Engenharia de Sistemas e Informática

Engenharia do Conhecimento

5º Ano 2º Semestre

Escola Superior de Tecnologia de Viseu

Departamento de Informática

CBR

RACIOCÍNIO BASEADO EM CASOS

Escola Superior de Tecnologia de Viseu

Instituto Politécnico de Viseu

Carlos Simões

ÍNDICE

1 Introdução ao Raciocínio Baseado em Casos 1
1.1 Motivação 1
1.1.1 Representação da Informação 1
1.1.2 Representação de Dados num Computador 2
1.1.3 Uma Representação Mais Poderosa 4
1.1.4 Sistemas Periciais Baseados em Regras 4
1.1.5 Limitações das Regras 7
1.1.6 Os elefantes nunca esquecem 7
1.1.7 Já tenho a resposta… Qual é a pergunta? 9
1.2 Algumas definições de CBR (Raciocínio Baseado em Casos) 10
1.3 Exemplos da vida quotidiana 11
1.4 Evolução do Paradigma do CBR 11
1.4.1 Teorias da Memória 11
1.4.2 Problemas com os Sistemas Baseados em Regras 12
1.5 Vantagens da Abordagem CBR 12
1.5.1 Credibilidade Psicológica 12
1.5.2 Casos *vs.* Regras 12
1.6 O Mecanismo Natural de Aprendizagem 13

2 O Processo de Raciocínio Baseado em Casos 15
2.1 O Algoritmo Básico do CBR 15
2.2 O Ciclo CBR 16
2.3 A Hierarquia de Tarefas no CBR 18
2.3.1 Recolha de Casos 19
2.3.2 Reutilização de casos 21
2.3.3 Revisão de casos 21
2.3.4 Retenção de casos - Aprendizagem 22

3 Uma descrição mais detalhada do CBR 25
3.1 Áreas problemáticas no CBR 25
3.2 Representação de Casos 25
3.2.1 O que é um caso? 25
3.2.2 O Papel do Conhecimento Geral 26
3.2.3 O Conteúdo dos Casos 26
3.3 Indexação de Casos 32
3.3.1 Indexação com base em listas 35
3.3.2 Indexação com base em diferenças 35
3.3.3 Indexação com base em explicações 36
3.3.4 Indexação com base em vários métodos 37
3.4 Armazenamento de Casos 39
3.4.1 O Modelo de Memória Dinâmica 40

3.4.2 O Modelo Exemplar-Categoria....42
3.5 Estruturas de Memória e Algoritmos de Pesquisa....43
3.5.1 Memória plana com pesquisa em série....44
3.5.2 Memória hierárquica e árvores de descritores partilhados....44
3.5.3 Memória hierárquica e árvores de discriminação....45
3.5.4 Memória hierárquica e árvores de discriminação com redundância....47
3.5.5 Memória plana com pesquisa em paralelo....49
3.5.6 Memória hierárquica e pesquisa em paralelo....49
3.6 Adequação Parcial e Ordenamento....50
3.6.1 Detecção de correspondências....51
3.6.2 Cálculo do grau de adequação entre conjuntos de características....53
3.6.3 Atribuição de relevância às dimensões da representação....54
3.6.4 Funções numéricas de ordenamento....54
3.6.5 Ordenamento dinâmico baseado em múltiplos valores de relevância....55
3.6.6 Ordenamento dinâmico baseado em heurísticas de preferência....55
3.7 Adaptação de Casos....56
3.7.1 Métodos de Substituição....58
3.7.2 Métodos de Transformação....60
3.7.3 Outros Métodos....61
3.7.4 Avaliação e Correcção....62
3.8 Aprendizagem....63

4 Aplicações do Raciocínio Baseado em Casos....67
4.1 O CBR na Resolução de Problemas....68
4.1.1 CBR para Projecto....68
4.1.2 CBR para Planeamento....69
4.1.3 CBR para Diagnóstico....70
4.2 CBR Interpretativo....71
4.2.1 Raciocínio Justificativo e Adverso (Concorrente)....72
4.2.2 Classificação e Interpretação....72
4.2.3 Projecção de Resultados....73
4.2.4 O CBR Interpretativo e a Resolução de Problemas.....73

5 Bibliografia....74

1 Introdução ao Raciocínio Baseado em Casos

1.1 Motivação

Esta secção faz uma breve introdução à forma como os computadores representam os dados e como podemos extrair o significado ou informação a partir dos dados. São referidos os conceitos subjacentes às bases de dados relacionais e orientadas a objectos e é mostrada a forma como podem ser usadas para executar *querys* simples. São depois introduzidos os conceitos por trás do **raciocínio baseado em regras** e dos **sistemas periciais**. Os problemas e limitações dos **sistemas periciais baseados em regras** são discutidos e confrontados com a forma como usualmente resolvemos problemas. São apresentadas em seguida as principais características do Raciocínio Baseado em Casos (CBR - *Case Based Reasoning*).

1.1.1 Representação da Informação

```
Como é que os computadores representam os dados
e como se obtém informação a partir dos dados?
```

Os computadores são usados para manipular informação, seja através de um programa de processamento de salários, de um processador de texto, de um jogo ou de um sistema de apoio à decisão. Os programas de computador usam diversos métodos para processar a informação e as aplicações são o resultado de sucessivas tentativas para melhorar e alargar a capacidade dos computadores processarem a informação automaticamente.

Os computadores efectuam transformações simples sobre símbolos. As transformações primitivas eram definidas pelos circuitos eléctricos nos *chips* que constituíam a unidade de processamento central. Em termos binários, estas transformações primitivas efectuam funções tais como somar duas sequências de números binários, subtraí-las, fazer deslocamentos e comparações.

É importante reconhecer que os computadores "não sabem" aquilo que fazem. Os seus programas são projectados para "imitar" a forma como nós representamos e efectuamos tarefas. Por exemplo, se escrevermos uma carta usando um processador de texto, o computador não sabe o que cada uma das palavras significa, não sabe sequer que são palavras.

O computador tem de usar uma representação interna para os símbolos (letras e palavras) de modo a conseguir manipulá-las. O computador tem assim de transformar os *inputs* numa representação que possa manipular e os *outputs* numa representação que nós possamos entender (figura 1).

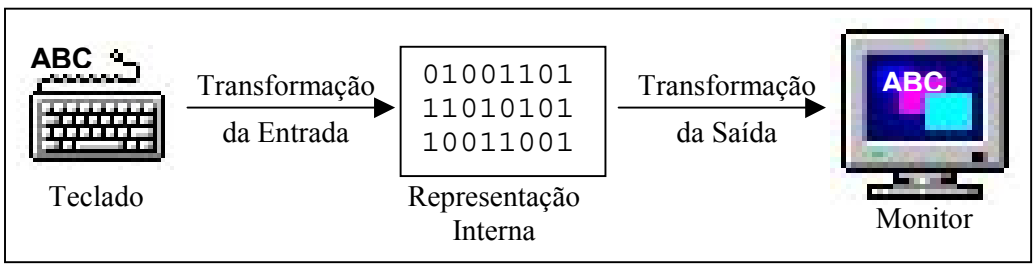

Figura 1 – Transformação de e para a representação interna do computador.

O desenvolvimento de um programa que possa manipular dados de forma útil envolve os seguintes passos:

1. Definir um esquema de apresentação dos símbolos ao computador que represente fielmente aquilo que pretendemos.
2. Definir quais as transformações permitidas sobre os dados e que são úteis para nós.
3. Definir um formalismo que permita ao computador apresentar os resultados ao utilizador de uma forma com significado.

Felizmente, grande parte deste trabalho base foi já feita, e muitos códigos e normas internacionais foram já definidos. Por exemplo, o ASCII (*American Standard Code for Information Interchange*) é um código normalizado internacionalmente para representar caracteres alfanuméricos e alguns símbolos especiais. Além disso, existem inúmeras linguagens de programação de alto-nível que disponibilizam uma diversidade de funções que permitem a transformação de símbolos para o utilizador. Assim, o processamento de dados é ajudado pela definição de representações apropriadas para os dados e de transformações sobre os dados.

É, no entanto, fundamental perceber que a escolha do tipo de representação define quais os tipos de transformações que são possíveis.

1.1.2 Representação de Dados num Computador

As bases de dados impõem uma estrutura nos dados que armazenam. Isto porque o significado (ou informação) que se pode retirar dos dados está contido nas relações entre os itens.

Na figura seguinte os dados elementares não fornecem, por si só, qualquer significado. A informação está contida nas relações entre os dados elementares, quando estes são colocados numa estrutura (*record*). Podemos reconhecer que são endereços e que o Sr. João Branco mora na Rua Costa do Castelo, 31, no Porto. Os sistemas de bases de dados definem representações de alto nível para as relações entre os dados individuais. As bases de dados relacionais e as bases de dados orientadas a objectos são duas abordagens diferentes que permitem resolver o problema de estruturar os dados por forma a que tenham significado.

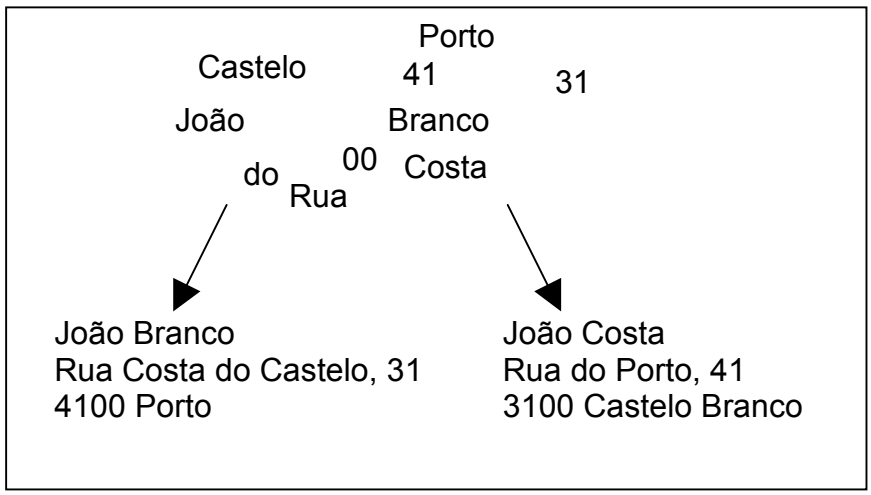

Figura 2 - Dados e relações.

Nas bases de dados relacionais, não só para reduzir o espaço de armazenamento, mas também para assegurar uma mais fácil manutenção, os itens de dados são apenas guardados uma vez. As relações entre os diversos itens são definidas usando tabelas relacionais e códigos de identificação - as chaves.

Assim, a partir da figura 3 podemos determinar que o ordenado do João Costa é 900 E através da tabela relacional Categorias Profissionais. Se os dados estiverem guardados numa base de dados relacional, uma linguagem de interrogação (*query language*) permite-nos perguntar qual o ordenado do João Costa. Além disso, se os ordenados da companhia forem aumentados de 5%, podemos aplicar uma função que actualize os ordenados na tabela Ordenados, sem alterar qualquer outro dado em qualquer outra tabela.

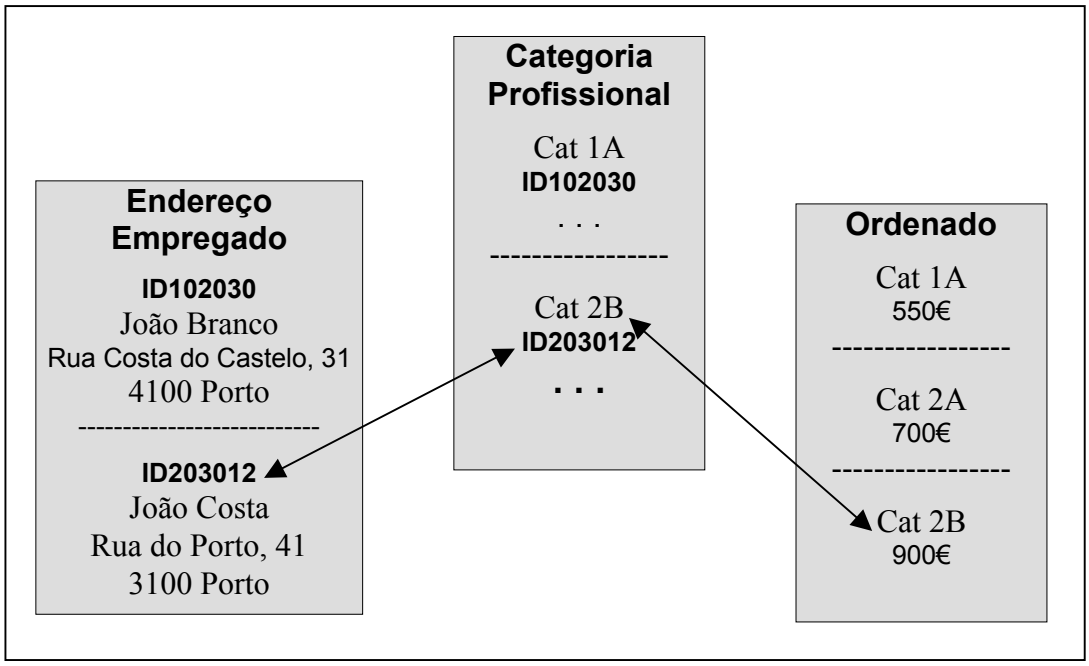

Figura 3 – Tabelas Relacionais.

Uma forma alternativa de representar os dados é usar técnicas orientadas a objectos. Numa base de dados orientada a objectos, as estruturas individuais são guardadas como instâncias de classes. Deste modo, como se pode verificar na figura 4, João Costa e João Branco são representados como instâncias da classe "`Empregados`".

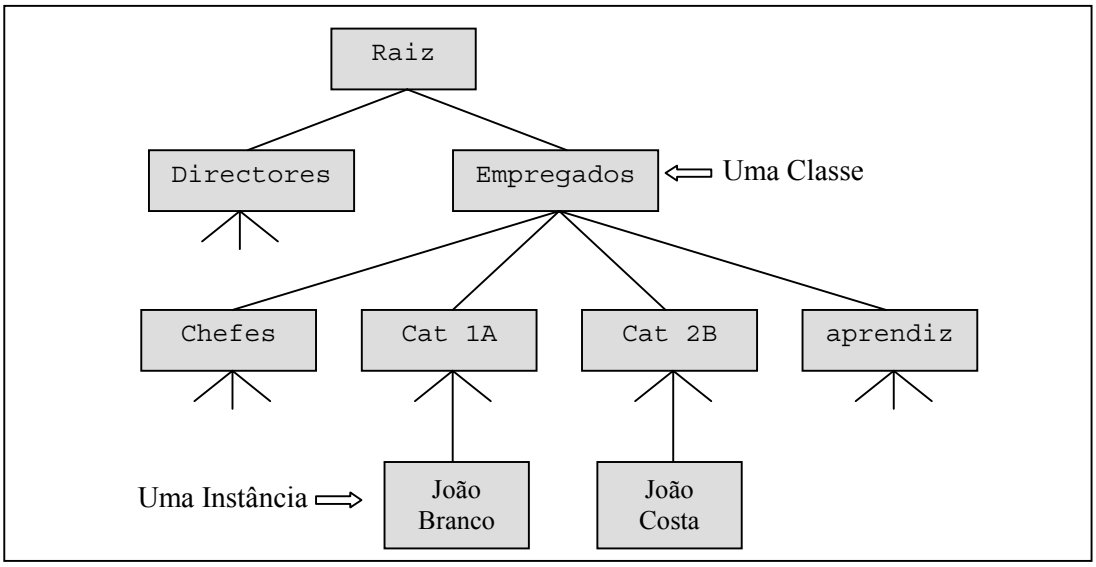

Figura 4 – Uma hierarquia de objectos.

A classe "`Empregados`" define tudo o que é comum a todos os empregados. Isto é, todos têm um nome, número de segurança social, morada, departamento e categoria profissional. A hierarquia de objectos pode ser expandida para classificar determinadas categorias profissionais, como na figura 4. A instância que representa João Branco é classificada como sendo da categoria profissional 2B. Através de um processo de herança, João Branco herda o ordenado da classe 2B. A classificação das entidades do mundo real em hierarquias de objectos e a utilização de processos de herança fornece às bases de dados orientadas a objectos uma capacidade de representação consideravelmente superior às bases de dados relacionais.

1.1.3 Uma Representação Mais Poderosa

Embora exista uma relação entre João Branco e o seu ordenado, esta está implícita na estrutura das tabelas relacionais. A base de dados orientada a objectos classifica explicitamente João Branco como uma instância da categoria profissional 2B. Qualquer alteração no ordenado dessa categoria será herdada por todas as instâncias dessa classe. Uma abordagem alternativa para representar essa informação é através da utilização de declarações lógicas.

Por exemplo, o facto da empresa XPTO empregar o João pode ser representado em PROLOG como:

```
        empregos(XPTO,João)
ou      empregos(João,XPTO)
```

O facto de João, Nuno, Marco e Pedro serem todos empregados da empresa XPTO pode ser representado por:

```
empregos(XPTO,[João,Nuno,Marco,Pedro])
```

A programação lógica torna-se mais expressiva usando variáveis, operadores lógicos (`AND`, `OR` e `NOT`) e declarações `IF-THEN`. Isto permite definir **Regras**, por exemplo, a regra de que a pessoa A normalmente ganha mais que o empregado X que é chefiado por A pode ser representada por:

```
ganha_mais(A,X) IF chefia(A,X)
```

Então, se afirmarmos que João chefia Nuno e Marco e é chefiado por Pedro:

```
chefia(João,Nuno)
chefia(João,Marco)
chefia(Pedro,João)
```

podemos inferir, pela aplicação da regra definida atrás, que:

`ganha_mais(João,Nuno)` é verdadeira, enquanto
`ganha_mais(João,Pedro)` é falsa.

Nesta representação, o conhecimento de quem chefia quem é definido explicitamente pela declaração. Além disso, a regra para determinar quem ganha mais é também definida explicitamente. É assim possível, através da utilização desta representação, inferir novos factos a partir dos dados e regras existentes, tal como o facto de João ganhar mais que Nuno.

1.1.4 Sistemas Periciais Baseados em Regras

A capacidade para definir as regras no formato `IF-THEN` descrito atrás tem diversa vantagens:

- As regras podem ser facilmente "entendidas" pelos programadores e pelos peritos de uma forma idêntica.
- As regras podem conter pequenos "pedaços" de conhecimento, que colectivamente podem modelar um problema bastante complexo.
- As regras são independentes entre si.
- As regras podem ser colocadas por qualquer ordem num programa.

No essencial, as regras permitiram aos investigadores na área da investigação artificial (IA) representar o conhecimento para a resolução problemas sob a forma de modelos que podem ser implementados computacionalmente. Nos sistemas baseados em regras, o conhecimento é representado como factos, i. e., relações entre entidades, e regras para manusear os factos. Esta aparente simplicidade é dificultada pelas situações em que mais do que uma regra se pode aplicar e ainda pelo facto de, quando se aplica uma regra, outras passam também a ser aplicáveis. Deste modo, um programa baseado em regras necessita de uma estrutura de controlo que permita determinar qual a próxima regra a aplicar e como encadear as regras.

Consideremos as seguintes regras simples:

```
IF A THEN B
IF B THEN C
IF C THEN D
```

Se A é verdadeira, podemos inferir logicamente que D é verdadeira pela aplicação das três regras. Isto pode ser representado esquematicamente como na figura 5.

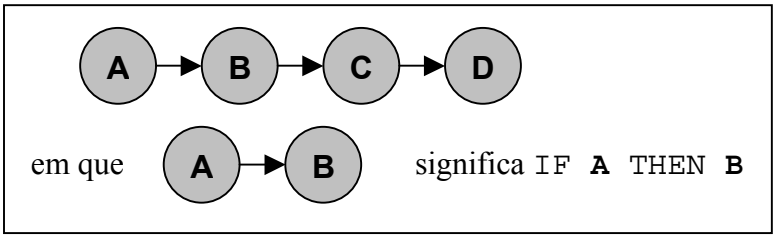

Figura 5 – Representação Gráfica de Regras.

Um conjunto de regras mais complexo, como:

```
IF A THEN B & C
IF B THEN D
IF C THEN E
IF D THEN G
```

pode ser representado como na figura 6.

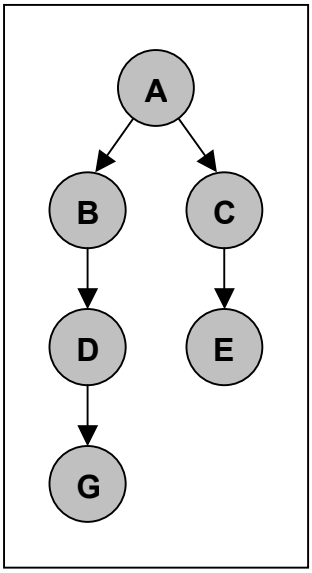

Figura 6 – Árvore de Regras.

O problema, nesta situação, é a ordem pela qual as regras são aplicadas - a menos que o computador possua processadores paralelos, é necessário definir qual a regra que deve ser

accionada primeiro. Se A for verdadeira, é melhor tornar B verdadeira antes de C ou o contrário? De facto, é tarefa do programador decidir, mas uma estrutura frequentemente usada e denominada *depth-first search* faz com que inferências sejam feitas pela ordem indicada pelos números na figura 7.

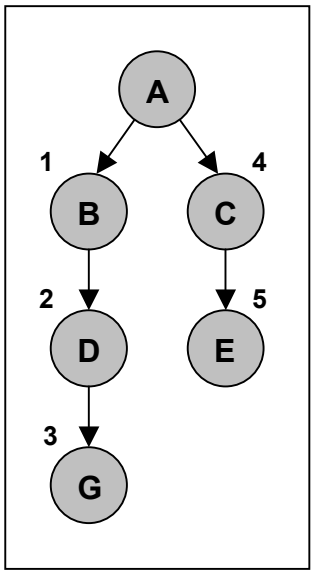

Figura 7 - Depth-first forward chaining.

