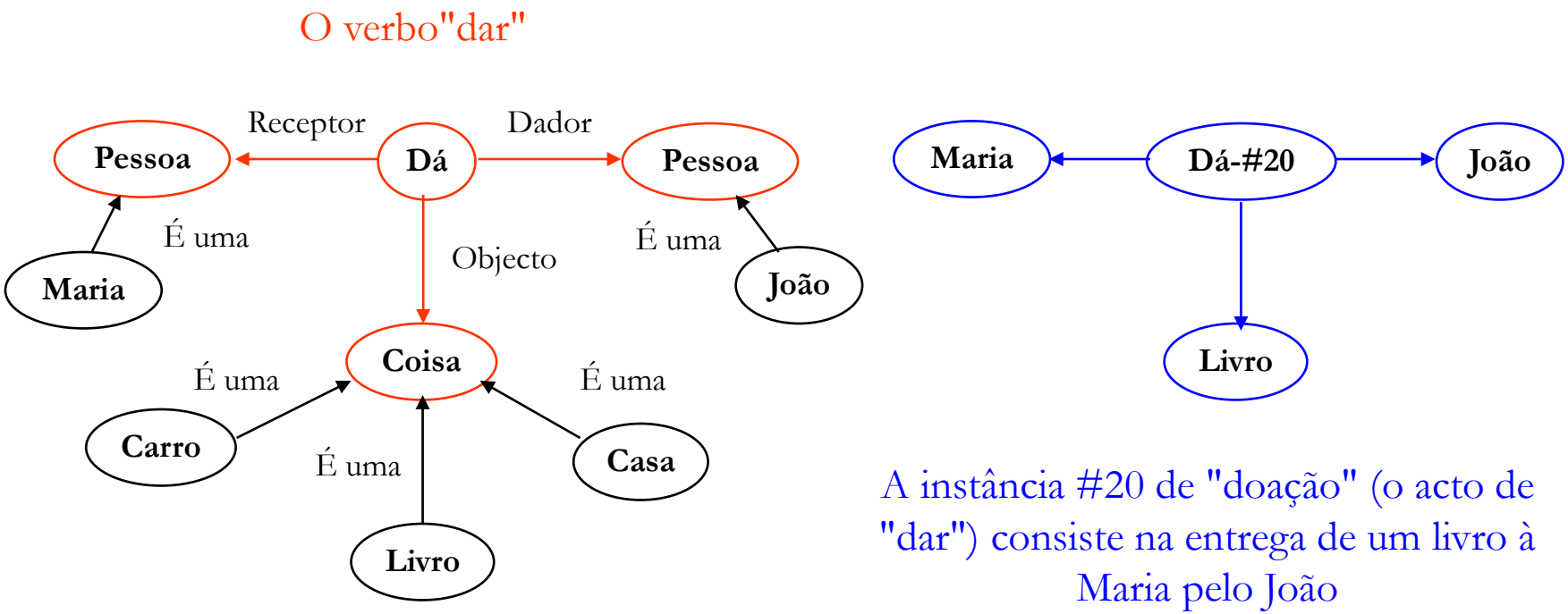
Objecto **Boeing 747**. Alguns atributos: Número de Motores, Tipo de Motor e Tipo de Asas. Valores: Número de Motores = 4; Tipo de Motor = Turbina; Tipo de Asas = Convencional

Objecto **Acção Judicial**. Alguns atributos: Queixoso, Réu, Tipo de Acção. Valores: Queixoso="João dos Azares"; Réu="Manuel das Carteiras"; Tipo de Acção="Roubo".