Deste modo o sistema infere que G é verdadeira antes de inferir que E é verdadeira. Na realidade, se G for uma solução, pode não ser necessário inferir que E é também uma solução potencial. Esta estrutura de controlo é mais correctamente denominada *depth-first forward chaining*. *Forward* significa que novos factos são inferidos de factos já conhecidos. É também possível usar o mesmo conjunto de regras pela ordem inversa - *backward chaining* - para descobrir quais as condições que necessitam de ser verdadeiras para que uma condição seja também verdadeira. Por exemplo, usando o mesmo conjunto de regras, podemos perguntar "Quais as condições que têm que ser verdadeira para que G seja verdadeira"?

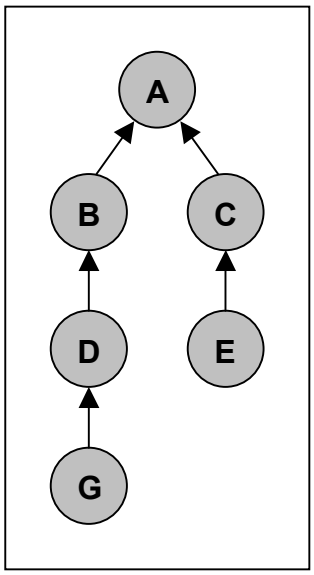

Figura 8 – *Backward chaining*.

A resposta é "A". A estrutura *backward chaining* é frequentemente usada nos sistemas periciais baseados em regras para permitir que se teste uma hipótese. Isto "imita" as estratégias de resolução de problemas por parte dos humanos. Por exemplo, quando está doente, o seu

médico levanta hipótese usando o conhecimento do que poderá ser a causa da sua doença. Os médicos tentam então confirmar as suas hipóteses tendo em atenção os sintomas característicos ou efectuando determinados testes. Se isto não confirmar a hipótese, o médico pensa noutra doença e testa a essa hipótese. Esta estratégia de resolução de problemas tem sido aplicada com sucesso em muitos sistemas periciais, especialmente na área do diagnóstico.

1.1.5 Limitações das Regras

Os sistemas periciais baseados em regras foram o produto com maior sucesso resultante da investigação na área da Inteligência Artificial, até há bem poucos anos. Muitos sistemas de sucesso foram implementados e estão em utilização. O seu sucesso deve-se a diversos factores:

- A estrutura de controlo assemelha-se a algumas estratégias de resolução de problemas humanas.
- A estrutura de controlo é relativamente simples, podendo ser entendida pela generalidade das pessoas, não só pelos especialistas em computadores.
- As regras fazem parte da vida quotidiana, e as pessoas estão familiarizadas com elas.
- Estão disponíveis muitas ferramentas comerciais, o que torna a tarefa de desenvolver um sistema baseado em regras bastante fácil.

Os sistemas baseados em regras têm, no entanto, limitações significativas. Na maioria das situações é extremamente difícil obter um conjunto de regras correcto. Este problema perseguiu o desenvolvimento dos sistemas periciais baseados em regras desde meados da década de 80, sendo frequentemente denominado como estrangulamento na aquisição do conhecimento (*knowledge-elicitation bottleneck*). Este estrangulamento tem diversas causas, cada uma das quais pode tornar extremamente difícil extrair conhecimento em domínios não triviais:

- Um perito ou especialista pode estar muito ocupado para dedicar o tempo necessário para que o engenheiro apreenda o seu conhecimento e o codifique.
- O perito pode ter o conhecimento e o tempo necessários, mas ser incapaz de o articular para o engenheiro o apreender.
- O engenheiro de conhecimento pode não compreender completamente o problema, devido à sua especialidade, e por isso ter dificuldade em produzir um modelo correcto.
- A representação do conhecimento adoptada pode não conseguir representar completamente o conhecimento adquirido.

Finalmente, existe um problema central aos sistemas baseados em regras, o próprio conhecimento. A abordagem baseada em regras assume que existe um corpo de conhecimento que a maioria dos especialistas na área usa e aprova. A maioria dos sistemas pioneiros é oriunda de laboratórios de investigação universitários, e em particular de cooperações entre cientista ligados à área da informática e outros cientistas (químicos, geologistas, médicos, etc.). Não surpreende então que estas pessoas acreditem que o raciocínio baseado em regras possa resolver a generalidade dos problemas. As suas disciplinas têm como fundamento regras explícitas e o raciocínio a partir de princípios fundamentais é intrínseco para eles.

No entanto, em muitas áreas quotidianas não existem modelos causais subjacentes nem princípios gerais, aceites pela generalidade dos especialistas, para produzir um modelo. Deste modo, dada a inexistência de um modelo explícito ou a extrema dificuldade em apreender o modelo, como poderão ser desenvolvidos sistemas de apoio à decisão úteis?

1.1.6 Os elefantes nunca esquecem

Alguns biólogos sugeriram que a longevidade e sobrevivência dos elefantes em ambientes austeros se deve à memória dos elefantes da manada, em especial da sua matriarca (fêmea mais velha) que lidera a manada. A memória da matriarca sobre onde existem alimentos e água e onde existem perigos ajuda a manada a sobreviver. A manada mantém uma memória colectiva dos

problemas e respectivas soluções. Por exemplo, uma manada de elefantes provavelmente não possui um modelo explícito da geologia e drenagem da área onde habitam, mas durante uma época de seca recordam-se que normalmente encontram água quando fazem buracos em determinados locais do leito seco de um rio. Deste modo, a matriarca conduz os elefantes da manada a esses locais e eles começam a escavar buracos. É quase certo que os elefantes não inferem a localização dos locais onde encontrar água raciocinando a partir de determinados princípios básicos, eles *lembram-se* onde normalmente se encontra água durante a época seca.

Assim, os elefantes podem resolver problemas deste género sem utilizarem qualquer modelo ou regras (figura 9).

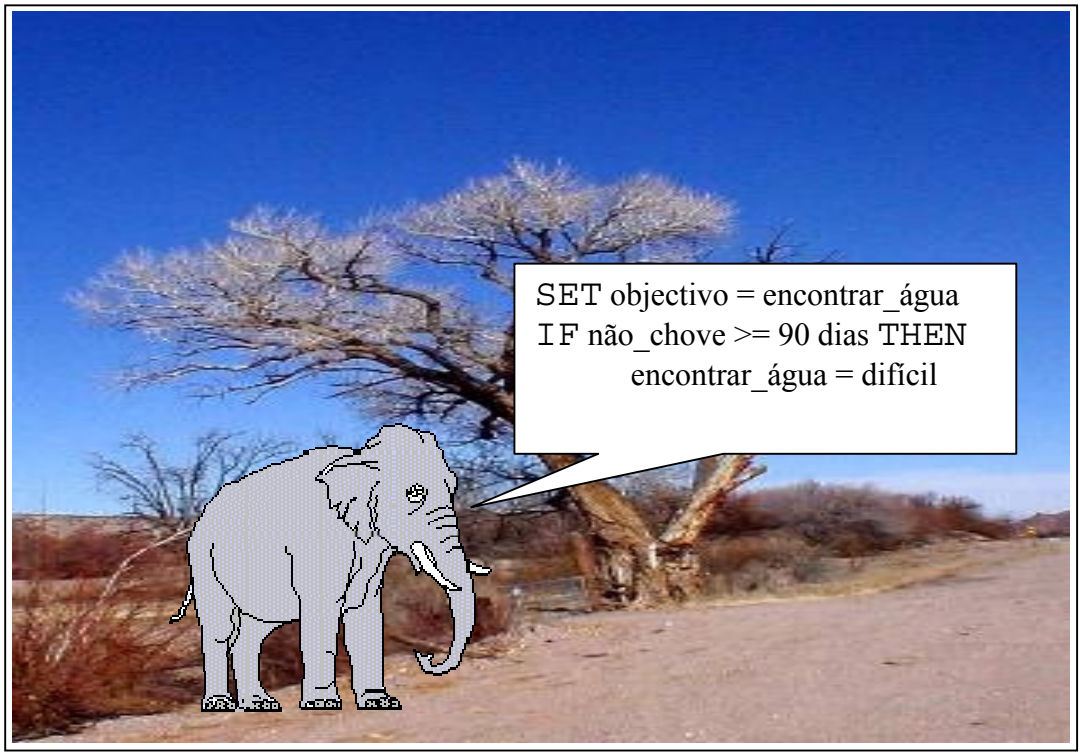

Figura 9 – Onde está a água?

Para implementar um sistema baseado em regras, temos de conhecer previamente como resolver o problema. Pode então ser implementado um sistema que resolva o problema de cada vez que este ocorrer. Mas para que resolver um problema cada vez que ele ocorre, se alguém já o resolveu anteriormente? Não seria mais simples recordar apenas a solução? Em termos de eficiência computacional, a resposta é, quase sempre, sim. Além disso, por vezes sabemos apenas como se resolveu um problema em determinadas circunstâncias, mas somos incapazes de generalizar para quaisquer circunstâncias.

Os computadores actuais, graças aos modernos meios de gravação magnéticos e ópticos, são já bastante bons a **lembrar** (i. e., armazenar e aceder). Na verdade, esta é até uma das funções mais importantes dos computadores actuais. Assim, se pudermos armazenar a descrição de problemas numa base de dados, conjuntamente com as respectivas soluções, teoricamente podemos implementar um sistema pericial, apesar de não existir qualquer modelo explícito. Tudo o que é necessário são simples relações que interliguem os problemas com as soluções, tal como indicado na figura 10.

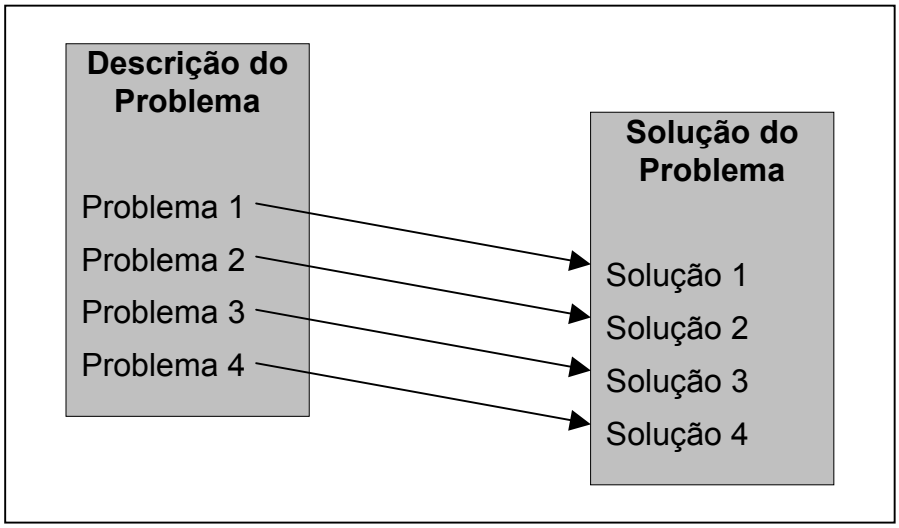

Figura 10 – Relação entre os problemas e as soluções.

1.1.7 Já tenho a resposta… Qual é a pergunta?

As bases de dados parecem a solução ideal para a tarefa de recolher soluções conhecidas de problemas:

- podem armazenar grandes quantidades de informação;
- mantêm relações entre os diferentes itens de informação;
- oferecem mecanismos eficientes de recolha.

No entanto, nem tudo são facilidades. Antes de encontrar a solução é necessário identificar e descrever o problema (qual é o problema?). É precisamente aqui que as coisas se complicam, uma vez que a tecnologia subjacente às bases de dados não resolve este problema.

Para determinar o problema correcto, e assim conseguir encontrar a solução conhecida, é necessário comparar o problema corrente com os problemas armazenados na base de dados. No entanto, os problemas do "mundo real" são normalmente bastante complexos e com muitas características específicas. Dificilmente existirá na base de dados um problema exactamente igual ao problema corrente. As bases de dados são excelentes a encontrar identificações exactas, mas não a encontrar identificações semelhantes. Isto deve-se ao facto da maioria das pesquisas em bases de dados usar uma das seguintes técnicas de pesquisa:

- Procura de palavras chave que têm de coincidir exactamente.
- Utilização de *wildcards*, como MAR* que coincide com MARÉ e MARESIA.
- Utilização de *wildcards*, como ?AR que coincide com MAR e DAR.
- Utilização de operadores lógicos, como AND, OR e NOT.

Embora estes métodos sejam úteis e eficientes na recolha de informação, têm elevada probabilidade de terem problemas em recolher informação quando ocorrem, por exemplo, erros ortográficos. Para recolher um problema coincidente com o actual é então necessário um método mais flexível que permita:

- Descrever o problema corrente tal como o vemos, idealmente utilizando a linguagem natural.
- Encontrar na memória o problema (ou conjunto de problemas) com descrição mais semelhante à do problema actual.
- Eventualmente, fazer questões com vista a confirmar a semelhança entre os problemas envolvidos.
- Devolver a solução conhecida correspondente ao problema mais semelhante.

- Eventualmente, adaptar essa solução ao problema corrente de modo a compensar as diferenças entre o problema corrente e o problema recolhido.

Estas funcionalidades caracterizam o Raciocínio Baseado em Casos (CBR - *Case-Based Reasoning*).

Sabemos que os peritos ou especialistas:

- exemplificam com casos;
- citam casos para persuadir;
- extrapolam a partir de casos;
- interpretam regras com casos;
- testam estratégias confrontando-as com casos;
- aprendem com casos;
- ensinam os novatos com casos.

Em resumo, podemos dizer que esta secção apresentou os conceitos subjacentes à representação dos dados ou informação num computador. É a estrutura dos dados que permite a extracção de informação dos dados. É possível definir regras que permitam a um computador fazer inferências a partir dos dados, e, na realidade, os sistemas periciais baseados em regras tornaram-se um sucesso para a IA. No entanto, o raciocínio baseado em regras não é certamente aquilo que as pessoas usam quando resolvem problemas. Normalmente tentamos recordar ou lembrar se alguma vez nos deparámos com um problema semelhante no passado e reutilizamos a solução que usámos na altura, se ela foi bem sucedida.

Verificou-se ainda que associar soluções com os respectivos problemas parece ajustar-se bem aos sistemas computacionais, mas a tecnologia das bases de dados não é suficiente para cumprir esta tarefa. Aquilo que é necessário é o Raciocínio Baseado em Casos. No capítulo seguinte serão introduzidos os conceitos e tecnologia do CBR - Raciocínio Baseado em Casos.

1.2 Algumas definições de CBR (Raciocínio Baseado em Casos)

```
A case-based reasoner solves new problems by
adapting solutions that were used to solve old
problems.
```

Riesbeck & Schank, 1989.

```
Case-based reasoning is both [...] the way people
uses cases to solve problems and the ways we can
make machines use them.
```

Kolodner, 1993.

```
Case-based reasoning is a recent approach to problem
solving and learning [...], to solve a new problem
by remembering a previous similar situation and by
reusing information and knowledge of that situation.
```

Aamodt & Plaza, 1994.

```
Case-based reasoning is [...] reasoning by
remembering.
```

Leake, 1996.

> In case-based reasoning, a reasoner solves a new problem by noticing its similarity to one or several previously solved problems and by adapting their know solutions, instead of working out a solution from scratch.
>
> *Mántaras & Plaza*, ?.

> Case-based reasoning solves new problems by adapting previously successful solutions to similar problems.
>
> *Watson & Marir*, 1994.

1.3 Exemplos da vida quotidiana

Como vimos atrás, o Raciocínio Baseado em Casos é uma técnica para resolver novos problemas adaptando as soluções usadas na resolução de problemas anteriores.

Existem muitos exemplos de utilização de raciocínio baseado em casos por pessoas na sua vida quotidiana:

- Um médico, após ter examinado um determinado paciente no seu consultório, recorda um doente que tratou há duas semanas. Assumindo que a recordação foi provocada pela semelhança dos sintomas (e não devido à cor do cabelo, por exemplo), o médico usa o diagnóstico e tratamento do doente anterior para determinar a doença e o tratamento a seguir pelo paciente actual.
- Um engenheiro de minas que já viveu duas situações dramáticas recorda rapidamente uma delas (ou ambas) quando o conjunto de medidas controladas corresponde ao que se verificava na altura do acidente. Em particular, ele pode recordar um erro que cometeu durante o acidente anterior e usar essa recordação para evitar comete-lo novamente.
- Um consultor financeiro que opera na difícil área de concessão de crédito usa a recordação de um caso anterior, que envolvia uma empresa com problemas semelhantes aquela que está em análise para recomendar que um empréstimo seja recusado.

Tal como os exemplos anteriores atestam, o raciocínio através da reutilização de casos anteriores é uma forma poderosa e de aplicação frequente na resolução de problemas por parte dos humanos e o CBR evolui, em parte, a partir da investigação da memória humana.

1.4 Evolução do Paradigma do CBR

O Raciocínio Baseado em Casos difere significativamente dos outros tipos de raciocínio e técnicas de resolução de problemas, sendo, nalguns casos, uma consequência directa destas técnicas. Abaixo descrevem-se dois factores importantes que contribuíram para a evolução do paradigma do CBR.

1.4.1 Teorias da Memória

O CBR desenvolveu-se, em parte, a partir da investigação da memória humana, especialmente da teoria da **Memória Dinâmica** desenvolvida por Schank (1982). Esta teoria introduziu as estruturas de memória conhecidas por *Memory Organization Packet* (MOP) e consideradas como as percursoras dos **Casos**. Os MOPs são usados para armazenar não só informação genérica acerca do mundo, mas também experiências específicas. A teoria da memória dinâmica fornece a estrutura para descrever como estas experiências isoladas podem ser armazenadas na memória, como podem ser combinadas e resumidas e como podem recuperadas e usadas quando for necessário. Especificar o conteúdo e processo de memória deste modo fornece os fundamentos para alguns dos objectivos do CBR.

1.4.2 Problemas com os Sistemas Baseados em Regras

Uma outra força na evolução do CBR foi a insatisfação com o **Raciocínio Baseado em Regras** (sistemas periciais), que era a principal técnica de resolução de problemas da época. Três dos problemas com os Sistemas Baseados em Regras que levaram à procura de um paradigma para resolução de problemas são:

- **Aquisição de Conhecimento** Na resolução de problemas baseada em regras a aquisição de conhecimento para codificar o sistema era uma tarefa bastante difícil. Os engenheiros de conhecimento consideravam demasiado difícil descobrir as centenas de regras que os "peritos" usam para resolver os problemas, principalmente porque os peritos frequentemente têm dificuldade em exprimir a sua habilidade na resolução dos problemas na forma de regras IF-THEN. Este problema tornou-se conhecido como o **Congestionamento na Aquisição de Conhecimento**. Além disso, tornou-se pouco claro se os peritos usam na realidade regras. Com frequência os peritos dizem que é a sua experiência que faz deles peritos. Adicionalmente, enquanto as regras se afiguram como uma representação compacta para reunir informação, normalmente têm muitas excepções, tornando a aquisição de conhecimento complicada.
- **Ausência de Memória** A segunda maior crítica aos sistemas baseados em regras é que a maioria não tem qualquer tipo de memória, i.e., o sistema não recorda anteriores encontros com o mesmo problema, e terá de o resolver novamente a partir do início. É claro que isto é extremamente ineficiente, mas mesmo pondo de parte a eficiência, uma característica ainda mais importante é que um sistema sem memória não será capaz de recordar os erros cometidos no passado. Sem esta capacidade, o sistema está claramente condenado a repetir esses erros novamente. Este tipo de "estupidez" não é tolerado nos humanos que resolvem problemas, havendo por isso um grande descontentamento com a incapacidade dos sistemas baseados em regras aprenderem com os erros.
- **Robustez** A terceira grande crítica aos sistemas baseados em regras é serem "frágeis". Como todo o seu conhecimento é recordado em termos de regras, se um problema não corresponder a nenhuma das regras, o sistema não será capaz de o resolver. Por outras palavras, os sistemas baseados em regras têm muito pouca ou nenhuma capacidade para operar para além da sua base de regras; não são capazes de lidar com situações novas.

1.5 Vantagens da Abordagem CBR

O Raciocínio Baseado em Casos tem inúmeras vantagens como teoria e metodologia para o raciocínio e a resolução de problemas. Muitas destas vantagens são a resposta directa aos factores, expostos atrás, que levaram ao desenvolvimento do paradigma CBR.

1.5.1 Credibilidade Psicológica

Dado que se desenvolveu com base na memória humana, não é surpreendente que um dos factores que torna o CBR interessante como modelo de raciocínio e de resolução de problemas é ser baseado na forma como os humanos raciocinam e resolvem os problemas. O raciocínio a partir de casos anteriores, contrariamente a um conjunto de regras (enorme), foi pensado para ser um modelo psicológico mais plausível do modo como as pessoas pensam.

Existe uma forte convicção de que as pessoas usam Raciocínio Baseado em Casos no seu raciocínio quotidiano.

1.5.2 Casos *vs.* Regras

Como foi discutido atrás, as idéias do CBR desenvolveram-se, em parte, devido aos problemas do Raciocínio Baseado em Regras. O Raciocínio Baseado em Casos é vantajoso face ao Raciocínio Baseado em Regras nos seguintes aspectos:

- **Aquisição de Conhecimento** Casos são mais fáceis de memorizar do que regras abstractas. Normalmente é mais fácil para os peritos recordar e articular exemplos específicos dos problemas com que se depararam e das respectivas soluções do que descrever as suas técnicas de resolução dos problemas em termos de um conjunto de regras. De facto, investigadores que conhecem como raciocinar usando casos acharam mais fácil implementar sistemas baseados em casos do que os sistemas tradicionais baseados em regras.
- **Aprender com a experiência** Os sistemas baseados em casos são, por definição, implementados numa memória de casos anteriores. De cada vez que o sistema resolve um problema, esse problema e a sua solução são guardados na memória como um caso. Deste modo os sistemas CBR podem facilmente aprender com a experiência, não têm de desperdiçar tempo resolvendo um problema que é igual a um já anteriormente tratado, nem irão repetir os erros cometidos quando resolveram o problema pela primeira vez. Enquanto os sistemas que resolvem os problemas a partir de regras desperdiçam imenso tempo resolvendo os seus problemas desde o início, os sistemas baseados em casos reusam os casos anteriores, sendo algumas ordens de grandeza mais rápidos.
- **Adaptabilidade** Enquanto os sistemas baseados em regras são "frágeis", os sistemas CBR são bastante robustos quando se deparam com novas situações. Esta robustez provem das técnicas de adaptação de casos. Aquando da tentativa de resolução de um novo caso, os sistemas CBR conseguem pesquisar na sua memória casos com características semelhante e adaptar a solução desses problemas de modo a serem úteis na resolução do problema actual.