2.2 Representação do Conhecimento

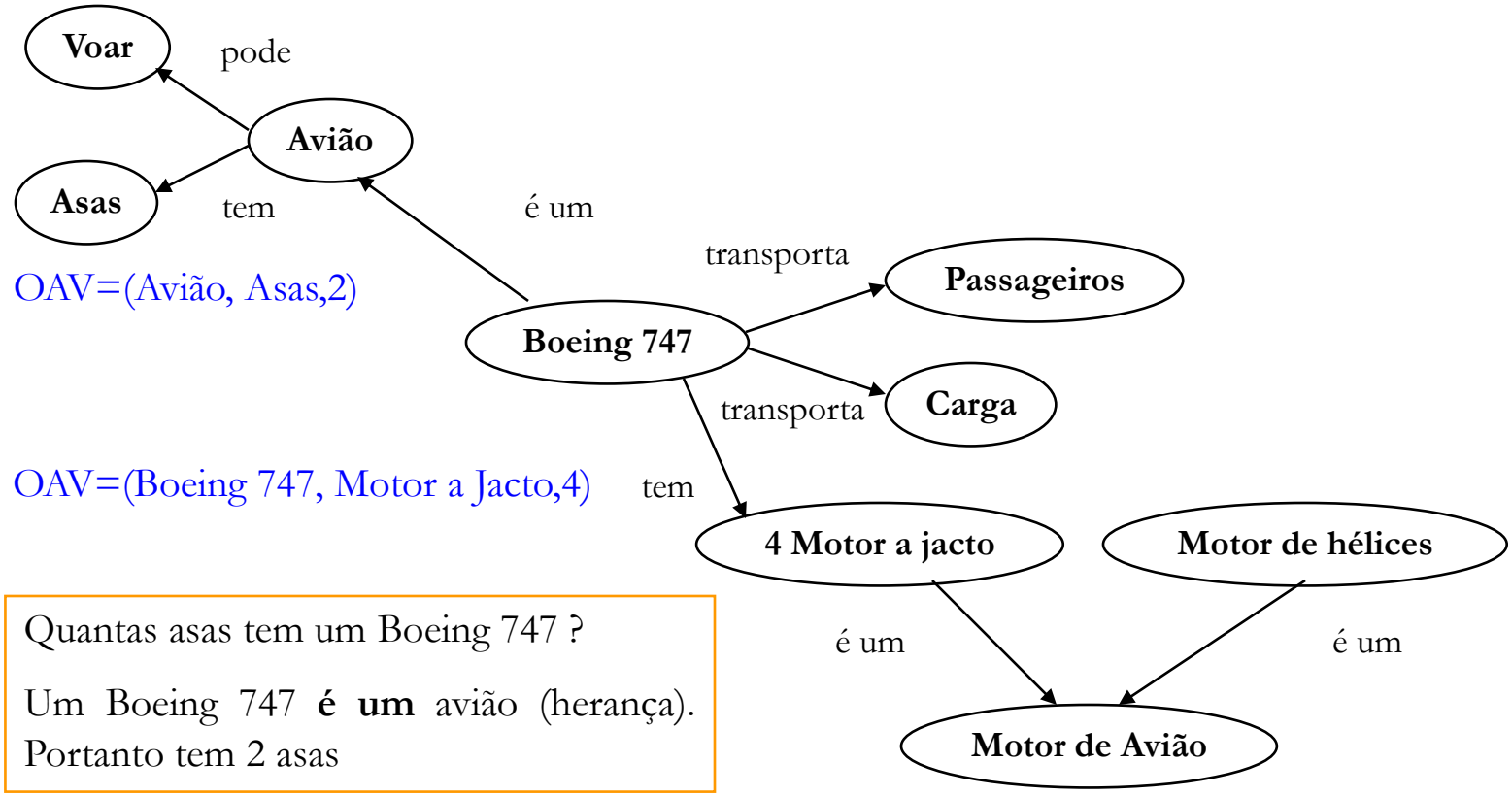
2.2.2 Redes Semânticas

Uma Rede Semântica é um grafo em que os nós são objectos e os arcos relações. São usadas na interpretação (compreensão) de Linguagem Natural e daí o seu nome (semântica quer dizer "o significado de" contrariamente a "morfologia" que significa " a forma de").



2.2 Representação do Conhecimento

Do ponto de vista dos SP as Redes Semânticas interessantes podem ser vistas como conjuntos de triplos OAV referentes a vários objectos interligados entre si por relações.



OAV=(Avião, Asas,2)

OAV=(Boeing 747, Motor a Jacto,4)

Quantas asas tem um Boeing 747 ?

Um Boeing 747 **é um** avião (herança). Portanto tem 2 asas

2.2 Representação do Conhecimento

2.2.3 Enquadramentos (*frames*)

Um *frame* é uma estrutura que descreve completamente uma dada entidade. Foram criados por (Minsky, 1975) antes de existir o conceito de OOP - *Object Oriented Programming*. São semelhantes a objectos mas, sob certos aspectos, "mais poderosos".

Num *frames* o conhecimento está organizado em *slots* que contêm, cada um, 1 ou mais *facets* (também chamados *subslots*):

- ***Slots***: contêm conhecimento declarativo (valores de atributos: "a cor de um carro") ou meta-regras (activar a regra *n* se o valor do atributo *x* exceder o limite *l*)
  - ***Facets***: Podem tomar muitas formas mas existem *facets* típicos, tais como:
    - *Values*: valor de um atributo do *slot* como p.e. "temperatura=80"
    - *Default*: valor por defeito de um dado atributo, p.e. "nº de rodas de um automóvel = 4"
    - *Range*: domínio de um atributo (p.e. [0,100]), tipo (p.e. numérico), formatação, etc.

2.2 Representação do Conhecimento

- *If added*: informação procedimental a executar quando num *slot* for adicionado ou modificado o valor de um atributo. Estes procedimentos são chamados *demons*
- *If needed*: informação procedimental a executar quando for necessário um dado valor e este não for conhecido (processo semelhante ao usado no *If added*)
- *Other*: Podem conter outros *frames*, regras, redes semânticas ou qualquer tipo de informação

Exemplos

1. SP de diagnóstico de falhas com funções de supervisão sobre um robot de soldadura (p.e.) dispondo de sensores que lhe fornecem informações várias tais como pressão de óleo. Se a certa altura o SP precisar de saber a pressão do óleo, o *facet If needed* "disparará" desencadeando um procedimento de aquisição de dados.

2. SP de prescrição de dosagens de medicamentos: se o SP souber qual a dosagem para um paciente de 80Kg, o *facet If needed* disparará se o SP pretender calcular a dosagem para um paciente de 20Kg. Neste caso o *demon* será composto pela fórmula de adaptação da dosagem ao peso da pessoa.

2.2 Representação do Conhecimento

Tal como entre classes e objectos, entre *frames* podem ser estabelecidas relações de hierarquia (*Is a*), dando origem a *parent* e *child frames*. Um *child frame* herda propriedades do seu *parent frame*.

SLOTS	**FACETS**		
Nome	Toyota Corolla		
Dono	Ver Registo		
NºCilindros	***Range***	4 a 6	
	If Needed	Perguntar ao dono	
Modelo	Sedan Sport		
	Range	2 a 4 portas	
	If Needed	Perguntar ao dono	

SLOTS	**FACETS**		
Is A	Toyota Corolla		
Dono	José Costa		
NºCilindros	4		
Modelo	4D Sedan		

Os f*rames*

- Proporcionam uma rápida visão do conhecimento que descrevem
- Os *facets default* podem considerar-se como representativos de senso-comum (p.e. um carro tem 4 rodas se nada for dito em contrário)

Contudo

- Como representações de esteriotipos (ideia inicial) levantam dificuldades uma vez que qualquer valor pode ser alterado num *child frame* não havendo geralmente mecanismos de protecção. Deste modo a noção de hierararquia e herança é posta em causa, tornando-se na prática cada *frame* um elemento primitivo.

2.2 Representação do Conhecimento

2.2.4 Lógica

Lógica Proposicional

• É a mais simples forma de lógica

• Expressa apenas factos (aquilo que existe num mundo). Cada proposição representa uma frase que expressa um facto

Símbolos

• Constantes: True, False

• Proposições: P, Q, etc.

• Conectivas: $\wedge \vee \neg \Rightarrow \Leftrightarrow$ (E, OU, NÃO, Implicação, Equivalência)

Proposições Atómicas: simplesmente P, Q

Exemplo: Está frio

Proposições Complexas: combinações de proposições atómicas através de conectivas, entre ()

Exemplo: ($P \wedge \neg Q$); É segunda-feira e não está frio

2.2 Representação do Conhecimento

A notação **P |- Q** é equivalente a $\frac{P}{Q}$ e significa que de P se pode inferir Q

Existem certos padrões de raciocínio cuja validade é sempre comprovável. São esses padrões que, formalizados, dão origem às Regras de Inferência. Eis algumas:

Modus Ponnens	**Modus Tollens**	**Resolução**
P $\underline{P \rightarrow Q}$ Q	$\neg Q$ $\underline{P \rightarrow Q}$ $\neg P$	$P \vee Q$ $\underline{\neg Q \vee R}$ $P \vee R$

Modus Ponnens

Está a chover

Se chover levo o guarda-chuva

Logo, levo o guarda-chuva.