1.6 O Mecanismo Natural de Aprendizagem

A aprendizagem a partir da experiência é, tal como foi referido atrás, uma das vantagens do CBR sobre os sistemas baseados em regras. O paradigma do CBR fornece um mecanismo natural de aprendizagem. Um sistema CBR "aprende", basicamente, de dois modos. Primeiro, o sistema pode tornar-se mais eficiente recordando soluções anteriores e adaptando-as em vez de ter de desenvolver a solução desde o início de cada vez que se depara com o problema. Se um caso foi adaptado de uma forma original, se foi resolvido usando um novo método ou se foi resolvido através da combinação das soluções de diversos casos, então quando posteriormente for recordado não é necessário repetir os seus passos para resolver o novo problema. Em segundo lugar, um sistema CBR torna-se mais competente ao longo do tempo, derivando melhores soluções do que quando tinha menos experiência. Uma das grandes vantagens do CBR é a sua capacidade de ajudar a prevenir quem raciocina e assim evitar erros anteriormente cometidos.

A aprendizagem baseada em casos como um paradigma de aprendizagem oferece muitas vantagens, nomeadamente:

- **Aquisição de conhecimento facilitada** Como o conhecimento é guardado na forma de casos, a quantidade de conhecimento necessário para o arranque do sistema é menor. Por outras palavras, o processo de aprendizagem pode iniciar-se com muito menos "dados" do que os requeridos normalmente pelos sistemas baseados em regras. Depurar a base de conhecimento é também mais fácil, pois tendem a existir menos interacções entre os casos do que entre as regras. Finalmente, existem já vários domínios com informação codificada no formato de casos, por exemplo, direito, matemática e projecto.
- **Aumento de desempenho** Como os sistemas CBR guardam os problemas enfrentados anteriormente, podem reusar as soluções anteriores em vez de terem de obter novas soluções começando do nada. Além disso, como são relembrados os erros anteriores, um sistema CBR para resolução de problemas pode evitar cometer os mesmos erros novamente. Estes factores contribuem para aumentar o desempenho (eficiência) dos sistemas CBR.

- **Aprender é fácil** Em geral, a aprendizagem num sistema CBR não necessita de nenhum modelo complexo do domínio nem do conhecimento detalhado do domínio. É claro que o acrescento de qualquer destes itens pode aumentar o desempenho e a capacidade de um sistema baseado em casos.
- **Os casos podem servir como esclarecimentos** Uma característica normalmente desejável nos sistemas de resolução de problemas é a capacidade de fornecer uma explicação ou esclarecimento para a solução obtida. Nos sistemas CBR estes esclarecimentos são simplesmente os casos que foram usados, sendo por isso fáceis de gerar. De facto, como o CBR resolve os problemas de forma semelhante aos humanos, uma explicação baseada num caso concreto anterior pode ser mais satisfatória do que explicações construídas a partir de uma série de regras, o método de explicação usado nos sistemas baseados em regras.
- **Escalável** O problema comum quando se "amplia" um sistema, a procura ou pesquisa em grande escala, é, de certa forma, evitada pelo CBR através do uso de índices. Com o desenvolvimento de algoritmos paralelos para recuperação, parece que sistemas CBR pesados podem ser capazes de se aproximar, ou mesmo alcançar um desempenho de tempo real.

2 O Processo de Raciocínio Baseado em Casos

Seguidamente serão descritos os métodos e sistemas de CBR usando uma estrutura constituída por três partes:

- Etapas do Raciocínio Baseado em Casos.
- Modelo processual para o ciclo CBR.
- Estrutura de tarefas para o Raciocínio Baseado em Casos.

Na primeira descrevem-se os principais passos do raciocínio baseado em casos ignorando qualquer formalismo subjacente. A segunda é um modelo dinâmico que identifica os principais subprocessos do ciclo CBR, as suas interdependências e resultados. A terceira é uma abordagem orientada para as tarefas, em que são descritas a decomposição das tarefas e os métodos de resolução de problemas associados. Esta estrutura será depois usada para identificar e discutir as principais áreas problemáticas do CBR, assim como as formas de lidar com eles.

2.1 O Algoritmo Básico do CBR

O ciclo básico do Raciocínio Baseado em Casos é "introduzir um problema, recolher as soluções anteriores relevantes, adaptá-las ao problema actual, guardar o novo caso juntamente com a solução". Descrevem-se a seguir os vários passos de um algoritmo de CBR típico. Embora diferentes sistemas CBR possam realçar diferentes partes deste ciclo, todos os sistemas visam de alguma forma os seguintes passos:

1. **Aceitação e Análise** Após a introdução de um novo problema, o primeiro passo de processamento é conhecido por **fase de análise**. Neste passo a entrada é analisada para extrair características a usar na recolha de casos com características semelhantes. Estas características, que têm uma grande importância no CBR, são chamadas **índices**. Assim, a principal tarefa deste passo é a **extracção de índices**. A extracção de índices é um problema bastante complexo, que será discutido num capítulo mais à frente.
2. **Recolha de Casos da Memória** Os índices calculados no passo 1 são usados para recolher casos da memória. O objectivo é não recolher um qualquer conjunto de casos. Aqueles casos que podem ser usados no raciocínio a efectuar nos passos seguintes e que têm o potencial de fazer predições úteis acerca do problema actual são o tipo de casos que devem ser recolhidos. Existem diversas técnicas para recolha os casos, incluindo modelos iterativos, paralelos e "*constraint-satisfaction*".
3. **Selecção do Caso(s) mais Relevante** O conjunto (potencialmente grande) de casos pertinentes obtido no passo 2 necessita frequentemente de ser restringido de modo a obter apenas alguns casos "mais relevantes". Estes casos serão os merecedores de processamento intensivo com vista à formação da base da nova solução. O problema neste passo é avaliar quão relevante cada caso é na realidade. Técnicas típicas para isto incluem uma diversidade de planos de hierarquia e métricas de semelhança ou parecença, sobreposição de características proeminentes e importância das características partilhadas. Os problemas da avaliação de semelhanças serão estudados num capítulo mais à frente.
4. **Construção da Solução** Este passo usa os casos seleccionados no passo 3 para criar uma solução ou interpretação (dependendo do objectivo) para o caso introduzido. Juntamente com a solução, muitos sistemas CBR construirão também as justificações ou argumentos de suporte para a solução nesta altura. Os casos recolhidos são usados na construção da solução em pelo menos duas formas. Primeiro, a solução actual é construída adaptando as soluções dos casos anteriores de modo a que sejam relevantes para o caso actual. Segundo,

os casos recolhidos podem ser usados para advertir de potenciais dificuldades na construção da solução, permitindo ao sistema antecipar e por isso evitar cometer o mesmo tipo de erros e enganos dos problemas anteriores.

5. **Avaliação da Solução** Após ter definido uma solução potencial, esta é sujeita a testes, avaliação e crítica. O objectivo a atingir por este passo é avaliar a utilidade, pontos fortes e fraquezas da solução proposta. Existem diversos métodos para executar esta tarefa, incluindo testar a solução com contra exemplos (reais ou hipotéticos), usar a solução como um índice para a memória para verificar se existirá algum exemplo desta solução que se saiba ter falhado em circunstâncias semelhantes ou simulando os resultados da solução proposta. Utilizar mecanismos de teste interno é especialmente crítico em domínios em que o custo de uma solução incorrecta ou ineficiente é elevado, por exemplo, no diagnóstico médico.
6. **Execução da Solução e Análise dos Resultados** Neste passo a solução é posta à prova no mundo real e o sistema obtém *feedback* do que acontece. Este *feedback* é sujeito a uma cuidadosa análise para verificar se os resultados são os esperados. Este processo de obter e analisar o *feedback* é crucial se se pretende que o sistema CBR aprenda a partir dos seus erros e evite repeti-los. Se qualquer coisa inesperada ocorrer, o sistema tenta explicar o evento anómalo. O problema de tentar decidir quais as partes da solução que provocaram problemas, conhecer completamente o domínio do *machine-learning* como o problema de atribuição de créditos/responsabilidade, aparece aqui. Uma técnica que pode ser usada para evitar parcialmente este problema é relembrar os desaires semelhantes enfrentados no passado e usar os esclarecimentos desses desaires na explicação da falha actual. Este processo é simplesmente outra versão do problema do Raciocínio Baseado em Casos e pode normalmente ser feito através de uma chamada recursiva ao sistema CBR.
7. **Actualização da Memória** Após os resultados do teste da solução no mundo real serem analisados, o próximo passo é actualizar a memória armazenando o novo caso. O novo caso é constituído não apenas pela solução a que se chegou, mas também pelas justificações e argumentos de suporte construídos no passo 4. O aspecto mais importante deste passo é onde colocar o caso na memória ou, usando a terminologia do CBR, como indexá-lo. Uma técnica comum é indexar um caso pelos problemas ou falhas que foram encontrados (nos passos 5 e 6) de modo a que os mesmos erros possam ser evitados quando o sistema se deparar com uma situação semelhante no futuro. O êxito ou fracasso de um sistema CBR depende grandemente deste passo. Boas estratégias de índices farão com que os casos pertinentes sejam recordados quando puderem ser úteis, resultando num bom desempenho do sistema.

2.2 O Ciclo CBR

De um ponto de vista de alto nível, o ciclo CBR genérico pode ser descrito pelos quatro processos seguintes[1]:

- **Recolha** do caso mais semelhante (ou casos).
- **Reutilização** da informação e conhecimento desse caso para resolver o problema.
- **Revisão** da solução proposta.
- **Retenção** das partes desta experiência que se suponha virem a ser úteis na resolução de problemas futuros.

Um novo problema é resolvido *recolhendo* um ou mais casos anteriormente enfrentados, *reutilizando* o caso de alguma forma, *revendo* a solução baseada num caso anterior e *retendo* a nova experiência, incluindo-a na base de conhecimento (base de casos). Cada um destes quatro

[1] Conhecidos pelos quatro ***RE's***

processos envolve um determinado número de tarefas que serão descritas no modelo de tarefas apresentado na secção seguinte. Na figura 11 é apresentado o ciclo CBR.

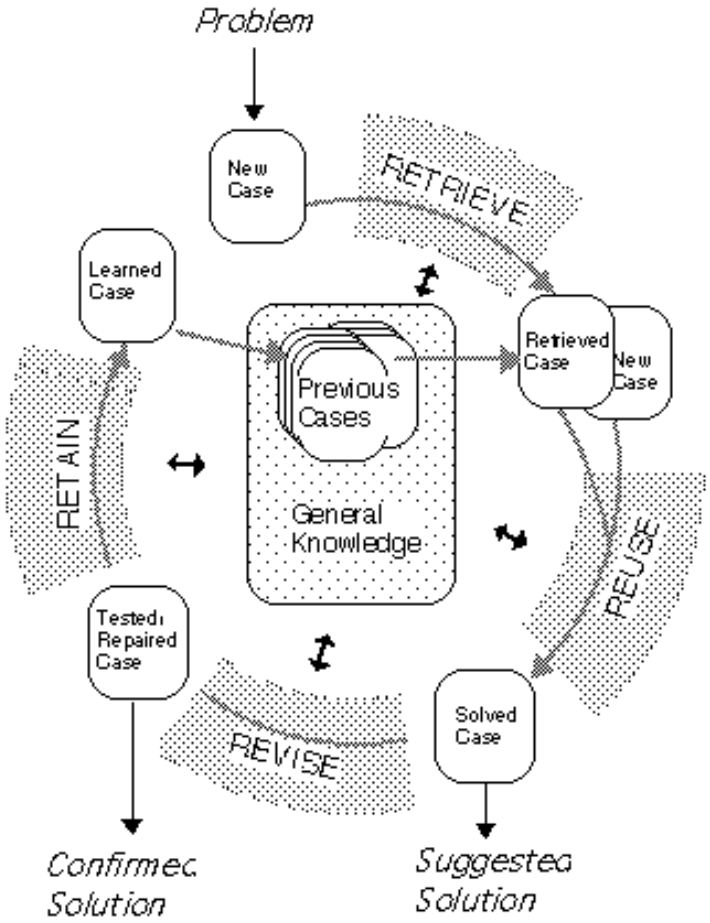

Figura 11 – O Ciclo CBR

Uma descrição inicial de um problema (topo da figura) define um novo caso. Este novo caso é usado para recolher um caso da memória de casos anteriores. O caso recolhido é combinado com o novo caso (reutilização) resultando num caso resolvido, i.e., numa solução proposta para o problema. Através do processo de revisão esta solução é testada para verificar o seu sucesso, seja pela sua aplicação no ambiente real seja pela avaliação de um perito, e reparada no caso de falhar. Durante a retenção, a componente útil desta experiência é guardada para futura reutilização, e a base de casos é actualizada através da inclusão do novo caso ou da modificação de alguns dos casos existentes.

De notar que o ciclo apresentado raramente ocorre sem intervenção humana. Por exemplo, muitas ferramentas de CBR funcionam principalmente como sistemas de recolha e reutilização de casos. A revisão (ou adaptação) é frequentemente da responsabilidade do administrador da base de casos. Este estímulo à colaboração humana no apoio à decisão não deve, no entanto, ser visto como uma fraqueza dos sistemas CBR.

Tal como é indicado na figura, o *conhecimento geral* é uma parte importante deste ciclo, apoiando o ciclo CBR. Este apoio pode variar de muito fraco (eventualmente nulo) a muito forte, dependendo do método de CBR. Por conhecimento geral entende-se o conhecimento base de um determinado domínio, contrariamente ao conhecimento específico contido nos casos. Por exemplo, no diagnóstico médico através da recolha e reutilização do caso de um paciente anterior, o modelo anatómico conjuntamente com as relações causais entre as patologia pode constituir o conhecimento geral usado pelo sistema CBR. Um conjunto de regras pode ter o mesmo papel.

2.3 A Hierarquia de Tarefas no CBR

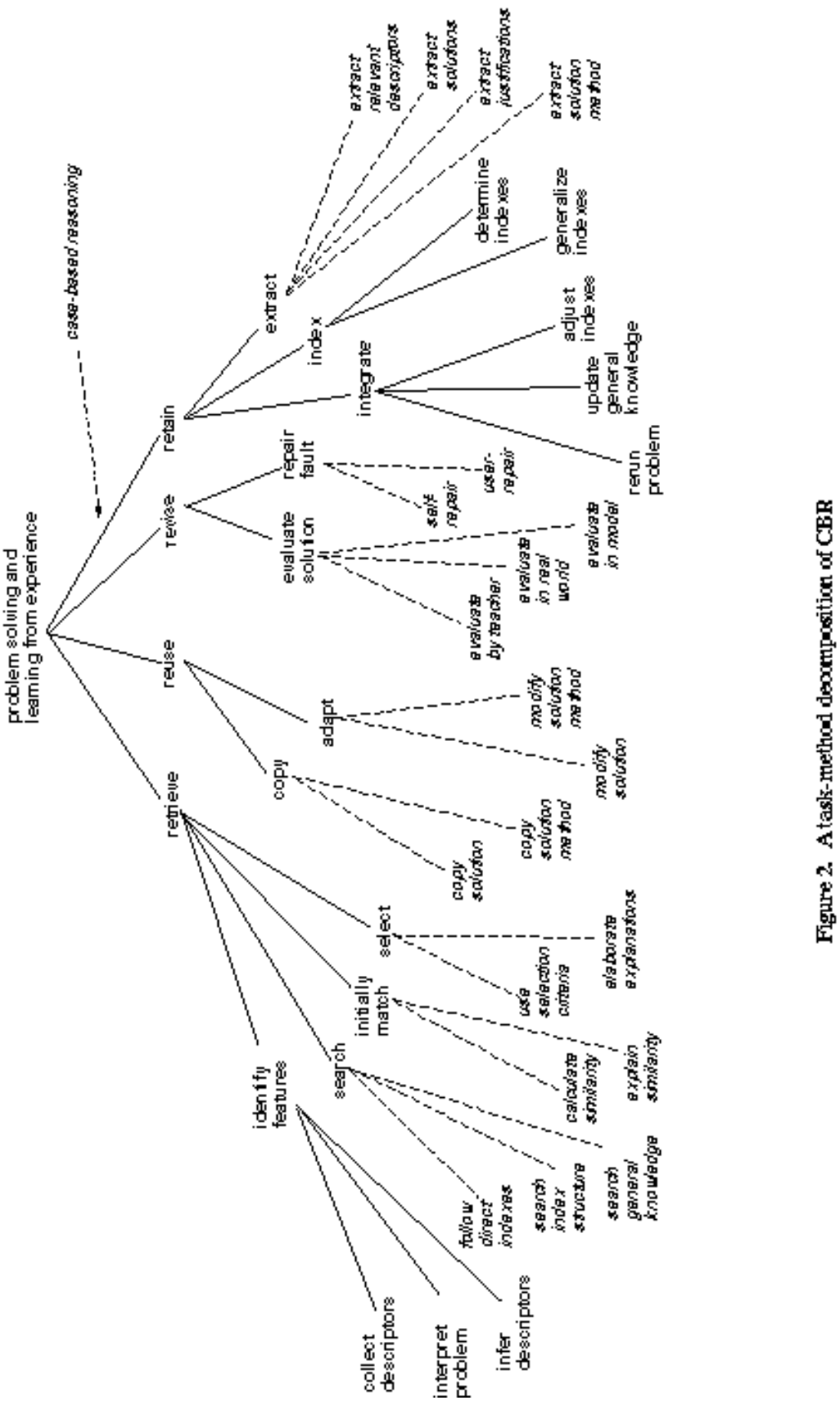

Figure 2. A task-method decomposition of CBR

Figura 12 - Hierarquia de Tarefas no CBR

O processo descrito atrás foi escolhido de modo a realçar o CBR como um ciclo de fases sequenciais. Para fazer uma decomposição adicional e descrever os quatro processos principais, vamos usar uma abordagem orientada para as tarefas em que cada fase, ou subprocesso, é visto como uma tarefa que o sistema CBR tem de efectuar. Enquanto uma visão orientada para os processos possibilita uma visão global ou externa do que se passa, uma visão orientada para as tarefas é mais apropriada para descrever os mecanismos de funcionamento detalhados do ponto de vista do próprio sistema CBR.

A estrutura do modelo de tarefas que será utilizado a seguir é a apresentada na figura 12.

As tarefas são indicadas por nodos com nomes a negrito e os métodos estão escritos em itálico. As ligações entre os nodos de tarefas (linhas a cheio) são decomposições das tarefas, i. e., partes das relações. A tarefa de mais alto nível é a *resolução de problemas* e a *aprendizagem a partir da experiência*. O método para realizar esta tarefa é o *Raciocínio Baseado em Casos* (indicado por uma linha ponteada). Isto provoca a divisão do nível mais acima nas quatro tarefas principais do CBR, que correspondem aos quatro processos da figura 11: Recolha, Reutilização, Revisão e Retenção. Todas as quatro tarefas são necessárias para executar a tarefa de alto nível. A tarefa **Recolha** é, por sua vez, dividida de forma semelhante (por um método de recolha) nas tarefas de *identificação* (dos descritores relevantes), *pesquisa* (para encontrar um conjunto de casos anteriores), *comparação inicial* (dos descritores relevantes com os casos anteriores) e *selecção* (do caso com maior semelhança).

Todas as divisões de tarefas apresentadas na figura são completas, i. e., o conjunto de subtarefas de uma determinada tarefa projectadas de modo a serem suficientes para completar a tarefa. A figura não mostra qualquer estrutura de controlo sobre as subtarefas, embora uma sequência aproximada das tarefas seja indicada colocando as subtarefas iniciais mais acima na figura do que aquelas que são realizadas posteriormente (em cada um dos grupos de subtarefas). As relações entre tarefas e métodos (linhas ponteadas) identificam métodos alternativos que podem ser aplicados para resolver a tarefa. Um método especifica um algoritmo que identifica e controla a execução de uma subtarefa e acede e utiliza o conhecimento para o fazer. Os métodos apresentados são na realidade classes de métodos, de onde se escolher um ou mais métodos específicos. Os conjuntos de métodos apresentados são incompletos, i. e., um dos métodos indicados pode ser suficiente para resolver a tarefa, diversos métodos podem ser combinados ou podem existir outros métodos que conseguem resolver a tarefa.

2.3.1 Recolha de Casos

A tarefa de recolha de casos começa com uma descrição (parcial) do problema e termina quando o caso anterior mais semelhante for encontrado. As subtarefas desta fase são, como já vimos, a *identificação das características*, *correspondência inicial* e *selecção*, executadas por esta ordem. A tarefa de identificação surge com um conjunto de descritores relevantes do problema, o objectivo da tarefa de correspondência é devolver um conjunto de casos que sejam suficientemente semelhantes ao novo caso (dado um limiar de semelhança) e a tarefa de selecção opera sobre este conjunto de casos e escolhe a melhor correspondência.

Enquanto algumas abordagens baseadas em casos recolhem os casos com base em semelhanças superficiais ou sintácticas entre os descritores do problema, outras tentam recolher casos com base em características mais profundas, as semelhanças semânticas. De modo a comparar casos com base em semelhanças semânticas e na importância relativa das características, é necessário um forte conhecimento geral do domínio para explicar porque é que dois casos são semelhantes e quão intensa é essa semelhança. A avaliação da semelhança sintáctica (também chamada de *conhecimento fraco*) é vantajosa em domínios onde o conhecimento geral do domínio é muito difícil ou impossível de obter. Por outro lado, as abordagens semânticas (também chamada de *conhecimento intenso*) é capaz de usar o contexto da descrição do problema na sua comparação, em domínios em que o conhecimento geral do domínio está disponível.

Uma questão que deve ser respondida aquando da decisão de uma estratégia de recolha é o objectivo da tarefa de recolha. Se o objectivo é recolher um caso que se pretende adaptar antes de o reutilizar isto pode ser tomado em conta no método de recolha.

2.3.1.1 Identificação das características

A identificação de um problema pode envolver simplesmente conhecer os seus descritores, mas na maioria das vezes é usada uma abordagem mais elaborada, em que se tenta "entender" o problema no seu contexto. Os descritores desconhecidos podem ser ignorados ou pode pedir-se ao utilizador que os explicite. O entendimento de um problema envolve filtrar os descritores confusos para inferir outras características relevantes do problema, verificar se os valores das características fazem sentido no contexto, etc. Outros descritores além dos fornecidos na entrada podem ser inferidos usando um modelo de conhecimento geral ou recolhendo uma descrição de problema semelhante da base de casos e usando características desse caso como características esperadas.

2.3.1.2 Correspondência inicial

A tarefa de encontrar uma boa correspondência é, tipicamente, dividida em duas subtarefas. Um processo de correspondência inicial que recolhe um conjunto de candidatos aceitáveis e um processo mais elaborado para seleccionar o melhor caso entre eles. O último corresponde à tarefa de selecção, descrita mais à frente. A determinação do conjunto de casos semelhantes é feita usando os descritores do problema (características de entrada) como índices para a base de casos. Existem, basicamente, três formas de recolher o caso ou conjunto de casos: Seguir os ponteiros directos dos índices, pesquisar a estrutura de índices ou pesquisar o modelo de conhecimento geral. Existem sistemas que combinam mais do que um dos métodos.

Os casos podem ser recolhidos usando exclusivamente as características de entrada ou a partir de características inferidas a partir da entrada. É claro que os casos em que todas as características coincidem com o caso actual são bons candidatos para a recolha, mas (dependendo da estratégia) os casos em que apenas uma fracção das características coincidem com as características do problema (introduzidas na entrada ou inferidas) podem também ser recolhidos. Normalmente os sistemas executam alguns testes para verificar a relevância dos casos recolhidos, especialmente se os casos foram recolhidos apenas com base num subconjuntos de características. Por exemplo, um teste de relevância simples consiste em verificar se a solução recolhida se adapta ao tipo de solução esperada para o novo problema. Verifica-se assim que é necessária uma forma de avaliar o grau de semelhança, tendo sido propostas diversas "métricas de semelhança" baseadas nas semelhanças superficiais do problema e nas características dos casos.

A avaliação da semelhança pode também ser feita usando mais intensivamente o conhecimento, por exemplo, tentando perceber mais profundamente o problema, usando os objectivos, restrições, etc. Uma alternativa é pesar os descritores do problema de acordo com a sua importância relativa na caracterização do problema, durante a fase de aprendizagem (retenção). No PROTOS, por exemplo, cada característica dos casos guardados tem associada um grau de importância para a solução do caso. Um mecanismo semelhante é utilizado pelo CREEK, que guarda a capacidade preditiva da característica relativamente ao conjunto de casos (valor discriminativo) e exigência da característica, i. e., a influência que a falta da característica tem na solução do caso.

2.3.1.3 Selecção

A partir do conjunto de casos semelhantes é escolhido o mais relevante. Isto pode ser feito durante a fase de correspondência inicial, mas normalmente esta fase devolve um conjunto de casos. O caso mais semelhante é normalmente determinado através da avaliação mais rigorosa do grau de correspondência inicial. Isto é feito tentando gerar explicações para justificar as características diferentes, com base no conhecimento da hierarquia semântica. Se se verificar que

a correspondência não é suficientemente forte, é feita uma tentativa de encontrar uma melhor correspondência seguindo ligações diferentes. Esta subtarefa é normalmente mais elaborada que a recolha, apesar de não existir distinção entre a recolha e esta correspondência mais elaborada em alguns sistemas. O processo de selecção normalmente gera consequências e expectativas para cada caso recolhido e tenta avaliar as consequências e explicar as expectativas. Isto pode ser feito usando o próprio modelo de conhecimento geral do sistema ou perguntando ao utilizador confirmações e informação adicional. Os casos são eventualmente classificados de acordo com alguma métrica ou critério de classificação. Os métodos de selecção com uso de conhecimento geram normalmente explicações que suportem este processo de classificação e o caso que tiver a melhor explicação de ser semelhante ao novo problema é escolhido.

Outras propriedades dos casos que são consideradas em alguns sistemas CBR são: a importância relativa e a eficácia discriminatória das características, a prototipicalidade de um caso dentro da sua classe e as diferenças nas ligações para os casos relacionados.

2.3.2 Reutilização de casos

A reutilização da solução do caso recolhido no contexto do novo caso centra-se basicamente em dois aspectos:

a) as diferenças entre o caso anterior e o caso actual e
b) qual a parte do caso recolhido que pode ser transferida para o novo caso.

2.3.2.1 Cópia

Nas tarefas de classificação simples as diferenças são desprezadas (são consideradas não relevantes enquanto as semelhanças são relevantes) e o tipo da solução recolhida é transferido para o novo caso como o seu tipo de solução. Esta é uma forma trivial de reutilização. No entanto, há sistemas que consideram as diferenças em *a*) pelo que a parte reusada *b*) não pode ser directamente transferida para o novo caso, requerendo um processo de adaptação que tenha em conta as diferenças.

2.3.2.2 Adaptação

Existem duas formas principais de reutilizar casos anteriores:

- reusar a solução do caso anterior - método de transformação e
- reusar o método que permitiu derivar a solução do caso anterior - método derivacional.

No método de transformação a solução do caso anterior não é uma solução directa para o novo caso, mas existe algum conhecimento da forma de transformação que aplicada à solução anterior a transforma na solução do novo caso. Uma forma de organizar esta transformação é indexá-la de acordo com as diferenças detectadas entre o caso recolhido e o caso actual. Um exemplo é o CASEY, em que uma explicação é formada a partir das explicações do caso anterior usando regras. O método de transformação não tem em conta como o problema é resolvido, mas sim a equivalência das soluções e isto implica um bom modelo dependente do domínio para a forma de transformação. As diferentes variantes deste método serão indicados e explicados mais à frente.

A reutilização derivacional considera como o problema foi solucionado no caso recolhido. O caso recolhido contém informação acerca do método usado para resolver o respectivo problema, incluindo a justificação dos operadores usados, objectivos considerados, alternativas geradas, falhas, etc. A reutilização derivacional adapta então o método recolhido ao novo caso e "repete" o anterior plano no novo contexto.

2.3.3 Revisão de casos

Quando a solução gerada pela fase de reutilização não está correcta, surge a oportunidade de aprender com as falhas. Esta fase é chamada revisão do caso e consiste em duas subtarefas:

1) avaliar a solução gerada pela reutilização. Se foi um sucesso aprende com o sucesso (retenção do caso).
2) senão, reparar o caso usando o conhecimento específico do domínio.

2.3.3.1 Avaliação da solução

A fase de avaliação recebe o resultado da aplicação da solução no ambiente real. Este passo é normalmente exterior ao sistema CBR, uma vez que, pelo menos num sistema em operação normal, envolve a aplicação da solução proposta ao problema real. Os resultados desta aplicação da solução podem levar algum tempo a aparecer, dependendo do tipo de aplicação. Num sistema de apoio à decisão médica o sucesso ou insucesso de um tratamento pode levar de poucas horas a alguns meses. O caso pode ainda ser retido e ficar disponível na base de casos neste período intermédio, mas tem de estar assinalado como não avaliado. A solução pode também ser aplicada a um programa de simulação que seja capaz de gerar a solução correcta. É o que acontece, por exemplo, no CHEF, em que a solução (uma receita culinária) é aplicada a um modelo interno que se assume suficientemente rigoroso para dar o *feedback* necessário à reparação da solução.

2.3.3.2 Reparação das falhas

A reparação de casos implica detectar os erros da solução corrente e recolher ou gerar explicações para eles. O melhor exemplo é o sistema CHEF em que o conhecimento causal é usado para gerar uma explicação para o facto de certos objectivos da solução não serem atingidos. O CHEF aprende as situações gerais que irão causar as falhas usando uma técnica de aprendizagem baseada em explicações. Isto é incluído numa memória de falhas que é usada na fase de reutilização para prever possíveis deficiências do plano. Esta forma de aprendizagem desloca a detecção dos erros para a fase de elaboração do plano onde os erros podem ser previstos, tratados e evitados. Uma segunda tarefa da fase de revisão é a reparação da solução. Esta tarefa usa as explicações das falhas para modificar a solução de modo a que essas falhas não ocorram. Por exemplo, o plano falhado no sistema CHEF é modificado por um módulo de reparação que adiciona passos ao plano que irão assegurar que os erros não ocorrem. O módulo de reparação possui conhecimento causal geral e conhecimento do domínio sobre como eliminar ou compensar causas de erros no domínio. O plano revisto pode ser retido directamente (se a fase de revisão assegurar a sua exactidão) ou pode ser avaliado e reparado novamente.

2.3.4 Retenção de casos - Aprendizagem

Este é o processo de incorporar o que for útil reter do novo problema no conhecimento existente. A aprendizagem a partir dos sucessos ou falhas da solução proposta é provocada pelo resultado da avaliação e possíveis reparações. Isto envolve seleccionar que informação do caso se deve reter, sob que forma a reter, como indexar o caso para futura recolha e como integrar o novo caso na estrutura da base de casos.

2.3.4.1 Extracção

No CBR a base de casos é actualizada independentemente de como o problema foi resolvido. Se foi resolvido usando um caso anterior, pode ser construído um novo caso ou o caso recolhido pode ser generalizado para incluir também o caso actual. Se o problema foi resolvido por outros métodos, inclusive questões ao utilizador, um novo caso tem que ser formado. Em qualquer das situações, é necessário decidir o que usar como fonte de aprendizagem. Os descritores relevantes do problema e as soluções do problema são candidatos óbvios. Mas uma explicação ou outra forma de justificação do porquê de uma solução ser solução do problema pode também ser incluída num novo caso. No CASEY e no CREEK, por exemplo, as explicações são incluídas nos casos retidos e reutilizadas em posteriores modificações da solução.

As falhas, i. e., a informação da tarefa de revisão, podem também ser extraídos e retidos, quer como casos de falha separados quer como casos completos. Quando é encontrada uma falha

o sistema pode relembrar falhas semelhantes anteriores e usar o caso de falha para melhorar o seu entendimento e corrigir a falha actual.

2.3.4.2 Indexação

O problema da indexação é um problema central do CBR. Ele equivale a decidir qual o tipo de índices a usar nas futuras recolhas e como estruturar o espaço de pesquisa dos índices. Os índices directos, já mencionados atrás, passam por cima do último passo, mas há ainda o problema de identificar o tipo de índices a usar. Este é na realidade um problema de aquisição de conhecimento. Uma solução trivial é usar todas as características do problema introduzido como índices.

No CASEY é usado um método de indexação de duas fases. Índices primários são estados no modelo de problemas do coração que são parte da explicação do caso. Quando é introduzido um novo problema, as falhas são transmitidas para o modelo de problemas do coração e os estados que explicam as características são usadas como índices para a base de casos. As próprias características observadas são usadas apenas como características secundárias.

2.3.4.3 Integração

Esta é a fase final da actualização da base de casos com novos conhecimentos de casos. Se nenhum novo caso ou conjunto de índices foi gerado, esta é a tarefa da fase de retenção. Através da alteração da estrutura de índices dos casos existentes, o sistema CBR aprende a ser melhor avaliador das semelhanças. A afinação dos índices existentes é uma parte importante da aprendizagem CBR. Os pesos ou importâncias dos índices num caso ou solução particular são ajustados devido ao sucesso ou falha da utilização do caso na resolução do problema introduzido. Para as características que foram consideradas relevantes na recolha de um caso de sucesso, a associação com o caso é fortalecida, enquanto é enfraquecida para as características que levam à recolha de casos de insucesso. Deste modo a estrutura de índices tem o papel de afinar e adaptar a base de casos à utilização. O PATDEX tem uma forma especial para aprender a importância das características: Uma matriz de importâncias liga as características possíveis aos diagnósticos para que são relevantes e atribui um peso a cada uma destas ligações. Existe depois um método para actualizar os pesos com base nos sucessos e falhas.

Nas abordagens do CBR com conhecimento intensivo, a aprendizagem pode também ter lugar no modelo de conhecimento geral, por exemplo, através de outros métodos de *machine learning* ou através da interacção com o utilizador. Assim, com uma interface correcta com o utilizador (quer seja um utilizador competente ou um perito), um sistema pode ampliar e refinar o seu modelo de conhecimento geral, bem como a sua base de casos anteriores, no decurso normal da resolução de problemas.

O caso precisamente aprendido pode finalmente ser testado reintroduzindo o problema inicial e verificando se o sistema se comporta como esperado.

3 Uma descrição mais detalhada do CBR

3.1 Áreas problemáticas no CBR

Tal como para a Inteligência artificial em geral, não existem métodos CBR universais adequados para todos os domínios de aplicação. De acordo com o modelo de tarefas apresentado atrás, os principais problemas relativos à investigação do CBR podem ser agrupados em cinco grandes áreas:

- Representação do Conhecimento.
- Métodos de Recolha de Casos.
- Métodos de Reutilização de Casos.
- Métodos de Revisão de Casos.
- Métodos de Retenção de Casos.

Um conjunto de soluções para estes problemas constitui um método CBR.

Nas secções seguintes será feita uma descrição sumária dos principais problemas relativos a estas áreas.

3.2 Representação de Casos

3.2.1 O que é um caso?

Um caso é um "pedaço" de conhecimento contextualizado, que representa uma experiência. Contém a lição que é objecto do caso e o contexto em que a lição pode ser aplicada.

Os casos podem ter muitas formas e tamanho. Podem abranger uma situação que evolui ao longo do tempo (por exemplo no projecto de um edifício ou no seguimento de um paciente através de uma sequência de doenças), podem representar um momento instantâneo ou podem cobrir um espaço de tempo entre estes dois extremos. O caso pode representar a resolução de um problema (por exemplo no projecto de um edifício), associar a descrição de uma situação com um resultado (por exemplo um caso legal) ou alguma combinação entre os dois.

O que é comum a qualquer caso é o facto de representar uma situação experimentada. Esta situação, quando relembrada mais tarde, forma um contexto onde o conhecimento embebido se presume aplicável.

Existe um outro critério para decidir se uma determinada experiência ou situação deve ser armazenada como um caso:

O caso ensina uma lição útil?

Este critério implica, no entanto, definir em que consiste uma lição útil. Talvez a melhor directriz que pode ser dada sobre esta matéria é que os casos armazenados numa base de casos devem contribuir para alcançar os objectivos do sistema de raciocínio. Existem diversos tipos de lições que os casos podem ensinar, nomeadamente:

- Como atingir um objectivo,
- como alcançar diversos objectivos simultaneamente,
- as circunstâncias em que a sequência de passos para alcançar um objectivo pode ser usada,

- o tipo de problemas que podem ocorrer quando se tenta atingir um objectivo (o que pode correr mal)
- os efeitos de uma acção.

Em geral, um caso ensina uma lição útil quando exemplifica uma nova forma de fazer qualquer coisa. Podemos dizer que a idéia chave para guardar um caso é:

> Se aquilo que é diferente numa nova situação ensina qualquer coisa que não pode facilmente ser inferido dos casos já guardados na base de casos, então é útil guardá-la como um novo caso.

3.2.2 O Papel do Conhecimento Geral

Existem diversas implicações da definição de caso exposta atrás. Primeiro, implica que os casos não são o único tipo de conhecimento para que os sistemas de raciocínio funcionem. Se estamos a implementar um modelo cognitivo completo queremos incluir tanto casos como generalizações desses casos. As generalizações são o meio que permite detectar o que é normal. Além disso, implica que a organização dos casos na memória se altere dinamicamente ao longo do tempo e com a experiência. Aquilo que começa como uma nova experiência, ensinando uma nova lição, pode eventualmente tornar-se a norma. O modelo cognitivo e a organização da memória têm de se alterar de modo a reflectir estas alterações.

Nos sistemas CBR o conhecimento geral tende a ser registado em estratégias de adaptação, e nos modelos que alguns procedimentos de adaptação e de comparação utilizam.

3.2.3 O Conteúdo dos Casos

Como já vimos atrás, existem duas partes funcionais essenciais num caso:

- A lição que ensina – o seu conteúdo,
- O contexto no qual pode ensinar a sua lição, descrito pelos índices que determinam em que circunstâncias o caso será correctamente recolhido.

No que respeita às componentes de um caso, consideram-se normalmente três elementos, que podem não estar todos presentes na representação adoptada para um determinado domínio:

- Descrição do problema/situação - Estado do universo no momento em que o caso ocorreu e problema a resolver no momento.
- Solução - Solução obtida ou derivada para o problema descrito ou reacção à situação apresentada.
- Avaliação - Anotações que especificam as alterações produzidas no estado do universo pela aplicação da solução e aspectos relativos à criação dessa solução e ao surgimento de reacções a essa solução.

A função desempenhada pelos casos no processo de raciocínio determina os elementos que o compõem. Por exemplo, episódios que compreendem a descrição de um problema e respectiva solução podem ser usados para derivar a solução de um novo problema (é o que se passa, por exemplo, no CASEY). Casos compostos por situações e avaliações podem ser usados na avaliação de novas situações. Se, para além disto, os casos especificam uma solução, podem ser aplicados na avaliação das soluções propostas e na antecipação de potenciais problemas (por exemplo, no MEDIATOR). Se os casos incluírem ainda explicações relativas às avaliações (ligações causais entre a situação inicial, a solução e as avaliações) e de como foram corrigidas facetas inadequadas da solução, esses casos podem também ser aplicados na correcção de futuras falhas semelhantes às que contemplam. Se a solução vem acompanhada de informação sobre a

forma como foi derivada (obtida), o método de resolução que foi aplicado na solução do caso pode ser tentado em novos problemas para os quais a solução descrita nesse caso não é adequada. Se a informação sobre a derivação compreender ligações entre o problema e a solução, estas podem ser usadas em processos de adaptação de casos.

Em geral, um sistema de raciocínio baseado em casos decide sobre a aplicação de um caso anterior a uma nova situação examinando as semelhanças entre as descrições do problema na situação anterior e na nova situação. Se a nova situação é suficientemente semelhante à descrição do problema num caso anterior, então esse caso é seleccionado. Daqui decorre que a representação do problema deve ser suficientemente detalhada de modo a possibilitar ajuizar a aplicabilidade dos episódios em memória.

Na **representação do problema** são consideradas três componentes:

- Objectivos perseguidos na resolução do problema.
- Restrições relativas a esses objectivos.
- Características do problema e relações entre as suas componentes.

Os objectivos expressos na descrição do problema representam a intencionalidade do sistema. Em geral, o sistema de raciocínio tenta cumprir com sucesso esses objectivos.

Alguns dos objectivos mais comuns na resolução de problemas são **planear**, **criar** e **diagnosticar** em que, adicionalmente, pode ser especificado o que se pretende planear, criar ou diagnosticar. Por exemplo, o sistema CHEF faz planeamento de receitas culinárias, enquanto o JULIA cria refeições e o PROTOS diagnostica complicações cardíacas. Nas tarefas de interpretação (discutidas no próximo capítulo) alguns dos possíveis objectivos são **compreender**, **explicar** e **avaliar**, em que por sua vez poderá ainda ser discriminado o que deve ser compreendido, explicado ou avaliado. Por exemplo, o BATTLE PLANNER tem como uma das suas tarefas compreender a relevância de determinados aspectos de uma estratégia militar, o SWALE tem o objectivo de explicar a morte do cavalo *Swale* e o JULIA necessita numas situações de avaliar a qualidade de parte de uma solução, enquanto noutras pretende avaliar a solução como um todo.

As restrições descritas no problema impõem condições na persecução dos objectivos. Por exemplo, no CHEF cada novo problema envolve diferentes restrições – o sistema deve usar um conjunto específico de ingredientes ou a receita deve resultar num determinado paladar ou deve ser respeitado um limite calórico ou de preço.

Nas características do problema é considerada a informação que não foi abrangida pelas outras componentes e que é relevante para atingir os objectivos enunciados. Por exemplo, num sistema de diagnóstico médico, os resultados dos exames médicos do paciente constituem descritores da situação, necessários para atingir o objectivo que é o diagnóstico do doente. Outro exemplo de características do problema pode ser dado por um sistema de planeamento de refeições que pode não estar restringido àquilo que existe no frigorífico (eventualmente pode obter os ingredientes para a refeição num supermercado), mas aquilo que está presente no frigorífico deve influenciar a solução do problema e aparecer especificado como característica do problema.

Uma segunda componente num caso é a **descrição da solução do problema**. Vários tipos de soluções são considerados nos sistemas baseados em casos. Por exemplo, para problemas de *design*, a solução descreve um artefacto. Em situações de interpretação a solução compreende a interpretação ou classificação atribuída ao caso. Já quando o que se pretende é a explicação para um problema, a solução aparece na forma de uma explicação. Em resumo, na solução são descritos os conceitos, objectos ou ligações relativos aos objectivos descritos no problema, tendo ainda em conta as restrições e restantes elementos contextuais especificados nesse mesmo problema.

Quando o sistema de raciocínio selecciona um caso para resolver um novo problema usa a descrição da solução para gerar uma nova solução. A descrição da solução inclui outras componentes para além da solução propriamente dita, as quais são úteis no processo de adaptação e avaliação de novos casos. As componentes da solução que são consideradas úteis no processo de raciocínio são as seguintes:

- A solução propriamente dita.
- O conjunto de passos usados na resolução do problema.
- O conjunto de justificações relativas às decisões tomadas na resolução do problema.
- Soluções alternativas que não foram escolhidas (e os passos de raciocínio e as justificações que lhe estão associadas).
- Soluções recusadas (e os passos de raciocínio e as justificações que lhe estão associadas).
- Expectativas sobre as consequências da aplicação da solução.