Modus Tollens

Não levo o guarda-chuva

Se chover levo o guarda-chuva

Logo, não está a chover

Resolução

Está a chover ou a fazer sol

Não está a fazer sol ou está vento

Logo, está a chover ou está vento

2.2 Representação do Conhecimento

Lógica de Predicados (1ª Ordem)

- Na sua forma mais simples chama-se Lógica de Primeira Ordem
- Estende a Lógica Proposicional recorrendo também a **objectos**, **predicados** e **quantificadores**
- Consegue assim traduzir frases mais complexas
- A sintaxe do PROLOG baseia-se na Lógica de Predicados

Um predicado é simplesmente uma afirmação acerca de um objecto, uma declaração acerca de um atributo que ele possui, ou uma relação em que o objecto está inserido.

Exemplos (objecto)

(predicado)	
Mamífero (cão)	O cão é um mamífero
Pai (joão, alberto)	O João é pai do Alberto

Os quantificadores são o Universal, ∀, e o Existencial, ∃ (conhecidos da matemática)

Exemplos

∀(x) Gosta(x, Gelado)	Qualquer que seja *x*, *x* gosta de gelado
∃(x) Irmão(x,Tareco) ∧ Gato(x)	Existe um *x* tal que *x* é irmão do Tareco e *x* é um gato

2.3 Regras de Produção

A representação do conhecimento por Regras de Produção ou simplesmente Regras, é o mais comum, porque:

1. São fáceis de compreender
2. É fácil aprender a programar com tal sistema
3. Quase todas as ferramentas de desenvolvimento usam regras
4. Possibilitam validação do conteúdo (consistência, redundância, etc.)
5. A manutenção pode ser considerada não muito difícil, embora para sistemas complexos (centenas ou milhares de regras) possa não ser trivial pelas consequências que pode acarretar

Conforme referido (Turban & Aronson) definem:

1. **Regras Declarativas**: Declaram os factos ...
2. **Regras Procedimentais, de Inferência ou Meta-Regras**: Regras a respeito de regras...

2.2 Representação do Conhecimento

2.3.1 Generalidades

As regras têm a forma IF...THEN

- A primeira parte de uma regra (IF...) é a **premissa** (*LHS - Left hand side*)
- A segunda parte (THEN...) é a **conclusão** (*RHS - Rigth hand side*)

Exemplo:

SE o indivíduo tem carta há menos de 2 anos
ENTÃO o risco da seguradora é grande

Regras subentendem triplos OAV. O exemplo anterior pode ser entendido assim:

Premissa: Objecto=indivíduo
Atributo=tem carta
Valor=2 anos ou menos

Conclusão: Objecto=seguradora
Atributo=risco
Valor=grande

1. O valor do atributo da premissa destina-se a ser comparado (*matched*) com outro proveniente de um dado facto
2. O valor do atributo da conclusão destina-se a ser actualizado em função do resultado dessa comparação, constituindo assim um novo facto

2.2 Representação do Conhecimento

A **premissa** de uma regra pode conter várias **cláusulas** interligadas por conectivas lógicas do tipo AND (conjunctivas) ou OR (disjunctivas).

A **conclusão** pode conter várias claúsulas mas apenas ligadas por conectivas do tipo AND (a presença de um OR significaria que poderia existir alguma conclusão falsa, o que não faz sentido)

Exemplos

Claúsula 1

Claúsula 2

SE o céu está limpo E a temperatura é baixa
ENTÃO a hipótese de nevoeiro é grande E as plantas devem ser cobertas

SE o carro é novo E está em bom estado
ENTÃO deve pegar bem E a viagem deve ser segura

Note-se como seria inconclusiva a regra seguinte, por conter uma disjunção na conclusão:

SE o carro é novo E está em bom estado
ENTÃO deve pegar bem OU a viagem deve ser segura

2.2 Representação do Conhecimento

Os atributos que figuram numa regra devem ter associadas todas ou algumas das seguintes propriedades:

- **Nome**: o nome do atributo
- **Tipo**: numérico (80ºC), simbólico ("alta"), etc.
- ***Prompt* ou *Query***: a pergunta a apresentar ao utilizador quando for necessário o valor do atributo. Atributos que figuram em conclusões nunca poderão ter um *prompt* a eles associado
- **Domínio**: gama de valores que o atributo pode ter, p.e. [0, 100ºC], {branco, azul, vermelho}
- **Valores Especificados**: muitas vezes são dadas a escolher ao utilizador várias opções, p.e. o tipo de vinho que prefere ou as cores de que mais gosta. Neste caso ele deve escolher de uma lista ou responder a um query. Eventualmente poderá fazer uma escolha múltipla.
- **Factores de Certeza** (*Certainty Factors*): são números que indicam o grau de certeza de um facto ou de uma conclusão. Veja-se, por exemplo, o RHS de uma regra do MYCIN já apresentada:

...THEN there is a suggestive evidence **(0.7)** that the organism is streptococus

2.2 Representação do Conhecimento

Um atributo pode assumir 2 valores muito particulares:

- **Ainda não conhecido (*not yet known*)**: quando num dado momento não se conhece o seu valor mas é possível determiná-lo
- **Desconhecido (*unknown*)**: quando é impossível determinar o seu valor

De acordo com isto, uma cláusula da premissa designa-se por

- **Livre**: enquanto o atributo que nela figura for *ainda não conhecido*
  IF x = 50 ... mas *x* ainda não é conhecido
- **Verdadeira**: se o valor do seu atributo for conhecido e satisfizer a claúsula
  IF x = 50 ... e *x* é conhecido como tendo o valor 50
- **Falsa**: se o valor do seu atributo for conhecido e não satisfizer a claúsula
  IF x = 50 ... mas *x* é conhecido como tendo outro valor qualquer

2.2 Representação do Conhecimento

2.3.2 Propriedades das Regras

As regras têm as seguintes propriedades:

- **Nome**: o nome da regra
- **Premissa**: a premissa de regra
- **Conclusão ou Conclusão Intermédia**: a parte THEN da regra (RHS)
- **Notas e Referências**: anotações explicativas da regra
- **Factor de Certeza**: uma medida da confiança na conclusão da regra
- **Prioridade**: usada pelo motor de inferência
- **Custo**: usada pelo motor de inferência
- **Encadeamento** (*chaining*): usada pelo motor de inferência
- **Estado**: activa, inactiva, activada, disparada