Relativamente às componentes da solução propriamente dita, estas já foram descritas. No que respeita ao conjunto de passos usados na resolução do problema, vários sistemas baseados em casos (por ex., ARIES, JULIANA e JULIA) incluem na representação do caso o conjunto de passos que foram efectuados. Isto permite que esses passos sejam repetidos na resolução de novos problemas.

Esta componente da solução aparece na forma de uma lista das inferências envolvidas na sua construção: primeiro teve lugar a inferência *x*, depois a *y*, e assim por diante. Cada passo tem também associadas as fontes de conhecimento envolvidas nas inferências que foram feitas. Por exemplo, se o sistema de raciocínio derivou uma parte da solução recordando um caso, adaptando-o de uma determinada forma, então é guardado que foi feita inferência baseada em casos e como fonte de conhecimento foi usado o caso *xpto*, que foi adaptado recorrendo a um determinado conjunto de heurísticas. Por exemplo, na resolução de um problema, o sistema JULIA decide servir uma pizza sem produtos lácteos. O conhecimento usado e o tipo de inferência realizada está representado na figura 13. Esta figura representa que o passo de raciocínio realizado foi uma inferência baseada em casos, o caso utilizado foi *Pizza Bela Itália* e que foi adaptado utilizando a regra *eliminar-elemento-secundário*, aplicada ao ingrediente *queijo* no *prato principal* dessa refeição.

```
Raciocínio:
  Passo 1:
    Tipo: inferência baseada em casos
    Caso fonte: Pizza Bela Itália
    Adaptação: eliminar-ingrediente-secundário(queijo, prato principal)
```

Figura 13 - Um passo de raciocínio.

As justificações associadas à solução do caso fornecem meios para conduzir o processo de adaptação de uma solução anterior. Por exemplo, o programa JULIANA adapta uma refeição que foi servida no verão para ser agora apresentada no inverno. A refeição de verão continha melão, que não é fácil de obter no inverno. Na justificação associada a esta parte da solução do caso em memória aparece que foi escolhido o melão porque é uma fruta de verão, a estação do ano é verão e é um produto económico. Substituindo verão por inverno na justificação, JULIANA é levado a escolher um fruto em que: o fruto é de inverno porque a estação é inverno e é um fruto económico. Com base nesta justificação o sistema JULIANA vai considerar a escolha de maçãs, pêras ou laranjas.

Finalmente, há que salientar que as justificações e os passos de raciocínio são dois conceitos fortemente relacionados. As justificações fornecem argumentos para as soluções propriamente ditas e para a escolha dos passos de derivação. A figura a seguir mostra uma instância de um par justificação/passo de raciocínio. Aqui a justificação fornece o argumento para o passo de raciocínio aplicado – o leque de escolhas foi aumentado devido à escassa quantidade de quantidade de ingredientes.

```
Pratos principais:
  Raciocínio:
       Passo 1: Aumenta leque de escolhas
  Valor:
       Prato principal 1: Salmão grelhado
       Prato principal 2: Grelhada de marisco
  Justificação: Escassa quantidade de alimentos
```

Figura 14 - Raciocínio e justificação para servir dois pratos principais.

Outra componente da descrição da solução é a descrição de soluções alternativas, mas que não foram seleccionadas. A estas juntam-se os passos de raciocínio conjuntamente com as justificações associadas. Quando um caso com esta informação é recolhido, as soluções alternativas podem ser usadas na derivação de uma nova solução quando a solução principal foi, por alguma razão, posta de parte. De notar que estas soluções alternativas não têm avaliações associadas (uma vez que não foram aplicadas no passado) o que impede saber *a priori* quais as suas implicações.

As soluções alternativas recusadas, juntamente com os passos de raciocínio e as justificações que lhe estão associadas, são úteis na avaliação da solução proposta. Uma solução que foi recusada no passado pressupostamente voltará a ser recusada caso venha a ser gerada, ainda que por um caminho de inferência diferente.

A descrição da solução pode ainda incluir expectativas sobre as consequências da sua aplicação. Quando uma solução é criada, é comum que o sistema de raciocínio comporte expectativas sobre os efeitos da solução proposta. O sistema pode ainda ter expectativas sobre a ocorrência de problemas no decurso da aplicação da solução, bem como sobre a natureza desses problemas. Se estas expectativas são representadas nos casos, então o mecanismo de raciocínio pode determinar se uma dada solução tem efectivamente os efeitos previstos por comparação dos efeitos reais com aqueles que eram esperados.

Com estas duas componentes, podem ser resolvidos novos problemas (com base em casos) encontrando primeiro um caso relevante e depois a solução desse problema para se ajustar à nova situação. Por exemplo, o CASEY, um sistema para diagnóstico de problemas cardíacos, usa casos com apenas estas duas componentes. Quando este sistema encontra na sua base de casos um caso anterior semelhante a partir do qual raciocina, é várias ordens de grandeza mais eficiente que o programa baseado em modelos em que se baseia.

No entanto, nas situações com muitas ambiguidades podem resultar diversas imprecisões do raciocínio com base em caso que apenas contêm o problema e a solução. Em particular, tomando em consideração apenas as soluções não testadas de casos anteriores para propor novas soluções, tanto podemos estar a propor boas soluções como soluções deficientes. O CASEY evita este problema porque armazena (retém) apenas as soluções correctas e porque o seu modelo do domínio de aplicação, que conduz o processo de adaptação, é bastante preciso. No entanto, em muitas situações não existe um modelo preciso e o sistema é "obrigado" a cometer erros. Um sistema que usa o conhecimento que possui de forma descuidada para resolver os problemas e

retém todos os novos problemas e soluções torna-se mais eficiente ao longo do tempo, mas também repete os seus erros tão frequentemente quanto sugere boas soluções.

A terceira componente num caso é constituída pelas **avaliações**. As avaliações especificam os efeitos da aplicação da solução e questões relativas a esses efeitos. As avaliações incluem os efeitos sobre o ambiente em que o sistema opera, bem como uma interpretação desses efeitos. Esta informação pode ser aplicada na antecipação de potenciais problemas e das consequências de uma nova solução. As avaliações podem ainda comportar informação sobre as expectativas falhadas, sobre a falha de uma solução, ou explicações sobre a ocorrência de efeitos indesejáveis e do que foi feito para corrigir o caso de modo a evitá-los. Fazem parte das avaliações os seguintes elementos:

- Os efeitos da solução sobre o universo.
- Se os efeitos comprometem ou não os resultados pretendidos com a solução.
- Se os efeitos estão ou não de acordo com as expectativas.
- Explicações das violações de expectativas e/ou falhas.
- Estratégias de correcção do caso.
- O que deveria ter sido feito para evitar os efeitos indesejáveis.
- Ponteiros para caminhos de inferência alternativos na construção de uma solução.

Um primeiro elemento das avaliações é a descrição das consequências da aplicação da solução. Por exemplo, a descrição dos efeitos de um caso de *design* deve especificar como é que o *design* proposto resultou quando materializado, e em que medida cumpriu com os objectivos do problema:

- A dificuldade da sua concretização foi a esperada?
- A sua funcionalidade revelou-se adequada?
- Agradou aos utilizadores?
- Foi esteticamente aceite?

Já em tarefas de planeamento, a descrição dos efeitos sobre o universo comporta a descrição das modificações resultantes das acções que compõem o plano. Por exemplo, depois de ter dividido uma laranja entre duas crianças, o efeito pode ter sido ambas terem comido a respectiva metade ou uma das crianças ter feito sumo com a sua metade e a outra ter usado a casca para fazer um bolo. Cada um destes efeitos da aplicação do plano leva a, no futuro, serem tomadas diferentes decisões na geração de um plano para uma situação idêntica. Por exemplo, se uma criança fez sumo com a parte que lhe foi entregue e a outra usou a casca, então o que faz sentido no futuro, em vez de dividir a laranja ao meio, é dar a uma delas a polpa e à outra a casca.

Outro tipo de informação fornecida pelas avaliações é se os efeitos da solução comprometem ou não os objectivos perseguidos. Uma avaliação com esta natureza permite ao sistema prever se uma solução anterior deve ou não ser ensaiada para um novo problema.

Para além dos efeitos da solução e de informação sobre consequências desses efeitos sobre os objectivos, outro elemento das avaliações diz respeito às expectativas que o sistema de raciocínio tem relativamente a esses efeitos. Uma solução pode ser bem sucedida relativamente aos objectivos estabelecidos e, no entanto, falhar no que concerne às expectativas do sistema, assim como uma solução pode falhar em termos dos objectivos e esse resultado estar de acordo com as expectativas.

As avaliações sobre se as expectativas se verificam e em que condições fornecem ao sistema um meio de prever os efeitos de uma determinada solução. Adicionalmente, expectativas não verificadas "alertam" o sistema de raciocínio para a inexistência de um conhecimento completo do domínio. A interpretação da informação sobre as expectativas falhadas dá pistas ao sistema sobre conhecimento que é necessário adquirir.

Quando os efeitos de uma solução estão de acordo com as expectativas e com os objectivos do sistema, não há muito mais conclusões a tirar. No entanto, quando os efeitos contrariam as expectativas ou comprometem o sucesso da solução, a questão que se põe é: porquê essa falha? É então útil guardar explicações relativas à solução e aos efeitos que comprometem as expectativas ou ainda sobre os efeitos que inviabilizam a solução.

As explicações sobre a violação de expectativas e sobre os efeitos indesejados de uma solução são úteis na derivação de estratégias de correcção de casos.

Se as explicações se referem a efeitos que inviabilizam a solução então essas explicações são usadas para derivar conhecimento sobre o que deve ser ou não ser feito para evitar essa falha.

Finalmente, é frequentemente útil ter ponteiros para métodos alternativos de resolução de um problema quando o que se está a utilizar está prestes a falhar na construção de uma solução interessante. Este conhecimento é também guardado como avaliação e aparece, por exemplo, no sistema PROTOS que guarda ponteiros para casos alternativos na forma de ponteiros etiquetados pelas diferenças entre casos, evitando assim, *in extremis*, que o sistema escolha um caso inadequado.

Até aqui foi feito um levantamento dos elementos que têm sido considerados na representação de episódios experimentais nos sistemas de raciocínio baseado em casos. A questão que agora se põe é de como representar este conhecimento diversificado.

Os formalismos de representação de conhecimento com origem na área de Inteligência Artificial (por exemplo, guiões, redes semânticas e regras) têm sido largamente utilizados para representar o conteúdo de um caso numa forma manipulável por sistemas automáticos de raciocínio. Há, no entanto, uma outra questão que se coloca: Que representação é adequada quando o sistema deve interactuar com seres humanos?

As representações adequadas para a máquina não o são necessariamente para o homem. Acresce que muito do conhecimento, que do ponto de vista do ser humano é útil representar num caso, não pode ser descrito usando os formalismos actualmente disponíveis.

A criação do programa ARCHIE é um bom exemplo deste tipo de questões. O ARCHIE foi conceptualizado como um sistema de apoio a arquitectos. A ideia subjacente era de que o arquitecto deveria fornecer ao ARCHIE a descrição do seu problema e o ARCHIE deveria recolher casos e apresentá-los ao arquitecto que os entendia como orientações e fonte de inspiração na realização de um novo projecto. Aconteceu que a representação de casos através de esquemas, usada no ARCHIE, embora fosse adequada do ponto de vista do sistema, era extremamente desconfortável para os arquitectos. Os esquemas comportavam demasiada informação e de difícil compreensão por parte dos especialistas. Esta constatação levou ao desenvolvimento do ARCHIE-2, uma evolução do sistema anterior. Neste novo sistema, os casos incluem representações gráficas e descrições em linguagem natural, essenciais para o arquitecto poder tirar partido da informação contida no caso.

Em resumo, embora seja necessário representar os casos recorrendo a uma notação simbólica com vista à sua utilização pelo sistema automático de raciocínio, é muitas vezes útil, e mesmo indispensável, explicar pelo menos partes dos casos recorrendo a texto ou imagem, por forma a tornar os conteúdos acessíveis ao utilizador. As formas de representação dirigidas para os seres humanos podem ainda assumir as mais variadas formas, sendo os sistemas multimedia um bom exemplo da utilização de representações ajustadas ao homem.

Importa ainda salientar que os elementos que foram considerados na representação de um caso constituem uma síntese daquilo que tem sido considerado nos sistemas mais significativos de raciocínio baseado em casos. Em geral, não aparecem num mesmo sistema todos estes elementos.

3.3 Indexação de Casos

A recolha dos episódios que são potencialmente úteis na resolução de um novo problema é um dos processos essenciais em sistemas raciocínio baseado em casos. A questão fundamental que se coloca é a de como fazer com que o sistema recorde os casos potencialmente úteis e no momento apropriado. Esta questão é geralmente referida como o problema da indexação de casos.

A maioria dos sistemas de bases de dados usa índices para acelerar a pesquisa e recolha de dados. Por exemplo, pode ser criado um índice para o sobrenome das fichas de pessoas numa base de dados. Um índice é uma estrutura que pode ser manipulada em memória e pesquisada de forma bastante rápida. Isto significa que o computador não tem que pesquisar em cada ficha guardada em disco, o que seria bastante mais lento. Os sistemas CBR usam também índices para acelerar o processo de recolha de casos relevantes. A informação contida num caso é, normalmente, de dois tipos:

- informação indexada que é usada no processo de recolha,
- informação não indexada que pode fornecer informação contextual importante para o utilizador, mas não é usada directamente no processo de recolha.

Por exemplo, numa base de dados de pacientes de um médico, podem ser usados a idade, o sexo, o peso e a altura como características indexadas que podem ser usadas na recolha e ser incluída uma fotografia do paciente como característica não indexada. A fotografia não pode ser usada para recolha, mas pode ser útil para relembrar ao médico quem é o paciente.

O problema da indexação tem várias facetas. Uma primeira refere-se à atribuição de índices aos casos, na fase de serem guardados em memória, com vista a assegurar que estes são posteriormente seleccionados no momento certo. Os índices de um caso definem em que circunstâncias este é potencialmente útil. Uma segunda questão refere-se à organização dos episódios de modo a que o processo de procura na biblioteca seja eficiente e preciso. Relacionada com esta questão está a escolha dos algoritmos de recolha de casos. Para já iremos abordar a questão da indexação de casos propriamente dita e deixamos as questões relativas à organização da memória e algoritmos de recolha para as secções seguintes.

A indexação de casos deve ter em conta os seguintes pontos:

- Definir o vocabulário que o módulo de recolha deve usar.
- Recorrer a termos correntemente usados para descrever os conceitos que estão a ser indexados, independentemente de serem descritores específicos ou abstractos.
- permitir antecipar as circunstâncias em que a recolha de casos deve ser activada.

O conjunto de índices de um caso representa uma interpretação da situação descrita nesse caso, no sentido em que envolve a formação de uma perspectiva sobre uma situação e sobre as circunstâncias em que é útil recordar o caso que comporta essa situação. A indexação dos casos permite assim recolher no futuro aqueles que são mais úteis, independentemente de se considerarem índices representados por descritores específicos ou abstractos.

De facto, por vezes, a semelhança entre um caso e uma nova situação ao nível de descritores superficiais deve ser preterida em favor de semelhanças encontradas ao nível dos descritores abstractos mais úteis. Por exemplo, um escalonador pretende acelerar a produção de um produto procurando colocar duas máquinas a produzi-lo simultaneamente. Entretanto, tem vários tipos de casos disponíveis para o conduzir no processo de melhoria da produção:

- Casos em que foi bem ou mal sucedido na tentativa de tornar um processo produtivo mais eficiente.
- Casos em que procurou usar duas máquinas simultaneamente.

- Casos envolvendo a produção do mesmo produto, no mesmo dia da semana e em que a temperatura exterior era idêntica à observada na situação actual.

Consideremos que a temperatura exterior e o dia da semana pouco ou nada influenciam o processo produtivo. Partindo deste pressuposto, embora os casos incluídos no terceiro conjunto tenham vários factos específicos em comum com a situação corrente, é pouco provável que sejam tão úteis no apoio à decisão sobre a nova situação como são os casos pertencentes ao primeiro e segundo conjuntos. A diferença está em que os casos pertencentes aos dois primeiros conjuntos partilham com a nova situação os mesmos objectivos.

Do exemplo anterior conclui-se que os descritores a serem usados como índices devem ser cuidadosamente escolhidos se se pretende que sejam os episódios mais úteis que sejam recolhidos da base de casos.

Verifica-se que a escolha de índices a partir de descritores específicos, fáceis de extrair de um caso, é por vezes pouco eficiente. Sobretudo se comparada com a criação de índices por combinação mais ou menos complexa de descritores desde que essa combinação permita distinguir os casos entre si na utilidade que têm para a resolução do novo problema. A procura do aumento da produção ou a utilização simultânea de duas máquinas, como aparece nos dois primeiros conjuntos de casos, correspondente à utilização de índices abstractos na descrição de um processo produtivo. No entanto, estes descritores são, em geral, mais úteis ao processo de recolha de casos do que os descritores específicos.

O exemplo anterior mostra existirem duas questões a ter em conta na escolha dos índices. Uma é a necessidade de encontrar formas de descrever e representar um caso adequadas à sua comparação com o novo problema, ao longo das dimensões apropriadas. Por outras palavras, as tarefas e os domínios devem ser analisados com vista a determinar que descritores são funcionalmente relevantes e devem ser usados para descrever os casos. Uma segunda questão diz respeito à determinação do tipo de descritores que devem actuar como índices.

Para cada caso é necessário ter índices que identifiquem as situações em que o caso é útil. Designa-se este processo por *atribuição* ou *selecção de índices*. A atribuição de índices é assim o processo de escolha dos descritores relevantes para a selecção de um caso. O vocabulário de indexação define que descritores devem ser usados como índices numa classe de casos.

Da análise dos processos envolvendo o recurso à memória e da experiência acumulada na construção de sistemas de raciocínio baseado em casos, resultaram as seguintes orientações na escolha de índices:

- devem ser preditores, i.e., devem ser aqueles que determinam a solução ou avaliação que compõem um caso.
- Os prognósticos obtidos devem ser úteis, ou seja, devem estar de acordo com a finalidade da selecção do caso.
- Devem ser suficientemente abstractos de modo a que os casos em que estes aparecem sejam seleccionados num leque tão variado de situações quantas aquelas em que são potencialmente úteis.
- Devem ser tão concretos quanto necessário de modo a serem reconhecidos no futuro sem grande esforço de inferência.

Da aplicação destas orientações resulta a selecção de índices diferenciadores dos casos segundo aspectos que permitem que no processo de recolha sejam seleccionados os episódios mais promissores na resolução do novo problema.

O problema da indexação compreende ainda uma outra questão que diz respeito ao vocabulário de indexação e de representação dos casos. Um vocabulário de representação

consiste em duas componentes: um conjunto de dimensões e um conjunto de símbolos que representam os valores que essas dimensões podem assumir. Dimensões e valores podem ser vistos, por exemplo, como predicados e argumentos da representação usada no cálculo predicativo de primeira ordem.

Na escolha das dimensões e valores constituintes do vocabulário de representação das situações, soluções e avaliações respeitantes a uma classe de casos emergem duas abordagens:

Abordagem funcional - é examinado o conjunto de casos disponíveis e as tarefas em causa, com ênfase para a determinação do papel de cada caso no processo de resolução de problemas .

Abordagem cognitiva - é examinado que casos são recordados pelo perito quando está a trabalhar sobre problemas na tarefa em questão. É dada particular atenção às semelhanças consideradas relevantes pelo perito, entre a nova situação e os casos recordados. Desta forma são detectados os descritores relevantes num determinado conjunto de circunstancias.

Qualquer destas análises fornece informações sobre que dimensões são importantes e que valores assumem em cada circunstância a considerar. Desta análise resulta ainda a aferição de qual o nível de detalhe a que devem ser representados os casos.

Uma das preocupações na área de raciocínio baseado em casos tem sido a de criar processos automáticos de selecção de índices. O número de métodos de atribuição automática de índices tem aumentado consideravelmente, nomeadamente:

- Indexar os casos pelas suas características que tendem a predizer o domínio, i. e., os descritores do caso que são responsáveis pela resolução do problema ou que influenciam o resultado. Neste método o domínio é analisado, são calculadas as características que tendem a ser importantes. Estas são colocadas numa tabela e todos os casos são indexados pelos valores nestas características. Os índices são assim escolhidos com base numa lista fornecida por quem desenvolveu o sistema que indica quais as dimensões que devem ser consideradas para indexação. Esta técnica é denominada por indexação com base em listas (*checklist-based indexing* no [Kolodner, 93]).
- Extracção de diferenças entre um caso e os restantes da base de casos e usar essas diferenças para indexação. Na indexação baseada em diferenças são seleccionados os índices que diferenciam um caso dos restantes (um exemplo é o `CYRUS`). Durante este processo o sistema determina quais as características de um caso que o diferenciam de outros semelhantes, escolhendo como índices aquelas características que melhor diferenciam os casos.
- Selecção dos índices com base em explicações sobre o sucesso ou insucesso de uma solução. Neste método, ao contrário dos dois anteriores, a selecção de índices é fortemente dependente do conhecimento disponível do domínio.
- Métodos baseados em explicação e parecença, que produzem um conjunto apropriado de índices para casos abstractos criados a partir de casos que partilham determinadas características comuns, enquanto as características não partilhadas são usadas como índices para os casos originais.
- Métodos de aprendizagem indutiva, que identificam características preditivas que são depois usadas como índices. Estas técnicas são amplamente utilizadas (por exemplo no `REMIND`) e são utilizadas diversas variantes do algoritmo ID3 para regra de indução.

- Técnicas baseadas em explicação, que determinam as características relevantes para cada caso. Este método analisa cada caso para determinar quais das suas características são preditivas. Os casos são depois indexados por estas características.

No entanto, apesar do sucesso de muitos dos métodos automáticos, Janet Kolodner acredita que as pessoas tendem a fazer uma melhor escolha de índices do que algoritmos e, por isso, os índices para aplicações práticas devem ser escolhidos à mão [Kolodner 1993].

Na prática, é comum combinar os métodos que recorrem a listas de indexação e a conjuntos de diferenças, ou ainda fazer uma combinação dos três primeiros métodos acima referidos.

3.3.1 Indexação com base em listas

A indexação automática com base numa lista de indexação é feita de uma forma fixa e sobre um conjunto bem delimitado de dimensões. Por exemplo, o sistema MEDIATOR, aplicado em tarefas de mediação com base em casos, usa como dimensões de indexação o tipo e função do objecto disputado, o tipo de disputantes e o tipo de relacionamento entre eles. No CHEF os casos de planeamento de receitas culinárias são indexados pela textura, sabor, forma de preparação e ingredientes envolvidos. Estas são as dimensões que as pessoas envolvidas no desenvolvimento dos sistemas MEDIATOR e CHEF, ao analisarem os respectivos domínios de aplicação, consideraram serem preditoras da utilidade dos casos na resolução de um novo problema.

Dada uma lista de indexação, o processo de selecção de índices é simples. Para cada dimensão referida na lista há que encontrar ou calcular o valor dessa dimensão nos casos, o qual passa a ter a categoria de índice.

Uma questão que se põe na construção de listas de indexação é a sua sensibilidade ao contexto. Por outras palavras, as dimensões segundo as quais os casos devem ser indexados dependem do contexto, mesmo nas situações em que a sua utilização está prevista para um único domínio. Por exemplo, no recrutamento de pessoas para uma empresa podemos considerar que a cor dos olhos é importante no recrutamento de modelos, mas já não o é no recrutamento de condutores de camiões.

Duas técnicas são correntemente utilizadas no sentido de ter listas de indexação sensíveis ao contexto:

- Recurso a várias listas, organizadas em volta dos diferentes contextos conhecidos.
- Existência de indicadores de quantas vezes cada índice foi útil na recolha de casos que contribuíram para a construção da solução. Esta informação serve depois para modificar a lista de indexação, retirando dela as dimensões que se revelaram irrelevantes.

O primeiro método permite considerar diferentes índices para diferentes contextos, enquanto o segundo compreende um mecanismo de adaptação de uma lista, removendo índices que se revelaram ineficientes na selecção de casos.

Ambos os métodos contribuem para criar listas de indexação específicas para os vários contextos considerados.

3.3.2 Indexação com base em diferenças

A indexação de casos tem como objectivo diferenciá-los de forma a que no momento de fazer a selecção sejam escolhidos aqueles cuja solução esteja mais próxima da solução do novo problema.

Se pode ser mantida em memória informação sobre o que é comum aos vários casos armazenados, então também é possível determinar quais os descritores de um caso que o diferenciam de outros semelhantes. São esses descritores que são considerados na indexação

baseada nas diferenças entre casos. Se, por exemplo, em geral, as reuniões do governo têm lugar em Lisboa, não faz sentido indexar casos sobre `reuniões diplomáticas por localização: Lisboa` dado que este descritor não é discriminatório. Já, pelo contrário, se um encontro se realiza em Viseu, a localização passará nesse caso a constituir um bom índice.

No entanto, nem todos os descritores que são diferenciadores de casos semelhantes constituem índices úteis. Por exemplo, as dimensões da sala de reunião podem diferenciar duas reuniões, mas provavelmente não serão preditoras de nada importante no caso. Assim, não há interesse em indexar casos por uma determinada dimensão, apesar de ser discriminatória, se tal não produzir nenhuma inferência útil.

Para assegurar que a indexação baseada em diferenças é eficaz, esta deve ser combinada com um método que indique quais são os descritores com boa capacidade de predição.

3.3.3 Indexação com base em explicações

Os métodos de indexação baseados em listas e em diferenças fornecem meios simples de computação de descritores potencialmente preditores de casos úteis à geração de uma nova solução. No entanto, estas abordagens têm fragilidades resultantes de envolverem a escolha de índices frequentemente preditores, sem analisar os casos individualmente. Isto conduz ao problema de serem escolhidos para indexação descritores que não são preditores para todos os casos na base de casos, enquanto outros descritores que são preditores para alguns casos, mas não o são em geral, não são considerados para indexação. Assim, estes métodos conduzem frequentemente à recolha de muitos casos supérfluos, pois não produzem uma discriminação adequada entre casos. Isto implica a necessidade de recorrer a um processo adicional de ordenamento dos casos que são recolhidos, de modo a seleccionar aqueles que são potencialmente mais úteis.

Os métodos de indexação baseados em explicações são vocacionados para uma escolha de índices caso a caso. Nestes métodos o sistema de raciocínio tenta explicar como obteve uma solução ou porque não a conseguiu obter, usando em seguida técnicas de generalização baseadas em explicações para as explicações criadas. Os índices são então escolhidos com base nas generalizações das explicações.

Na indexação baseada em explicações, o conhecimento sobre o domínio é usado na determinação de quais os factos mais relevantes num caso e quais os que podem ser ignorados.

Um exemplo de indexação baseada em explicações surge no CHEF. Neste sistema os índices criados a partir de explicações ajudam o sistema de raciocínio a evitar erros anteriormente cometidos. Consideremos, por exemplo, que o CHEF acabou de derivar uma receita de *soufflé* de morangos. Põe a receita em execução e a massa não alteia como era previsto. Seguidamente o CHEF questiona-se porque não alteou a massa. Recorrendo a conhecimento sobre o domínio chega a uma explicação que relaciona a solução, o estado do universo e as avaliações relativas à execução do plano: um efeito lateral que é a libertação de líquido durante a cozedura dos morangos elimina uma condição necessária para que a massa alteie, que é o equilíbrio entre líquidos e fermento. Em termos de ingredientes envolvidos, os morangos são responsáveis pelo efeito lateral ao produzirem mais líquido do que a quantidade de fermento utilizada pode suportar. Os índices baseados em explicações a serem criados neste caso incluem que a receita é do tipo `soufflé` e inclui `morangos`. Estes índices não são, no entanto, suficientemente interessantes dado serem muito específicos. Um conjunto de índices de aplicação mais geral pode, no entanto, ser construído generalizando os descritores da situação que são responsáveis pela falha até ao nível mais abstracto em que a explicação ainda se aplique. Neste domínio, qualquer fruto fresco causará o mesmo problema e o índice escolhido será: receita tipo `soufflé` incluindo `frutos frescos`.

Em termos gerais, a indexação com base em explicações sobre falhas funciona da seguinte forma: depois do sistema de raciocínio descobrir que cometeu um erro, tenta explicá-lo; depois

de explicar o erro, obtém os descritores da situação que aparecem na explicação e que foram responsáveis pelo erro; seguidamente generaliza esses descritores até ao ponto em que a explicação ainda se aplica (supondo que `morangos` são definidos como `fruta fresca` e `fruta fresca` como `fruta`, a explicação para a falha na preparação do *soufflé* continua a aplicar-se se substituirmos `morangos` por `fruta fresca`, mas já não se aplica quando substituímos `fruta fresca` por `fruta`, já que, por exemplo, os frutos secos não produzem qualquer líquido quando cozinhados).

Um segundo exemplo é o sistema JULIA. Aqui os índices criados a partir de explicações servem à geração de novas soluções. Neste caso o sistema de raciocínio escolhe, para índices, descritores observados como responsáveis na resolução de um problema.

JULIA acaba de planear uma refeição e decidiu fazer uma tarte de tomate como prato principal. Escolheu tarte de tomate porque é verão, nesta estação é fácil encontrar tomate fresco à venda e a Ana, que é vegetariana, foi convidada para a refeição. Esta explicação (do porquê ter sido escolhida esta solução) é depois generalizada resultando na seguinte indexação do caso: o objectivo é escolher um prato principal, é verão, há um(a) convidado(a) vegetariano(a) e deve ser incluído tomate.

O conjunto de índices é obtido a partir da informação armazenada sobre o que levou à tomada de cada decisão e de como essas decisões afectam outras decisões. O JULIA conhece as justificações para escolher tarte de tomate. Algumas derivam de particularidades do problema (uma vegetariana vai participar na refeição) outras do estado do universo (é verão). Assim, os descritores da situação que determinam a solução (neste caso o tipo de convidado e a estação do ano) são os escolhidos. Estes descritores podem depois ser generalizados como acontece no CHEF, produzindo assim outros de aplicação mais alargada.

Em conclusão, o processo de selecção de índices baseados em explicações envolve os seguintes passos:

1. Criação das explicações.
2. Selecção dos descritores relevantes que são observáveis a partir das explicações.
3. Generalização dos descritores observáveis tão forte quanto essas generalizações se mantenham observáveis a partir das explicações.

Como já foi referido anteriormente, vários sistemas baseiam o processo de indexação em mais do que um dos métodos descritos. Na secção seguinte apresenta-se a forma como vários métodos podem ser combinados.

3.3.4 Indexação com base em vários métodos

Como já foi dito, alguns sistemas baseiam o processo de indexação em listas de indexação e nas diferenças entre casos, outros juntam ainda técnicas baseadas em explicações.

Já que nos métodos baseados em listas, estas identificam que dimensões devem ser consideradas para indexação e nos baseados em diferenças, estas designam que valores segundo uma dada dimensão são úteis para indexação, a combinação das duas abordagens leva a indexar unicamente os valores segundo uma dimensão que seja preditora de uma solução e que diferenciem um caso de outros similares. O algoritmo usado no sistema CYRUS na selecção de índices mostra como esta combinação se desenrola:

1. Determina o contexto de um problema através da sua classificação como um problema de um determinado tipo.
2. Selecciona as dimensões que são conhecidas como sendo preditoras relativamente ao tipo de problema em questão (usa listas de indexação contextualizada).

3. Para cada uma das dimensões calcula o valor que assumem nesse problema (resultam pares dimensão/valor).
4. Dos pares dimensão/valor criados, rejeita os que são normativos.
5. Os pares dimensão/valor que não foram rejeitados são aqueles que são usados na selecção de um caso.

Este algoritmo permite ao CYRUS indexar no subconjunto dos descritores indexáveis unicamente aqueles que não aparecem ao longo de vários casos. No primeiro passo o problema é classificado com vista à determinação de que informação contextual deve ser tida em conta na escolha dos índices. No passo seguinte são escolhidas as dimensões para indexação. Isto é feito recorrendo a listas de indexação sensíveis ao contexto. No terceiro passo, o CYRUS calcula o valor que o problema tem para cada uma das dimensões seleccionadas. No fim deste passo é criada uma lista de pares dimensão/valor (por exemplo, `localização = Viseu`). Seguidamente apagada da lista os pares que conclui não serem preditores. Os restantes são aqueles que são úteis na selecção de um caso

Quanto aos métodos de indexação baseada em explicações, embora forneçam índices úteis, não garantem que estes discriminem entre casos semelhantes. Assim a combinação destes métodos com técnicas baseadas em diferenças é, em geral, vantajosa. Um método baseado em explicações pode ser aplicado para derivação de um conjunto de índices preditores, enquanto que um método baseado em diferenças é usado para eliminar aqueles índices que não são discriminatórios.

Um outro problema que ocorre quando técnicas baseadas em explicações são aplicadas isoladamente é que a realidade dos resultados obtidos com estes métodos é limitada pela qualidade do modelo do domínio usado para gerar explicações. Acontece que, na generalidade dos domínios nos quais o recurso a raciocínio baseado em casos é vantajoso, não existe um modelo completo do domínio. Uma forma de lidar com esta situação é combinar métodos baseados em explicações com métodos baseados em listas de indexação.

O CABER, um sistema de reparação e avarias, ilustra a utilização de técnicas baseadas em explicações associadas com outro método, neste caso listas de indexação. Este programa tem como tarefa a recuperação de avarias de sistemas mecânicos autónomos. O modelo do domínio permite construir partes de explicações, mas não é suficientemente completo para possibilitar a derivação de explicações completas sobre avarias. As listas de indexação são então usadas para indicar que descritores tendem a ser preditores adicionalmente aos encontrados a partir das explicações criadas. Aos índices derivados a partir das explicações é dada prioridade relativamente aos apontados como potencialmente relevantes pelas listas de indexação. O CABER usa estes dois níveis de prioridade para organizar hierarquicamente os casos em memória. Esta prioridade diferenciada pode também ser usada para determinar qual de dois casos em que se verifica a adequação parcial, relativamente a uma nova situação, deve ser o seleccionado para a resolução do problema.

O CHEF resolve o problema da não existência de conhecimento completo de outra forma. Utiliza indexação baseada em listas para os índices usados na selecção de planos, pois o melhor modelo que tem para a escolha entre planos alternativos são os próprios casos. No entanto, recorre à selecção de índices baseada em explicações para escolher os índices que são preditores de potenciais falhas. Aqui uma questão que se pode pôr é porque, tendo disponível um modelo causal para a detecção de falhas, este não é usado também para indexar as soluções correctas geradas pelo CHEF. O que acontece é que a utilização do modelo sobre o domínio é computacionalmente demorada e, em grande parte das situações, nem sequer é necessário ter disponíveis os detalhes sobre porque uma solução é correcta. Para além disto, um modelo sobre insucessos na aplicação de uma solução nem sempre é adequado para explicar porque uma

solução é correcta. Concretamente, no caso do CHEF, o modelo causal existente não é suficientemente completo para explicar os sucessos para todos os cenários. O facto dos planos funcionarem é por vezes, por si só, um modelo suficiente. No entanto, relativamente às falhas, o facto do CHEF ter meios de desenvolver raciocínio mais profundo permite a antecipação de potenciais insucessos futuros.

3.4 Armazenamento de Casos

O armazenamento de casos é um aspecto muito importante na implementação de sistemas CBR uma vez que deve reflectir o que os casos representam e considerar os índices que caracterizam os casos. A base de casos deve ser organizada numa estrutura manipulável que permita métodos de pesquisa e recolha de casos eficientes. Deve ser encontrado um compromisso entre os métodos de armazenamento que preservem a "riqueza" dos casos e os seus índices e métodos que simplifiquem o acesso e a recolha dos casos relevantes. Estes métodos são normalmente definidos pelo modelo da memória de casos. Os dois modelos de memória de casos com maior sucesso são o *modelo de memória dinâmica* desenvolvido por *Schank* e *Kolodner* e o *modelo exemplar-categoria* desenvolvido por *Porter* e *Bareiss*. Estas técnicas que estruturam os casos de forma hierárquica são ainda largamente utilizadas pelos investigadores na área da ciência cognitiva, mas nenhuma das ferramentas comerciais de CBR disponíveis usa estas técnicas. Em vez disso, as ferramentas comerciais ou guardam os casos em simples ficheiros de dados com uma estrutura horizontal ou em bases de dados relacionais, e usam os índices para referenciar os casos.

Na estrutura de memória horizontal ou linear os casos são armazenados sequencialmente numa lista, vector ou ficheiro e as características de cada caso são indexadas independente umas das outras. A principal vantagem é a facilidade em aprender com os casos recém resolvidos, pois acrescentá-los à memória é uma operação simples, bastando inseri-los no final ou no início da estrutura.

Na estrutura de organização hierárquica somente um pequeno subconjunto dos casos necessita ser considerado durante a recuperação. Geralmente a hierarquia é obtida com a ajuda de métodos de agrupamento indutivo. A vantagem das estruturas hierárquicas em relação à organização linear está na maior eficiência do processo de recuperação, já que não é necessário efectuar a comparação do novo caso com todos os casos de memória.

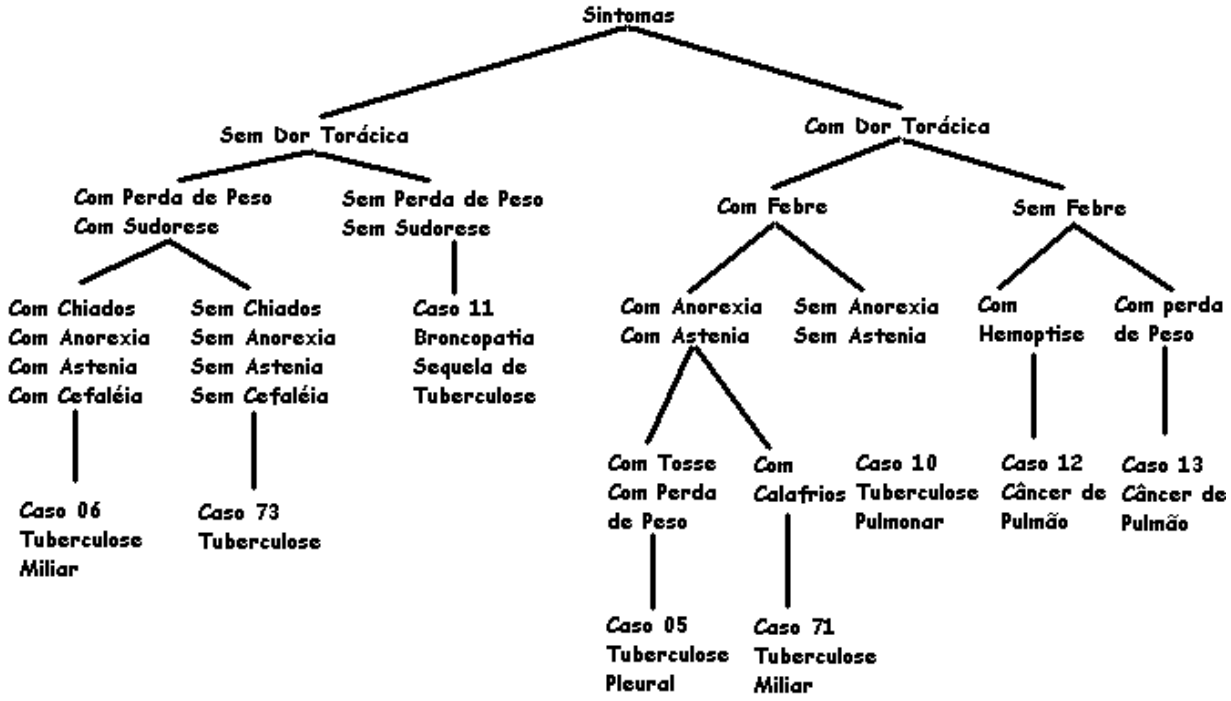

Figura 15 – Exemplo duma estrutura hierárquica de casos.

A principal desvantagem das organizações hierárquicas em relação à organização linear é a capacidade de aprendizagem, pois a adição de novos casos em estruturas hierárquicas não é uma tarefa simples, já que o caso deve ser colocado no local correcto da estrutura. Outra desvantagem é que estas estruturas consomem mais espaço de armazenamento.

3.4.1 O Modelo de Memória Dinâmica

O modelo da memória de casos neste método compreende pacotes de organização da memória ou MOP's. Os MOP's são um tipo de estrutura e são a unidade básica da memória dinâmica. O conhecimento acerca de classes de eventos pode ser representado usando dois tipos de MOP's:

- Instâncias representando casos, eventos ou objectos e
- abstracções representando versões gerais de instâncias ou de outras abstracções.

A memória de casos no modelo de memória dinâmica é uma estrutura hierárquica de pacotes de organização de memória episódicos (os E-MOPs), também chamados episódios genéricos (GEs), desenvolvidos a partir da teoria dos MOPs de *Schank*. A idéia básica é organizar casos específicos com propriedades semelhantes sob uma estrutura mais geral (um episódio geral ou genérico). Um GE contém três tipos de objectos: **normas**, **casos** e **índices**. As normas são características comuns a todos os casos sob um episódio geral. Os índices são características que distinguem os casos num GE. Um índice podem apontar para um episódio geral mais específico ou para um caso e é composto pelo **nome do índice** e pelo **valor do índice**. A figura seguinte ilustra a estrutura complexa de um episódio genérico, com os casos subjacentes e um episódio genérico mais específico.

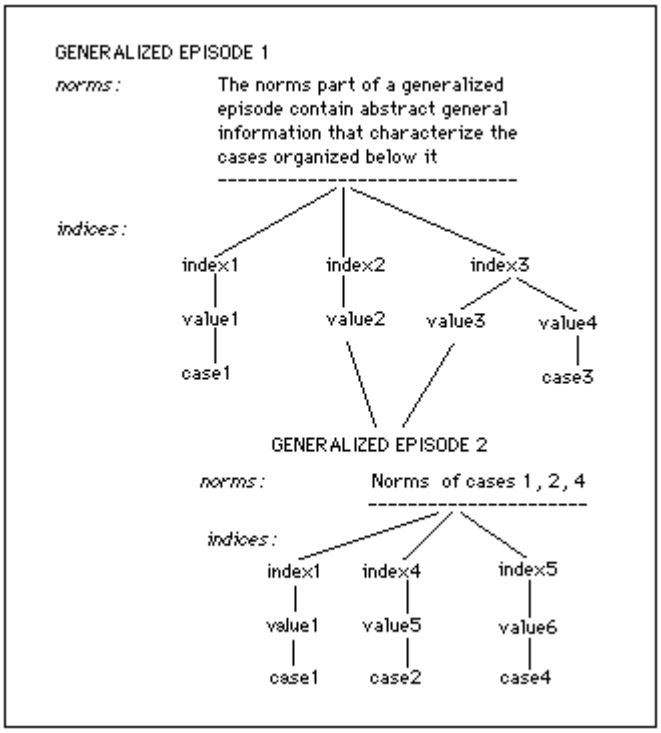

Figura 16 – Estrutura de casos e de episódios genéricos.

A memória de casos completa é uma rede de discriminação em que um nodo é um GE (contendo as normas), o nome de um índice, o valor de um índice ou um caso. Cada par **nome--valor** de um índice aponta de um GE para outro GE ou para um caso. O valor de um índice pode apenas apontar para um caso único ou para um GE único. O esquema de indexação tem redundância, uma vez que existem diversos trajectos para um determinado caso ou GE. Este facto está ilustrado na figura pela indexação do caso 1.

Quando é introduzida a descrição de um novo caso e se procura o caso mais semelhante, a rede de discriminação é percorrida de cima para baixo começando pelo nodo raiz. Quando uma ou mais características do novo caso coincide com uma ou mais características de um GE o caso continua a ser discriminado com base nas características restantes. A recolha de um caso é feita descobrindo o GE com a maioria das normas idênticas às das descrição do problema. Os índices sob este GE são percorridos em sequência para determinar o caso que contém a maioria das restantes características do problema.