Conclusão e Conclusão Intermédia

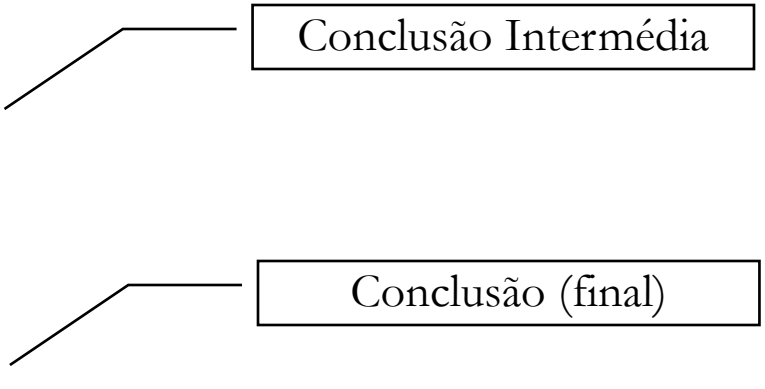
Uma Conclusão Intermédia serve como premissa para outra regra
Uma Conclusão (final) não serve como premissa para mais nenhuma regra

2.2 Representação do Conhecimento

Exemplo

SE o animal nasce vivo
ENTÃO o animal é mamífero

SE o animal é mamífero
E tem cor castanho claro
E é carnívoro
ENTÃO o animal é um leão



Prioridade e Custo

Nalguns *shells* é possível assinalar prioridades às regras, i.e. se houver várias prontas para disparar, determinar assim qual a primeira. No CLIPS, p.e., esta propriedade chama-se *salience*. Normalmente disparará a regra de maior prioridade e menor custo.

Encadeamento

Como veremos as inferências podem fazer-se em dois sentidos: "para a frente" e "para trás". Nalguns *shells* é possível indicar qual o modo preferido para cada regra, originando encadeamentos mistos.

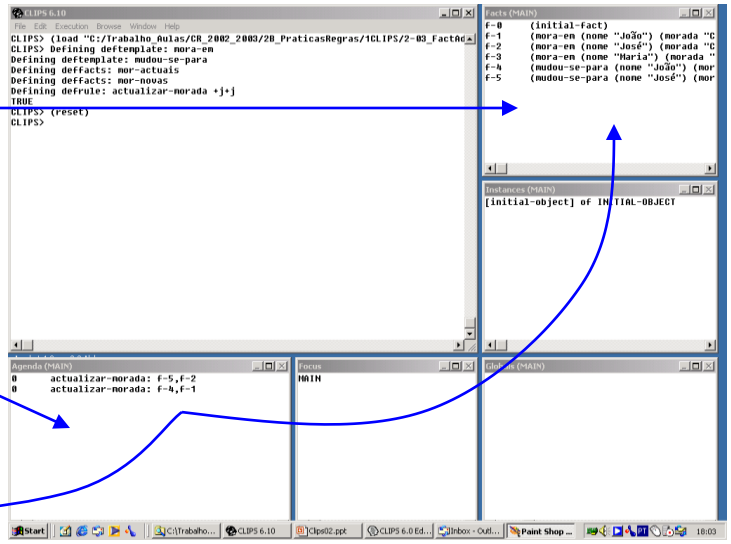
2.2 Representação do Conhecimento

Estado de uma regra

- **Activada** (*trigered*): uma regra é activada quando a sua premissa for verdadeira
- **Disparada** (*fired*): uma regra pode ser disparada apenas quando estiver activada. O disparo determina que a sua conclusão seja asserida (como verdadeira ou falsa)
- **Inactiva**: depois de disparada, uma regra torna-se inactiva (não voltará a ser activada nem disparada)
- **Activa**: uma regra "que aguarda" activação; ela poderá vir a ser activada e disparada, mas ainda o não foi porque a sua premissa nunca foi verdadeira

No CLIPS podemos ver, num dado momento:

- Os factos (na janela de factos)
- As regras activas (numa janela de regras)
- As regras quando forem activadas (elas aparecem então na janela da Agenda)
- As regras quando forem disparadas (saem da Agenda e surgem novos factos resultantes das suas conclusões)

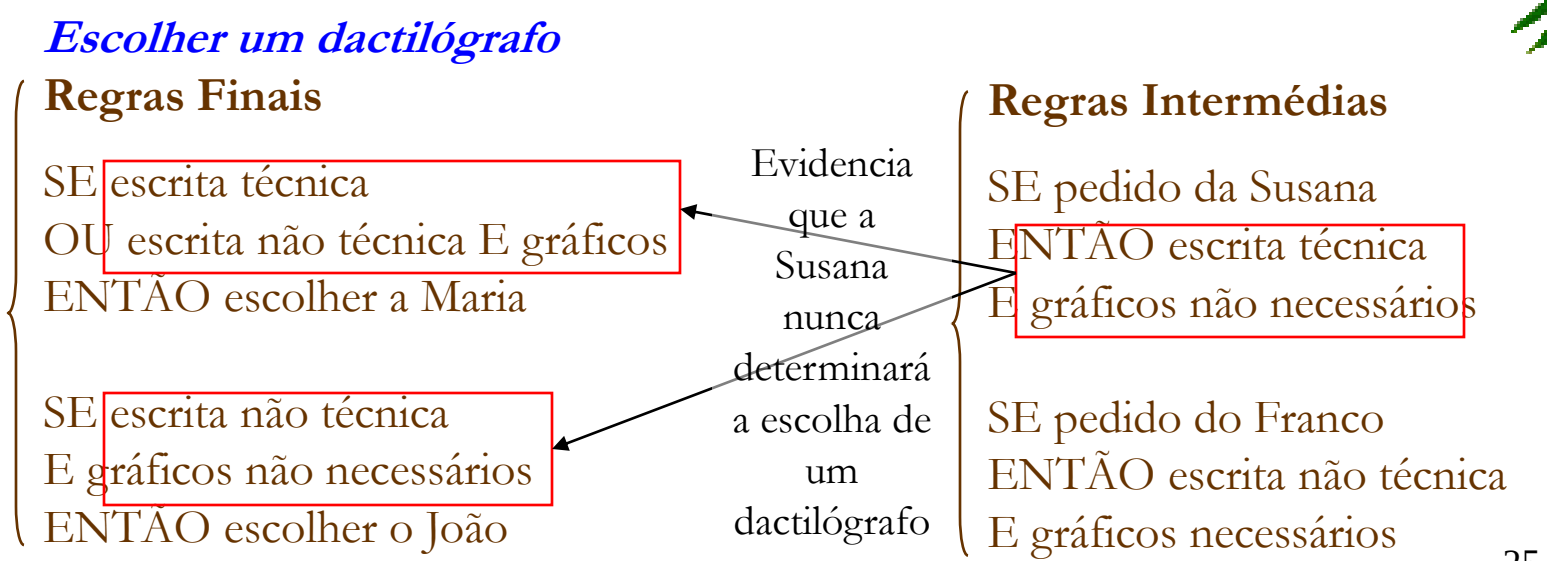


2.2 Representação do Conhecimento

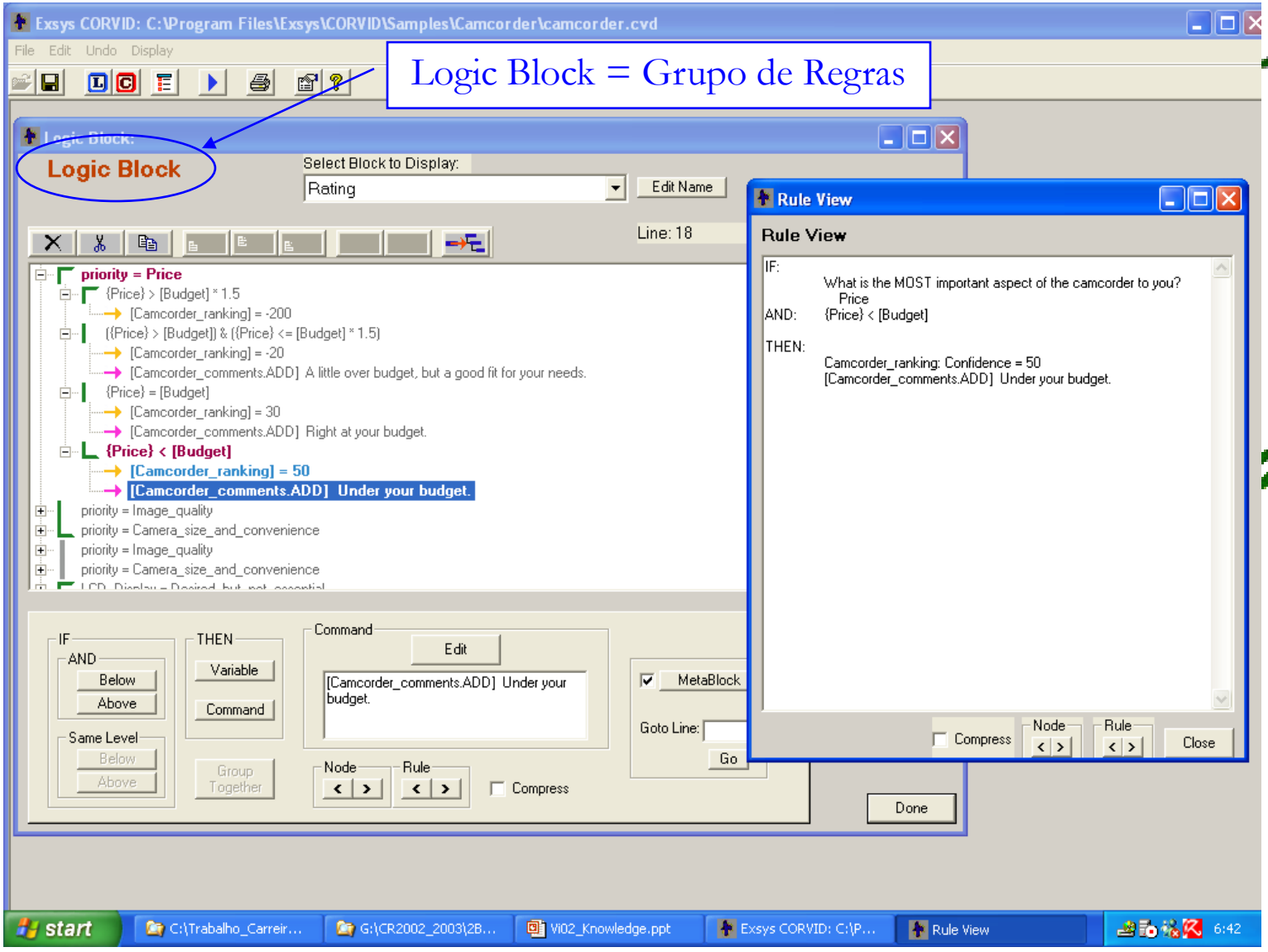
2.3.3 Agrupamento de Regras

Devido ao seu (eventualmente) elevado número, as regras devem **agrupar-se** de alguma forma. O ExSys, p.e., com o seu interface gráfico, permite realizar este agrupamento de forma clara e a gosto do utilizador.

Normalmente as regras serão agrupadas em função da sua conclusão: todas as que tenham como conclusão o objectivo pretendido, ficarão num grupo; as intermédias "anteriores" a estas, noutro; etc.



2.2 Representação do Conhecimento



2.2 Representação do Conhecimento

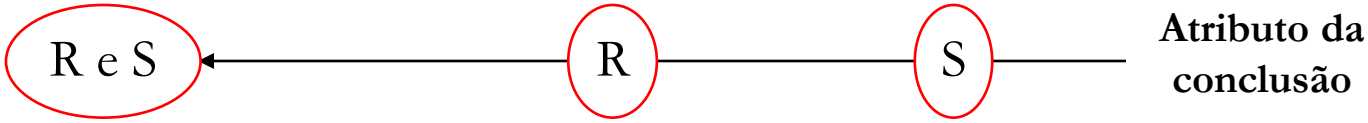
2.3.4 Conclusão Única

Consideremos as seguintes regras:

SE A=a
ENTÃO R=w E S=x

SE B=b
ENTÃO R=u

SE A=c
ENTÃO S=z



Nestas condições, e de acordo com o princípio de agrupamento anterior, como agrupar estas regras? Todas juntas ? 1 só grupo de 3 regras ?