Durante a fase de armazenamento de um caso, quando uma característica (nome do índice e valor do índice) de um novo caso coincide com a característica de um caso existente é criado um novo GE. Para discriminar os dois casos indexam-se com diferentes índices sob o novo GE (considerando que os casos não são idênticos). Deste modo a memória é dinâmica uma vez que partes semelhantes de dois casos são generalizados sob um novo GE, sendo depois os casos sob um GE indexados com base nas suas diferenças.

No entanto, este método pode conduzir a um aumento explosivo no número de índices à medida que o número de casos aumenta. Por esta razão, a maioria das implementações de CBR baseadas neste método limitam o número de índices admissíveis a um vocabulário limitado. É o que acontece no sistema CYRUS.

O CASEY guarda uma enorme quantidade de informação nos seus casos. Além de todas as características observáveis, são armazenadas as explicações para os diagnósticos encontrados e também a lista de todos os estados no modelo de problemas do coração. Estes estados, denominados estados genéricos, são também índices para os casos.

O papel principal de um GE é ser uma estrutura de índices para permitir o armazenamento, comparação e recolha de casos. As propriedades dinâmicas desta estrutura de memória podem,

no entanto, ser vistas como uma tentativa de construir uma estrutura de memória que integre o conhecimento de episódios específicos com o conhecimento geral dos mesmos episódios.

3.4.2 O Modelo Exemplar-Categoria.

O sistema PROTOS (desenvolvido por *Porter* e *Bareiss*) propõe uma forma alternativa de organizar os casos numa memória de casos. Este modelo organiza os casos com base na opinião de que o mundo real deve ser definido explicitamente com casos referidos como exemplares. A memória de casos é uma rede estruturada de **categorias**, **relações semânticas**, **casos** e **ponteiros para índices**. Cada caso é associado com uma categoria e um índice pode apontar para um caso ou para uma categoria. A cada uma das diferentes características é atribuída uma importância diferente na discrição de um caso pertencente a uma categoria. Estão disponíveis três tipos de índices, cada um dos quais pode apontar para um caso ou para uma categoria:

- **ligações para características**, que apontam dos descritores do problema (características) para um caso ou categoria (também chamados **recordações**),
- **ligações para casos**, que apontam das categorias para os casos a elas associados (os **exemplares**), que estão ordenados segundo a sua capacidade de tipificar a categoria, e
- **ligações diferenciais**, que apontam de casos para os casos vizinhos que apenas diferem num pequeno número de características.

Uma característica é, normalmente, descrita por um par **nome-valor**. Os exemplares de uma categoria são armazenados de acordo com a sua capacidade de modelar a categoria. Na organização da memória, as categorias estão inter-relacionadas por uma estrutura de regras semânticas (rede semântica) que contêm as respectivas características. Esta estrutura representa um conhecimento geral do domínio que possibilita as tarefas de explicação e interpretação de algumas tarefas do CBR.

A Figura 17 representa parte desta estrutura de memória, ilustrando as ligações das características e casos (exemplares) às categorias. Os índices sem nome são recordações (ligações das características para a categoria).

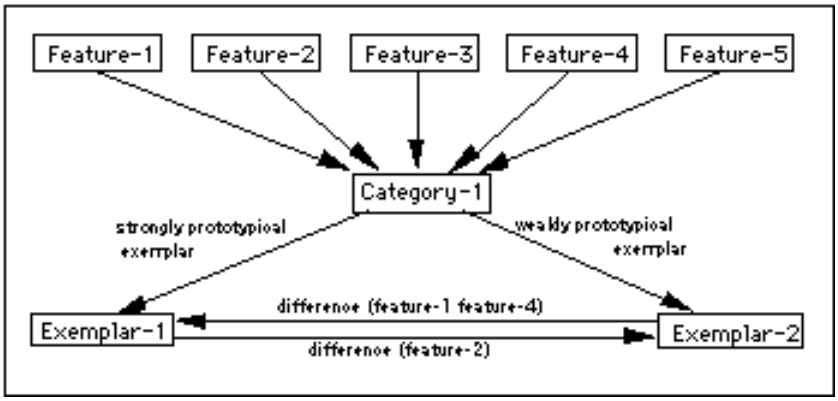

Figura 17 – Estrutura de Categorias, Características e Exemplares.

Encontrar um caso na base de casos que se assemelhe uma descrição introduzida é feito combinando as características do caso introduzido num ponteiro para o caso ou categoria que partilhe a máximo de características. Se uma recordação apontar directamente para uma categoria, as ligações para os seus casos mais prototípicos são percorridas e os casos recolhidos. Tal como foi dito atrás, o conhecimento geral do domínio é usado para permitir a correspondência das características semanticamente semelhantes.

Um novo caso é guardado procurando um caso semelhante e estabelecendo os índices das características relevantes. Se o caso encontrado tiver apenas diferenças subtis relativamente ao novo caso, o novo caso pode não ser armazenado ou os dois casos podem ser fundidos num só, através da generalização de algumas características.

Quase todos os sistemas CBR existentes usam organizações de memória inspiradas no modelo de memória dinâmica de *Schank*, no modelo exemplar-categoria de *Porter*, ou numa combinação entre as duas abordagens. É o que se passa como o sistema BOLERO que usa os episódios genéricos do modelo de memória dinâmica conjuntamente com ligações para exemplares e ligações diferenciais do modelo exemplar-categoria.

A própria estrutura dos casos é também uma característica importante. Enquanto a maioria dos sistemas CBR guarda cada caso como uma unidade, outros há que os guardam em pedaços juntamente com ponteiros para posterior reconstrução. A grande vantagem da primeira abordagem é que guardando todo o caso numa só localização o podemos recolher e usar na resolução de um novo problema de uma só vez. A desvantagem é que é difícil criar soluções baseadas em partes de diversos casos. Para o conseguir é necessário percorrer a memória de casos e ir recolhendo os "pedaços" apropriados dos diversos casos. São exemplos de sistemas que usam casos unitários o CASEY, o CHEF e o HYPO. A segunda abordagem (casos divididos em pedaços) é mais vantajosa quando é necessário criar soluções baseadas em soluções parciais de diversos casos, uma vez que é mais fácil identificar e aceder às partes necessárias.

3.5 Estruturas de Memória e Algoritmos de Pesquisa

Apresentam-se nesta secção as várias estruturas para armazenamento de casos, os algoritmos de pesquisa associados e descrevem-se as principais vantagens e desvantagens de cada uma das abordagens.

Uma base de casos pode ser vista como um tipo especial de base de casos e, tal como numa base de dados, também armazena um grande número de registos. Quando a base de casos é interrogada, os algoritmos de pesquisa a que recorre deve seleccionar os registos apropriados e, tal como numa base de dados, a pesquisa deve ser eficiente.

O processo de pesquisa numa base de casos pode ser visto como um problema de procura massiva, mas com uma importante diferença relativamente às bases de dados: Numa base de casos não é esperado que haja uma correspondência perfeita entre a nova situação e um caso. O processo de procura deve resultar na selecção de um caso para o qual ocorra uma adequação parcial tão extensa quanto possível. Dado que os algoritmos de adequação parcial são computacionalmente pesados, o processo de selecção deve ser circunscrito aos casos com alguma relevância potencial à nova situação. Assim, em geral, os algoritmos de pesquisa em bases de dados não são aplicáveis na pesquisa de bases de casos. A comunidade científica ligada ao raciocínio baseado em casos teve de desenvolver os seus próprios algoritmos de pesquisa.

Em geral, a eficiência dos mecanismos de pesquisa depende da quantidade e complexidade dos processos de adequação (semelhanças) envolvidos. Já a precisão da pesquisa vai depender essencialmente dos índices disponíveis. As estruturas de memória que têm sido consideradas dão normalmente ênfase à criação de partições nos casos com vista a tornar o processo de recolha eficiente e, simultaneamente, preciso. O objectivo das estratégias de pesquisa é seleccionar um pequeno conjunto de casos potencialmente relevantes e garantir que pelo menos alguns dos casos mais úteis à resolução do novo problema se encontram nesse conjunto. A estrutura organizacional, bem como os algoritmos de pesquisa mais adequado numa situação particular, depende largamente do número de casos que compõem a base de casos, da complexidade dos índices e das tarefas a que a base de casos deve dar suporte. Nas subsecções seguintes são descritas as estruturas de memória e algoritmos de pesquisa actualmente mais comuns no raciocínio baseado em casos:

- Memória plana

Pesquisa em série

- Memória plana
  Pesquisa em paralelo
- Memória hierárquica, árvores de descritores partilhados
  Pesquisa em árvores primeiro em largura
- Memória hierárquica, árvores de discriminação
  Pesquisa em árvores primeiro em profundidade
- Memória hierárquica, árvores de discriminação com redundância
  Pesquisa em árvores primeiro em largura
- Memória hierárquica
  Pesquisa em paralelo

A seguir descrevem-se estas estruturas e algoritmos, das mais simples para as mais complexas.

3.5.1 Memória plana com pesquisa em série

Numa memória plana os casos são simplesmente guardados numa lista. A recolha é feita aplicando sequencialmente uma função de semelhança a cada um dos casos e guardando a informação respeitante ao grau de semelhança de cada episódio relativamente à nova situação/problema. Os casos que apresentam uma maior semelhança são recolhidos.

Nestas estruturas não existe qualquer tipo de organização acima dos casos e o algoritmo de recolha é muito simples. De facto, o trabalho de escolha de casos está todo centrado nas heurísticas de adequação parcial.

A principal vantagem desta abordagem é que toda a base de casos é pesquisada. Em resultado disso a precisão do processo de recolha depende unicamente da de cálculo da adequação parcial ou semelhança. Se as heurísticas de adequação estão correctas, então o caso com maior semelhança será sempre recolhido. Outra vantagem está em que o processo de adicionar novos casos à base de casos é trivial.

A maior desvantagem desta abordagem é ser dispendiosa, dado que o tempo necessário à recolha de casos cresce linearmente com o aumento do número de casos armazenados. Assim, este esquema não é adequado para sistemas que possam vir a conter um elevado número de casos.

3.5.2 Memória hierárquica e árvores de descritores partilhados

Quando existe uma base de casos com um número elevado de episódios é importante organizá-los numa estrutura hierárquica por forma a que apenas um pequeno subconjunto tenha de ser considerado no processo de recolha e selecção. Esse subconjunto terá, no entanto, de reunir boas hipóteses de conter o caso com a maior semelhança ou os casos potencialmente mais úteis à resolução do novo problema.

Foram desenvolvidos vários métodos de agrupamento indutivo que podem ser aplicados na organização dos casos em memória. A razão de base para a aplicação estas técnicas indutivas é que se os casos forem agrupados de acordo com as semelhanças entre eles e se for seleccionado o grupo de casos que têm uma maior semelhança com o problema actual, então só os elementos desse grupo têm de ser analisados para decidir qual o caso mais útil à resolução do novo problema. A divisão dos grupos em outros mais pequenos dá então lugar a uma hierarquia de casos.

O recurso a árvores de descritores partilhados é uma forma de agrupar casos que partilham vários descritores. Cada nó de uma árvore deste tipo é etiquetado pelos descritores partilhados pelos casos associados aos nós descendentes desse nó. Itens que não partilham esses descritores estão associados a nós irmãos ou descendentes dos irmãos desse nó. Os nós que não têm

descendentes comportam os próprios casos. A Figura 18 mostra uma árvore de descritores partilhados para casos sobre mediação no sistema MEDIATOR.

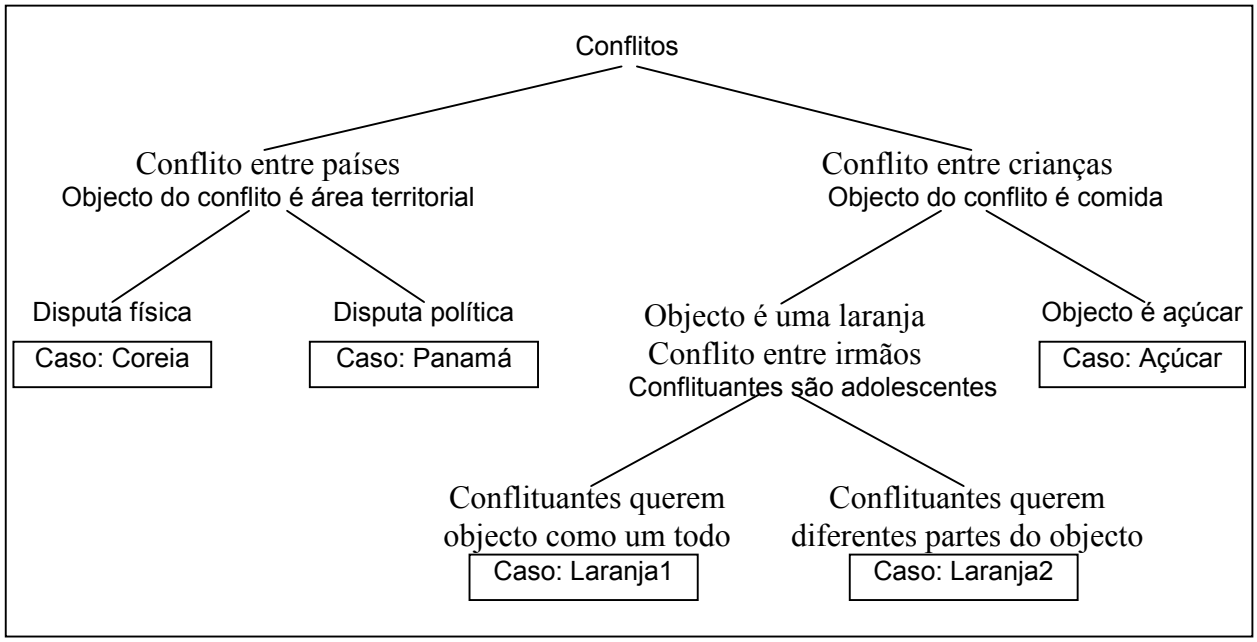

Figura 18 - Uma árvore de descritores partilhados no sistema MEDIATOR.

Nas árvores de descritores partilhados a organização que é dada à árvore determina os casos que são recolhidos. Assim, se os nós mais altos na hierarquia correspondem aos descritores mais importantes, então esses mesmos descritores serão considerados primeiro e grupos de casos que comportem esses descritores são considerados prioritariamente. Se, pelo contrário, descritores secundários aparecem no topo da hierarquia, então haverá fortes hipóteses de que casos que partilham descritores importantes com a nova situação não cheguem a ser recolhidos.

A principal vantagem das árvores de descritores partilhados é tornar a recolha de casos mais eficiente do que quando estes estão organizados numa lista. Em vez de ter de verificar se existe adequação ou semelhança para todos os casos armazenados, consideram-se apenas subconjuntos de casos.

Este método tem, no entanto, algumas desvantagens. A inclusão de novos casos na estrutura requer algum esforço computacional. Enquanto numa memória plana os casos são simplesmente acrescentados a uma lista, numa árvore de descritores partilhados os casos têm de ser colocados na posição correcta da árvore. Para além disso, a própria árvore ocupa espaço em memória. No entanto, os custos adicionais são compensados pelas vantagens obtidas em termos de eficiência da recolha e selecção de casos. Para além disso, os algoritmos de construção e manutenção da árvore não são complexos e por sua vez o espaço de memória é um recurso cada vez mais acessível.

3.5.3 Memória hierárquica e árvores de discriminação

As árvores de descritores partilhados agrupam casos semelhantes e como efeito secundário descriminam casos. Um esquema de organização alternativo que simultaneamente agrupa e diferencia são as árvores de discriminação. A principal diferença entre árvores de discriminação e as de descritores partilhados está nas prioridades postas no agrupamento e diferenciação, já que nas árvores de discriminação á a diferenciação que aparece como efeito principal, enquanto o agrupamento é um efeito secundário. Nas árvores de discriminação cada nó é constituído por uma interrogação que vai subdividir o conjunto de itens no nível seguinte da árvore. Assim, cada

nó filho de um nó interrogação representa uma resposta distinta para a questão posta. A figura 19 mostra uma árvore de discriminação para casos de mediação no MEDIATOR.

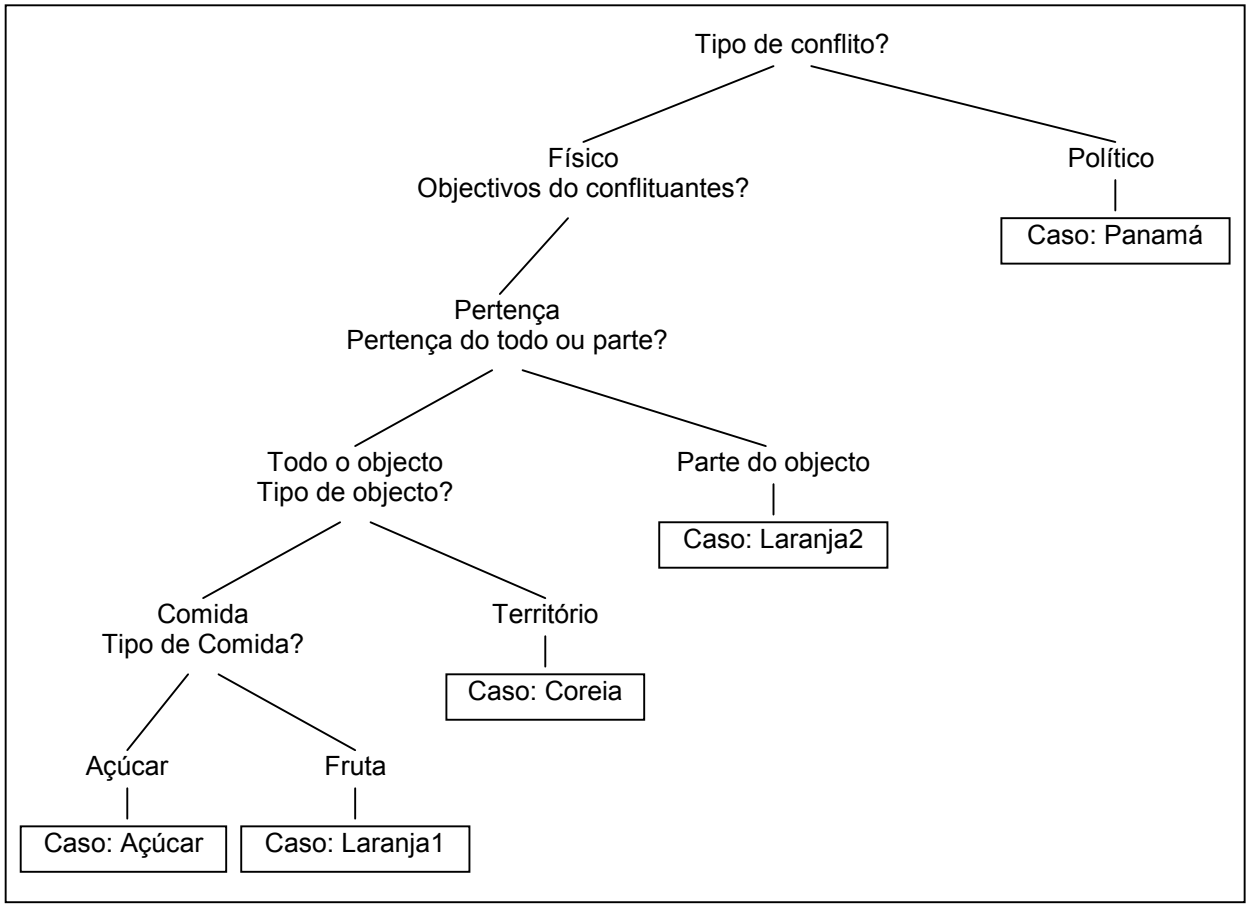

Figura 19 - Uma árvore de discriminação no sistema MEDIATOR.

Nas árvores de discriminação (tal como nas de descritores partilhados) é importante considerar descritores ao longo das dimensões mais importantes antes dos respeitantes a dimensões menos relevantes, de modo a assegurar que os casos para os quais ocorre semelhança ao longo das dimensões mais relevantes são seleccionados. Isto é conseguido nas árvores de discriminação colocando as questões mais importantes no topo da hierarquia.

As árvores de discriminação apresentam muitas das vantagens e desvantagens encontradas nas de descritores partilhados. Subdividem o conjunto de casos, tornando a pesquisa mais eficiente que numa lista não organizada. Tal como nas árvores de descritores partilhados, as próprias árvores ocupam espaço de memória. Apesar do cuidado posto na forma como é organizada a estrutura hierárquica, de modo a que os casos potencialmente mais úteis sejam recolhidos, não é possível evitar que nesta estrutura, tal como nas árvores de descritores partilhados, alguns casos com interesse não fiquem fora do processo e recolha.

As árvores de discriminação apresentam, no entanto, algumas vantagens relativamente às de descritores partilhados. Uma delas é a eficiência. A colocação de questões isoladas relativamente à nova situação e a travessia dos arcos correspondentes às respostas pode ser implementada de forma mais eficiente do que aplicar processos de adequação ou semelhança aos nós da árvore como é feito para atravessar uma árvore de descritores partilhados.

Outras vantagens que as árvores de discriminação apresentam são de caracter conceptual. Primeiro, é de fácil compreensão a relação existente entre os índices e a organização de casos numa árvore de discriminação. Existe uma correspondência directa entre os atributos dos índices e as questões associadas aos nós da árvore, bem como entre os valores assumidos por esses

índices nos casos e as respostas a essas questões. De facto, as questões com as respostas associadas são nem mais nem menos que os índices dos casos.

Uma segunda vantagem diz respeito à separação conceptual dos vários atributos e valores. Quando os vários atributos estão separados torna-se mais fácil determinar que atributos foram úteis na recolha de um caso. Nas árvores de descritores partilhados a ligação entre atributos e grupos de casos é mais difusa, já que nesta cada nó é etiquetado não por um descritor, mas sim pelo conjunto de descritores partilhados pelos casos que lhe estão associados.