A solução é evidente se em vez da regra 1 usarmos 2 regras: 1a) e 1b) de conclusão simples:

SE A=a
ENTÂO R=w

SE A=a
ENTÂO S=x

É agora evidente que a regra 1a) deverá ser agrupada com 2 e a regra 1b) com 3

2.2 Representação do Conhecimento

2.3.5 Simplificar com Cuidado

Consideremos a regra:

SE escrita técnica
OU escrita não técnica
E gráficos requeridos
ENTÃO escolher a Maria

Pode parecer equivalente a

SE gráficos requeridos
ENTÃO escolher a Maria

Mas, eliminando as 2 premissas (escrita técnica OU não técnica):

- Perde-se o conhecimento de que a Maria além de gráficos faz escrita técnica e não técnica
- Se for contratado um novo funcionário que só faz gráficos, ele ficará em iguais condições de selecção que a Maria
- Quando for necessário rever as regras para actualização do sistema, terá de se voltar a procurar o conhecimento perdido (o que sabe fazer a Maria?)

A eliminação de cláusulas nas premissas deve ser feita muito cuidadosamente.

2.3 Aquisição do Conhecimento

2.3.1 As Dificuldades da Aquisição

A transmissão de conhecimento Perito → *Knowledge Egineer* (**KE**) é difícil:

1. Os Peritos

- Para resolver um problema os peritos recolhem informação do mundo e por um processo mental complexo propõem uma solução:
  - Não é evidente como o fazem
  - Por vezes nem eles sabem explicar clara e metodicamente a sua decisão

2. A Máquina

- Pelo contrário, o computador exige clareza e detalhe

3. Os Intervenientes

- O diálogo entre os intervenientes numa sessão de aquisição de conhecimento pode ser muito difícil:
  - O **perito** e o **utilizador** podem nada saber de computadores
  - O ***KE*** e o **programador** podem nada saber acerca do domínio
  - O **vendedor** provavelmente pouco sabe sobre domínio e computadores
- Todos estes intervenientes usam **vocabulário diferente**

2.3 Aquisição do Conhecimento

Exemplo - Como recuperar *passwords*

"*impedance mismatch*" entre KE e Perito do domínio

Perito: Sam
Knowledge Engineer: Ken
...
Sam: Well, if it's a YP password, I log on as root on the YP master
Ken: Er...what's the YP master?
Sam: It's the diskfull machine that contains the database of network information.
Ken: 'Diskfull" meaning ...?
Sam: It has the OS installed on its local hard disk
Ken: Ah! So you log on...
Sam: As root. Then I edit the *password file*, remove the encripted entry and make the new *password map*
Ken: ...password map... What happens if you forgot your password ?
Sam: I could reboot to single user mode and edit */etc/password*. Or I could reload the whole system which I'd rather not do. Or I could use the *passwd* command.
Ken: Oh, Sorry -:)
...

2.3 Aquisição do Conhecimento

2.3.2 Métodos de Aquisição

A aquisição de conhecimento é um verdadeiro *botleneck* na implementação de SP. Os métodos utilizados nesta fase são:

1. Entrevistas
    1.1 Não estruturadas
    1.2 Estruturadas
2. Análise de Protocolo
3. Observação
4. Análise de Casos, Incidentes, esquemas e modelos, etc.
5. Relatórios elaborados pelos peritos
6. Guiados por computador
7. RGA- *Repertory Grid Analysis*
8. *Machine Learning*
    8.1 Indução (ID3) *Data-Mining*
    8.2 *Case-Based-Reasoning* (CBR)
    8.2 Redes Neuronais
    8.3 Agentes Inteligentes

2.3 Aquisição do Conhecimento

2.3.2.1 Entrevista

Não Estruturada

É centrada no trabalho do *knowledge engineer* (KE) que conduz o encontro entre ele e o perito, sem que exista qualquer suporte ou tenha havido alguma preparação especial.

O KE actua como estudante e o perito como mestre:

- Descrevendo o que faz, como o faz e porquê
- Ensinando como interpretar documentos em análise

Método moroso e dispendioso

Estruturada

É orientada por certos procedimentos, alguns a realizar antes da própria entrevista.

Método mais sistemático, divulgado, mas podendo introduzir polarizações provocadas pelo próprio KE .

2.3 Aquisição do Conhecimento

Exemplo - Escolher a localização para ampliação de um hospital

Perito: E, **Knowledge Engineer**: KE

Entrevista não estruturada

...

KE: So, what's the most important factor in determining where to put a new facility?

E: It's demographics. I've noticed that facilities must be located near people.

KE: Is it also important that it not be close to another facility?

E: Not necessarily. It is important that it not be located too close to the main hospital or another of our facilities but if the population density is high enough we can locate it near one of our competitors.

KE: What kind of demographics are you looking for?

E: At least 2000/square mile over about 4 square miles for a profitable clinic

KE: What about competitor's locations?

E: If a competitor's is within 2 miles, the density has to exceed 3500 people/square mile. And if there are 2 competitors within 2 miles of each other there's no point in even trying, except for some special services.

KE: We'll talk about special services latter. What about income? Is it important?

E: Not really...well yes. We must limit our indigent cases to no more than 2% and we look for an average family income of about 30.000/year. We have a facility near a low-income area and near a bus line.

KE: Is being near public transportation important?

...

2.3 Aquisição do Conhecimento

Entrevista estruturada

Linhas Gerais para uma Entrevista Estruturada

1. Fazer algum estudo prévio do domínio
2. Rever as capacidades do perito a entrevistar
3. Delinear perguntas a realizar
4. Planear a entrevista: disposição física, objectivos e agendas, principais temas a focar
5. Assegurar que o perito compreende os objectivos
6. Tentar que o perito também prepare a entrevista
7. Durante a entrevista, não fugir do plano previamente traçado
8. Usar técnicas de entrevista (psicologia)

Recomendação adicional

O KE poderá entrevistar utilizadores ou peritos "menores" antes da entrevista com o perito "principal" de modo a perceber os contornos do problema

2.3 Aquisição do Conhecimento

2.3.2.2 Análise de Protocolo

Faz parte de um conjunto de métodos genericamente designados por *Tracking Methods* originários da psicologia cognitiva.

- Consiste em documentar a forma de pensar, a abordagem seguida por um perito na resolução de um problema
- Pede-se ao perito que *pense alto* descrevendo o raciocínio seguido
- A documentação consiste normalmente numa **gravação**. Este suporte é designado por ***protocolo***
- Os elementos recolhidos são depois **analisados** para se apreenderem os elementos mais significativos e transformá-los em regras de produção

Vantagens

- Consciencialização de heurísticas, alternativas, valores de atributos, etc.
- Observação directa do processo de decisão com análise *a posteriori*

Desvantagens

- O perito tem de ter consciência do seu método de decisão
- A interpretação das decisões pode ser subjectiva (as explicações podem não justificar completamente uma dada decisão)

2.3 Aquisição do Conhecimento

2.3.2.3 Observação

Pode considerar-se também um *Tracking Method*:

- Consiste na observação (passiva) da actividade de um perito
- Esta actividade pode documentar-se numa **gravação** e centra-se:
  - Na actividade motora (onde vai, o que manipula, o que diz)
  - Na atenção (p.e. onde fixa o olhar, onde se detém)

2.3.2.4 Outros métodos

- **Análise de Casos**: como foram resolvidos problemas passados
- **Análise de Incidentes Críticos**: o mesmo que o anterior mas aplicado a um conjunto de casos especiais, *memoráveis*
- **Discussão** com os utilizadores
- **Gráficos e Modelos**: a desenhar pelo perito como documentação do seu raciocínio
- ***Brainstorming***: sessões com vários peritos em simultâneo
- **Protótipos**: modelos simplificados do domínio
- **Métodos de Escalonamento**: (*multidimensional scaling, Johnson's hierarchical clustering*): identificam dimensões de conhecimento...

2.3 Aquisição do Conhecimento

2.3.2.5 Relatórios e Questionários Periciais

Baseiam-se em questionários a que o perito deve responder e em relatórios que deve elaborar.

- Exigem esforço e disciplina por parte dos peritos que, ao longo do tempo, tendem a deixar de os elaborar com a devida meticulosidade
- Os peritos tendem a descrever como idealmente um problema deveria ser resolvido, não como o foi, realmente.
- Requerem conhecimento, por parte dos peritos, de certas ferramentas (diagramas de fluxo, p.e.)
- Os peritos tendem a ser vagos e esquecer pormenores para eles evidentes

2.3.2.6 Guiados por Computador

Recorrem a aplicações que guiam a aquisição de conhecimento:

- REFINER+: baseia-se na análise de casos numa base de dados de doentes. Infere um protótipo que descreve cada classe de casos, a rotular pelo perito, que pode fazer modificações.
- TIGON: para análise de um sistema de diagnóstico de falhas em turbinas, infere regras de analogia automaticamente.

2.3 Aquisição do Conhecimento

Gráficos, diagramas, modelação visual em geral, são outros meios de aquisição de conhecimento recomendados e frequentemente utilizados:

- São auxiliares de memória; modelizam o mundo real; podem revelar inconsistências

2.3.2.8 *RGA - Repertory Grid Analysis*

O RGA é um método do tipo Entrevista de Classificação. Tem como objectivo obter, de forma sistemática, quais os atributos de um dado conjunto de entidades que influenciam a decisão por esta ou aquela solução.

Baseia-se numa teoria psicológica (*Personal Construct Theory*) em que cada ser humano classifica as suas percepções do mundo de forma personalizada, segundo um modelo perceptual próprio. Decide e age em função destas classificações.

Exemplo

Seleccionar uma linguagem de programação

Para construir um SP destinado a aconselhar uma dada linguagem de programação para um certo fim, o RGA poderia ser aplicado assim:

2.3 Aquisição do Conhecimento

1. Numa entrevista, o perito identifica os objectos importantes no domínio considerado. Por exemplo LISP, PROLOG, C++ e COBOL
2. O perito identifica os atributos importantes de cada objecto, p.e. *disponibilidade comercial*, *orientação*, *facilidade de programação* e de *tempo de aprendizagem*
3. Para cada atributo, o perito estabelece uma escala bipolar (i.e. factores positivos e negativos) de características (perfil) de cada atributo. Por exemplo:

RGA

Atributos	Traço	Oposto
Disponibilidade	Alta	Não disponível
Facilidade de Programação	Alta	Baixa
Tempo de Aprendizagem	Baixo	Alto
Orientação	Simbólica	Numérica

4. O KE selecciona ternos de objectos e pergunta "Que atributos e traços distinguem 2 quaisquer destes objectos do terceiro?" As respostas são dadas numa escala (1-3) ou (1-5). Este processo é repetido para todos os ternos.

Atributos	Orientação	Fac.Prog.	Tempo Ap.	Disponib.
Perfil	Simbólica (3)	Alta (3)	Alto (3)	Alta (3)
Oposto	Numérica (1)	Baixa (1)	Baixo (1)	Baixa (1)
LISP	3	3	1	1
PROLOG	3	2	2	2
C++	3	2	2	3
COBOL	1	2	1	3

5. Depois de completada a grelha, o perito pode fazer alterações. A grelha final pode ser usada como base para recomendações. Neste exemplo, "SE orientação numérica muito importante ENTÃO escolher o Cobol".

2.3 Aquisição do Conhecimento

Exemplos de ferramentas desenvolvidas com base no RGA (objectivo: ajudar a conceptualização do domínio):

- ETS - Expertise Transfer System (Boose & Gaines, 1990): proporcionou tempos de aquisição de cerca de 2 horas na construção de protótipos de SP muito simples
- AQUINAS (Boose & Bradshaw, 1993): incorpora o ETS e estende-o permitindo a representação do conhecimento em hierarquias
- KRITON (Diederich et al. 1987): conduz as entrevistas com os peritos. Analisa protocolos e documentos. Proporciona estatísticas de palavras-chave que guiam os peritos na selecção de textos relevantes. Nalguns casos pode substituir um KE.
- PCGRID (Univ. de Leighigh)
- WebGrid: primeira ferramenta com acesso à Web para aquisição de conhecimento
- Circumgrids (Univ. South Florida)

2.3 Aquisição do Conhecimento

2.3.3 Ajuda e Suporte ao KE

Na sua tarefa, o KE pode ser auxiliado por uma série de ferramentas desde as destinadas apenas a simplificação, até à aquisição automática através de algoritmos da área *Machine Learning* (secção seguinte):

1. Editores e Interfaces

- A simples introdução de Regras de Produção pode ser muito facilitada pelo interface do *shell* utilizado.
- O editor de regras do ExSys é um exemplo disso (testa sintaxe, mostra regras em "linguagem natural", agrupa-as em blocos lógicos, etc.)

2. Explicação e Justificação

- O módulo de explicação e justificação de um SP, respondendo a perguntas típicas tais como WHY, HOW, WHY NOT e WHAT IF, ajuda a compreender o funcionamento dos SP (principalmente se complexos)

2.3 Aquisição do Conhecimento

3. TEIRESIAS

- O EMYCIN (Empty Mycin) é uma versão do MYCIN original mas que não incorpora conhecimento (i.e. basicamente trata-se do motor de inferência que pode assim ser usado noutro domínio)
- O TEIRESIAS (um exemplo clássico) proporciona interface em linguagem natural no desenvolvimento de SP baseados no EMYCIN: p.e. permite mostrar todas as regras que têm a mesma conclusão e corrigir as necessárias ou adicionar outras, escrevendo em linguagem natural
- Também detecta inconsistências e conflitos entre regras

4. KADS

KADS - Knowledge Aquisition and Documentation System: trata-se de uma linguagem de modelização formal destinada a ajudar o KE nos processos de aquisição e documentação do conhecimento adquirido.

5. ACQUIRE

ACQUIRE - Interage com o perito (sem necessidade de um KE), filtra respostas e gera regras IF...THEN. Trata-se na realidade de uma ferramenta de indução (ver mais adiante)

2.3 Aquisição do Conhecimento

6. empolis e:kms

A **empolis**, na Alemanha (ex-tecInno) comercializa um produto designado por e:kms - *knowledge management suite*.

About empolis

empolis
BERTELSMANN MOHN MEDIA GROUP

Information is the basis for decisions, a prized commodity, a company value and the basis for our communication. Quick and easy access to the required information, its structures, its links and distribution networks is often more essential than the information itself. This applies to both internal organizational processes and to the dialog with the customer.

empolis is the leading supplier of software solutions with extensive industry experience and corresponding services for enterprise content and knowledge management. With the empolis product lines, e:kms, e:catalox, and e:solutions, an organization's information is transformed into indispensable and profitable capital that achieves a fast return on investment.

empolis has influenced the development of innovative industry standards such as XML, Java and TopicMaps and, in turn, these standards have decisively influenced the development of empolis products for the benefit of our customers.

empolis is part of arvato – a Bertelsmann company – and employs 320 people in Germany, Hungary, Scandinavia, Poland, UK and the US.

empolis – Transforming Information into Value

R&D @ empolis:

1. Intelligent Agent Technology
2. Peer-to-peer networks
3. Advanced distributed architectures
4. Automated text analysis
5. Ontologies and Semantic Web technologies
6. Knowledge Engineering
7. Case-Based Reasoning
8. Machine Learning
9. Knowledge Management
10. Methodologies and Processes
11. ANSI Standardisation Work

http://www.empolis.com/

2.3 Aquisição do Conhecimento

k42

empolis k42 Knowledge Server

Intelligent Retrieval with Topic Maps

Integração de conhecimento:
Local + Web

k42 is an innovative product that forms the basis for answering the challenges of the new information revolution. k42 is a highly extensible object-orientated product with a powerful API that users can utilise for their own application needs. The product is cross-platform and adaptable, so that it can enable users to capture and expose complex knowledge within the whole range of their infrastructures.

Add value to existing data.

By adding structure where there is none, we can query and re-use knowledge. The usefulness of existing resource can be extended if they are represented within a structured knowledge framework.

Connect data (with semantics) in disparate heterogeneous data sources.

To unlock data it needs to be connected in meaningful ways: these connections together with the data can be delivered to users of knowledge. It shouldn't matter where the data is stored if it is important information to the user. There should be no reason why data from lots of different types of repositories should not be connected together at the »knowledge layer«.

Connect different Data Formats together in the »knowledge layer«.

The knowledge layer is the Topic Map space where the ideas that are known about data are connected and used. This neutral form layer can sit on top of any proprietary system so that a user can navigate across different format domains.

Capture the knowledge of the corporation.

Information and experience in peoples head can be connected with data. This creates an intelligent resource that can be stored,navigated, re-used and expanded.

2.3 Aquisição do Conhecimento

2.3.4 Aquisição Automática

O termo Aquisição Automática de Conhecimento (outros termos próximos também utilizados: *Machine Learning*, *Knowledge Discovery* e *Data-Mining*) engloba 4 tópicos:

- **Raciocínio Baseado em Casos** (CBR): a estudar em secção própria
- **Redes Neuronais**: introduzidas em Sistemas Inteligentes, naturalmente que representam conhecimento dado que aprendem a reconhecer contextos, objectos. Por exemplo um caracter, um número, uma palavra.
- **Agentes Inteligentes** (*software robots*): refira-se o KQML - *Knowledge Query and Manipulation Language* - uma linguagem e protocolo para troca de mensagens entre agentes para intercâmbio de conhecimento e o KIF - *Knowledge Interchange Format* - um protocolo para troca de conhecimento entre programas diferentes (http://www.cs.umbc.edu/kqml/).
- **Indução**: O algoritmo mais vulgar chama-se ID3 e produz uma árvore de decisão a partir de exemplos (o processo de geração da árvore tem, sob este aspecto, algo de comum com o treino das redes neuronais: o princípio é geral - "nada compreendemos que não tenha passado anteriormente pelos nossos sentidos"). A estudar em detalhe noutra secção.

2.3 Aquisição do Conhecimento

Uma Tabela de Indução (ou Mapa de Conhecimento) pode ser usada para geração de regras IF...THEN. Ela representa atributos e os seus valores num dado domínio, seguidos da decisão respectiva.

Exemplo - Tabela de Indução para escolha da localização de ampliação de um hospital (a entrevista foi dada num exemplo anterior)

Densidade de População	Densidade ao longo de quantos Km2 ?	Nº de concorrentes num raio de 2 Km	Rendimento médio por família	Transporte público perto?	Decisão
>2000	>4	0			Yes
>3500	>4	1			Yes
		>2			No
			>30,000 USD		No

NOTA:
O ID3 detecta os atributos mais relevantes (maior *ganho de informação*) e pode concluir que um certo atributo é irrelevante para a decisão

A tabela será completada e refinada ao longo das entrevistas seguintes (p.e. *transporte público* ainda não foi considerado nesta fase).
Cada linha é candidata a uma futura regra IF...THEN. Por exemplo, da 1ª linha:

IF densidade > 2000 AND densidade ao longo de mais de 4 Km^2 AND concorrentes = 0 THEN localização aceitável