Por outro lado, as árvores de descritores partilhados têm uma vantagem sobre as árvores de discriminação, que é o facto de nas árvores de discriminação um erro na selecção de um nó no topo da hierarquia levar o sistema de raciocínio a não ter mais a possibilidade de chegar ao nó associado ao caso que deveria ser recolhido. Nas árvores de descritores partilhados é mais difícil ocorrer esta situação, já que a escolha de um nó para prosseguir a pesquisa na árvore é determinada por um conjunto de descritores e não por um único item.

Uma desvantagem comum às árvores de discriminação e às de descritores partilhados é o facto de que se a descrição do novo problema for incompleta, ou seja, se um ou mais descritores estiver omisso não é possível prosseguir a navegação nos nós da estrutura. Assim, para atravessar uma árvore de discriminação é necessário ter respostas (ou forma de as calcular) para todas as questões encontradas nos nós que compõem o percurso na árvore.

Quando ocorrem omissões há três opções possíveis: terminar a pesquisa, continuar a pesquisa a partir de todos os descendentes do nó em que esta foi interrompida e seleccionar o caminho na estrutura que se apresenta como mais plausível na condução ao sucesso da pesquisa. Qualquer das três opções é problemática. Por um lado, terminar prematuramente a pesquisa porque falta um dado na informação é indesejável, já que o facto da descrição do problema estar incompleta não deve impedir a recolha de casos para os quais existe uma adequação parcial.

Continuar a pesquisa ao longo dos caminhos com origem nos descendentes do nó em que esta foi interrompida é indesejável por razões diferentes. Quando é permitido que a pesquisa prossiga ao longo de todos os descendentes, a capacidade discriminatória da árvore é reduzida.

Escolher o percurso alternativo mais plausível é talvez a opção mais satisfatória, mas requer um esforço de computação adicional na inspecção dos níveis inferiores da árvore ou, em alternativa, conhecimento adicional que permita decidir sobre que percurso seguir sem ter de inspeccionar a árvore.

3.5.4 Memória hierárquica e árvores de discriminação com redundância

As árvores de discriminação com redundância constituem uma possível solução para o problema da incompletude de informação nas árvores de descritores partilhados e de discriminação. Numa estrutura deste tipo os descritores são organizados em várias árvores de discriminação, cada uma com um ordenamento das perguntas diferente. As várias árvores são pesquisadas em paralelo. Se numa das árvores não é encontrada a resposta para uma questão, então a pesquisa nessa árvore é interrompida, continuando nas outras árvores. Em resultado da redundância existente, é normal que em pelo menos uma das árvores seja encontrado um caso útil à resolução do novo problema.

Com vista a reduzir o peso computacional destas estruturas, é comum combiná-las com árvores de descritores partilhados. Tal como nas árvores de discriminação, cada nó interno comporta uma resposta à questão associada ao nó de que descende. Os nós que têm associadas questões têm ainda um conjunto de descritores que são comuns aos casos que estão nas folhas da subárvore definida por esse nó. A redundância é então mantida sob controlo eliminando de consideração questões que não digam respeito ao grupo de casos definido pelo conjunto de descritores associado ao nó. A figura 20 mostra o resultado da combinação de uma árvore de discriminação com redundância com uma árvore de descritores partilhados no sistema CYRUS.

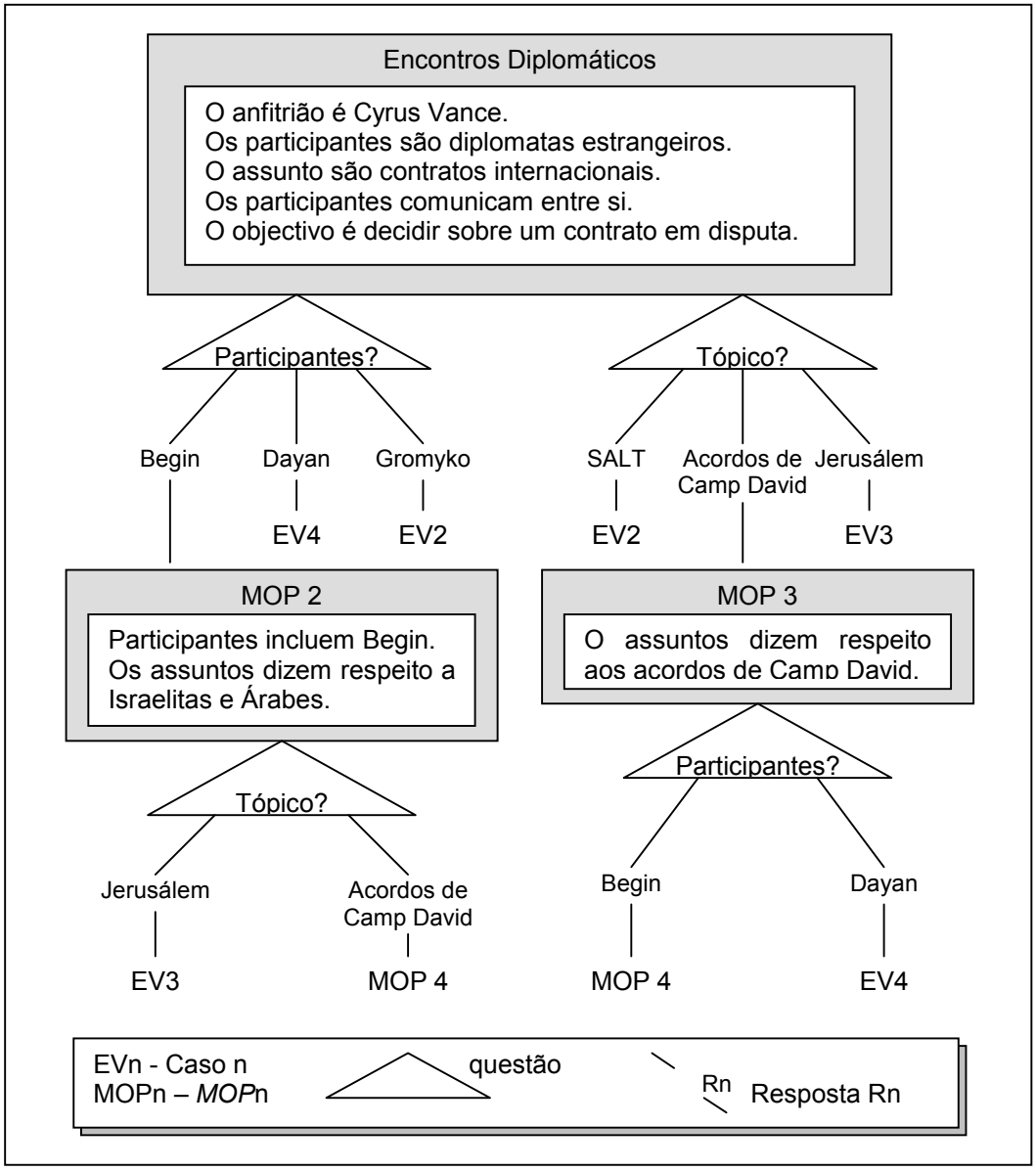

Figura 20 - Uma árvore de discriminação com redundância no sistema MEDIATOR.

As árvores de discriminação redundantes compreendem algumas vantagens para além daquelas referidas para as árvores de discriminação simples. Em primeiro lugar, como foi já referido, a questão de ausência de informação sobre algumas dimensões do problema é, em princípio, contornada. Segundo, o facto dos casos serem indexados de vários modos cria percursos independentes autónomos para chegar a um caso, minimizando os problemas decorrentes de uma possível organização incorrecta da árvore que levasse a considerar em primeiro lugar descritores menos relevantes.

Como desvantagens aparecem os factos destas estruturas requererem uma grande quantidade de espaço de memória adicional e de ser computacionalmente pesado modificar a árvore quando são introduzidos novos itens. Finalmente, o algoritmo de recolha pode fornecer uma grande quantidade de casos com fraca adequação (semelhança diminuta) que necessitam de passar por um segundo processo de adequação aplicado a um número considerável de casos.

Vários esquemas de organização dos casos em memória foram entretanto criados no sentido de desenvolver os princípios em que se baseiam as árvores de discriminação. O objectivo comum às diferentes abordagens é tornar o processo de recolha mais eficaz e os mecanismos de adequação parcial mais flexíveis, mantendo as vantagens da indexação redundante.

3.5.5 Memória plana com pesquisa em paralelo

Até aqui falámos de métodos de pesquisa em série em que o objectivo é organizar um conjunto de casos de modo a que um processador sequencial possa seleccionar os casos com maior semelhança. Quando se pretende introduzir processamento paralelo há que considerar novas questões.

À primeira vista, poderá parecer que com processamento paralelo os esquemas de indexação e de organização da memória apresentados passam a ser desnecessários. De facto, em máquinas massivamente paralelas os casos podem ser ordenados aplicando funções de semelhança ou adequação a todos eles em simultâneo. No entanto, se o problema de indexação for visto como um processo cujo objectivo é predizer em que condições um determinado caso deve ser seleccionado, então o problema da indexação mantém-se, passando apenas para o lado da função de adequação ou semelhança. Ou seja, a utilização de máquinas paralelas torna desnecessários os índices para direccionar a pesquisa, mas mantém a necessidade de definir quais os descritores que devem ter prioridade no processo de adequação parcial e no ordenamento dos casos. Assim, a adequação entre determinados conjuntos de descritores de um caso e do novo problema deve conferir uma posição mais alta a esse caso no processo de ordenamento do que quando se verifica sobre outros conjuntos de descritores.

Uma forma de memorizar casos numa máquina paralela SIMD (*single instruction, multiple data*) é guardá-los como vectores de descritores e colocar cada vector de descritores num processador. A recolha de casos é feita por adequação entre a nova situação e todos os vectores de descritores em paralelo.

As vantagens inerentes ao uso de máquinas paralelas para selecção de casos são óbvias: o algoritmo de selecção é simples, a adequação e recolha são realizadas num só passo e a inserção de novos episódios na base de casos é trivial, já que não existe nenhuma estrutura de organização dos casos.

Como desvantagens deste modelo surge o preço bastante elevado do *hardware* necessário. Para além disso, funções simples de semelhança, que são as mais adequadas para máquinas SIMD, limitam a sensibilidade da adequação ao contexto.

3.5.6 Memória hierárquica e pesquisa em paralelo

O interesse em ter uma memória plana advém de não ser necessário manter uma estrutura de organização dos casos. No entanto, a ausência desta estrutura leva a que não existam ligações entre os episódios em memória, acontecendo, no entanto, que essas ligações se afiguram úteis em vários sistemas de raciocínio baseado em casos. As árvores de descritores partilhados e de discriminação fornecem um meio de representar o que há de comum entre os casos. Em geral, as estruturas hierárquicas fornecem um meio simples de especificar conhecimento normativo e de organizar processos de inferência. Seria assim interessante conseguir juntar as vantagens decorrentes de ter uma memória hierárquica e um algoritmo de recolha de casos para máquinas paralelas.

As árvores de discriminação com redundância são uma das estruturas candidatas à pesquisa por máquinas paralelas do tipo MIMD (*multiple instruction, multiple data*) em que cada processador fica encarregue de pesquisar uma das árvores que compõem a estrutura. Também estruturas resultantes da combinação de árvores de discriminação e de descritores partilhados como as que são utilizados no programa CYRUS podem ser pesquisadas por máquinas do tipo MIMD de forma a tornar a recolha de casos mais eficiente. Numa estrutura deste tipo um processador encarrega-se de processar a questão associada a um nó enquanto outros processadores se encarregam da adequação relativa ao conjunto de descritores associados ao nó.

As vantagens de aplicar pesquisa paralela sobre estruturas hierárquicas constituem um somatório das vantagens de ter estruturas hierárquicas e algoritmos de pesquisa para máquinas paralelas. Os casos, tal como as suas generalizações, são mantidos em memória e são acessíveis de forma idêntica à que ocorre quando temos estruturas hierárquicas sobre máquinas sequenciais.

No que respeita às desvantagens, estas são mais difíceis de enumerar já que poucas implementações incorporando mecanismos de recolha e selecção com estas características estão actualmente disponíveis. Uma das desvantagens que é evidente e que é herdada das árvores de discriminação com redundância é que do processo de recolha pode resultar um grande número de casos para os quais ocorre uma adequação fraca e que necessitam de passar por um segundo processo de adequação aplicado a um número considerável de episódios.

3.6 Adequação Parcial e Ordenamento

A escolha do caso mais útil à resolução de um novo problema é, em primeiro lugar, um processo de adequação parcial. O processo inicia-se com a recolha de casos para os quais se verifica um determinado grau mínimo de semelhança relativamente à nova situação. Este grau de adequação ou semelhança é calculado por uma função de avaliação, segundo determinadas dimensões representadas como índices. Com base nos resultados da função de cálculo do grau de semelhança é recolhido um conjunto de casos. Sobre esse conjunto é feita uma avaliação mais elaborada do grau de semelhança, tomando agora em conta a importância de cada dimensão na utilidade do caso. Este segundo processo será aqui entendido como um processo de *ordenamento de casos*. Desse ordenamento resulta a selecção de um número restrito de casos.

A importância atribuída a cada dimensão da representação de um caso é função, entre outras coisas, do objectivo do processo de recolha e selecção, sendo que o processo de ordenamento tem em vista seriar os casos pela sua utilidade em relação aos objectivos do sistema de raciocínio.

Antes de entrar em maiores detalhes sobre os processos de adequação e de ordenamento vamos começar por introduzir novos conceitos.

Três conceitos que são muitas vezes usados com significado similar são os de *característica*, *descritor* e *dimensão*. Consideremos de forma mais precisa estes conceitos. Um *descritor* de um caso é um par atributo/valor usado na sua representação. Os descritores são usados na descrição de aspectos do problema ou situação, da solução e da avaliação. Podem representar características concretas, abstractas ou estruturais, ou ainda relações entre características. *Dimensão* refere-se ao atributo de um descritor. Quando comparamos dois casos ao longo das suas dimensões estamos a extrair descritores correspondentes nos dois casos e a comparar os seus valores. Quando nos referimos ao valor ao longo de uma dimensão ou ao valor de uma característica estamos a considerar o valor dado a um atributo. O termo *característica* aparece na literatura umas vezes com o significado de *dimensão* e outras no sentido de *descritor*. Consideramos aqui o termo característica como equivalente a descritor.

Os procedimentos de adequação podem actuar ao nível das dimensões dos casos ou ao nível dos próprios casos. Na primeira situação diz-se que temos *adequação dimensional*, no segundo temos *adequação agregada*.

O processo de adequação parcial é o processo de comparação entre dois casos ou entre um caso e uma situação e resulta na atribuição de um grau de semelhança. O processo de ordenamento resulta na seriação dos casos de acordo com a extensão da semelhança e com a sua potencial utilidade na resolução de um novo problema. O processo de adequação pode fornecer um valor que representa o grau de semelhança ou simplesmente determinar se existe ou não uma adequação suficiente. No processo de ordenamento são determinados quais dos casos em que há uma adequação parcial são mais promissores na resolução do novo problema. É comum o procedimento de ordenação recorrer ao resultado fornecido pelo processo de adequação para seriar os casos de acordo com a sua utilidade.

Os critérios de adequação ou semelhança determinam a importância relativa das diferentes dimensões de representação no processo de adequação. Em geral, o grau de importância atribuído a cada dimensão é dependente do contexto.

Os critérios de adequação podem ser definidos *globalmente*, ou seja, idênticos para todos os casos da base de casos, ou definidos *localmente*, quando o são para cada caso ou pequenos conjuntos de casos.

Quando os critérios de adequação são definidos globalmente, o número de critérios considerado é reduzido e estes são pouco sensíveis ao contexto. Esta opção é em geral seguida quando os casos são usados para resolver um único tipo de problemas ou quando a relevância das várias dimensões pouco depende dos objectivos da utilização do caso. Uma maior sensibilidade ao contexto é obtida quando os critérios de adequação são definidos localmente, variando esses critérios em função de dois aspectos considerados no contexto (objectivos e importância no passado das características consideradas).

Os critérios globais de adequação resultam num *esquema estático de adequação*, ou seja, não há variantes nas funções de adequação utilizadas. Quando critérios sensíveis ao contexto são introduzidos, resultando portanto em critérios de adequação local, temos um *processo dinâmico de adequação*.

É comum os critérios de adequação estarem incluídos na função que calcula o grau de semelhança entre dois casos. Designaremos esta função por *função de avaliação*. É comum a função de avaliação recorrer a um critério de adequação que é o dos *valores de relevância dimensional* para calcular o grau de semelhança entre a nova situação e um caso para o qual ocorre adequação parcial. Como já foi referido, é frequente os processos de adequação devolverem a determinação de um grau de adequação entre dois casos sob a forma de um valor numérico. O valor do grau de adequação, além de ser obtido com base numa adequação dimensional, pode ainda ser *absoluto* ou *relativo*. Um valor absoluto de adequação é calculado independentemente de outros casos. Estes valores absolutos são em regra fornecidos por uma função numérica. Podem, no entanto, ser qualitativo, sendo derivados por processos de raciocínio qualitativo.

O cálculo de um valor relativo de adequação requer a comparação com outros casos. Enquanto os procedimentos de atribuição de valores absolutos calculam um valor que descreve o grau de adequação entre uma nova situação e um caso, os de atribuição de valores relativos usam um critério de ordenamento relativamente à nova situação. Este critério, que define em que condições um caso é melhor que outros, fornece uma explicação para o ordenamento criado.

Na construção de um esquema de adequação parcial entre dois casos há que ter um método de reconhecimento de quais as características que devem ser consideradas correspondentes nos dois casos. É necessário definir um método de cálculo do grau de semelhança entre os descritores dos casos. Há ainda que determinar a relevância dos descritores envolvidos no processo, em função dos objectivos do sistema.

Nas secções seguintes serão descritos alguns métodos de determinação de correspondências entre descritores, apresentam-se métodos de cálculo do grau de adequação parcial entre descritores de casos. Descreve-se em seguida como são atribuídos pesos às dimensões dos descritores envolvidos na representação dos casos. Com estes elementos passa-se ao processo de ordenamento. São então apresentados métodos de ordenamento estático baseados em funções numéricas e métodos de ordenamento dinâmico baseados em múltiplos pesos atribuídos às dimensões consideradas, bem como outros baseados em heurísticas de preferência.

3.6.1 Detecção de correspondências

A detecção das correspondências entre os descritores de duas situações tem como objectivo determinar para que características da nova situação deve haver adequação em relação a características da situação em memória. Em termos gerais, deve haver adequação entre características que desempenham os mesmos papéis funcionais. A equivalência de funcionalidades pode ser determinada de várias formas:

- Detecção de dois valores que partilham uma mesma dimensão.

- Utilização de heurísticas de reconciliação.
- Aplicação de evidências sobre os papéis funcionais equivalentes dos descritores, fornecidas por um modelo causal.
- Detecção de que dois valores têm o mesmo papel estrutural nas duas representações.

Nas situações mais simples, e como primeiro passo, as correspondências podem ser encontradas por simples verificação das relações entre as representações de um caso em memória e de uma nova situação. Por exemplo, dois valores num conjunto de cláusulas predicativas podem desempenhar os mesmos papéis nas duas descrições.

No entanto, os valores representados em dois casos podem apresentar correspondências que não são detectáveis duma forma tão óbvia como a apresentada anteriormente. Algumas situações que tornam a detecção de correspondências mais complexa são:

- Número de características diferente nas duas representações. Uns casos podem ter um maior número de características que outros. Por exemplo, uma refeição tem um número variável de pratos ou diferentes doentes têm, provavelmente, um número diferente de sintomas.
- Profundidade de representação diversa em diferentes casos. Alguns casos podem ser representados ao nível de acções individuais enquanto outros a um nível mais abstracto. Por exemplo, uma refeição pode ser descrita ao nível dos pratos envolvidos ou da sequência de acções a empreender na sua elaboração.
- Casos representados a partir de diferentes pontos de vista. Uma refeição pode ser descrita por si só como um caso, ou pode ser apresentada do ponto de vista de um episódio sobre uma comemoração festiva.

O recurso a heurísticas de reconciliação é uma das formas de identificar correspondências não óbvias. Por exemplo, o sistema JULIA recorre a quatro heurísticas de reconciliação, usadas pela ordem que se segue:

- Correspondência entre características iguais.
- Correspondência entre características com o mesmo papel funcional.
- Correspondência entre características que satisfazem as mesmas restrições.
- Correspondência entre um conjunto de características e uma característica mais geral e vice-versa.

A justificação para a primeira das quatro heurísticas é evidente. As restantes procuram criar correspondências entre características para as quais essa correspondência não se apresenta como óbvia.

Quando está disponível um modelo causal, este pode fornecer ajuda adicional na distinção de quais as características da nova situação e do casos recolhido que desempenham os mesmos papéis funcionais. Esta possibilidade é particularmente importante em domínios em que a funcionalidade associada aos descritores aparece indiferenciada na representação ou ainda quando cada característica pode assumir diferentes papéis funcionais.

Quando os casos pertencem todos ao mesmo domínio, encontrar a correspondência entre os seus descritores apresenta dificuldades, como já foi evidenciado. Se os casos pertencem a diferentes domínios essas dificuldades são acrescidas. Um dos métodos aplicáveis quando temos casos de diferentes domínios é designado *mapeamento estrutural*.

No mapeamento estrutural ocorre a adequação entre dois descritores quando estes partilham uma mesma relação estrutural, independentemente da semântica que lhe está associada. O pressuposto para criar correspondências entre descritores por via do mapeamento estrutural é que características que têm o mesmo papel estrutural terão também a mesma funcionalidade.

Este método exibe algumas particularidades importantes. Uma é que a correspondência criada é baseada nas características estruturais dos descritores e não nas especificidades dos seus conteúdos. Este facto torna o método largamente aplicável entre domínios em que as correspondências não estão aprofundadamente estudadas. Segundo, em termos de sistemática, é intuitivamente interessante preferir correspondência a um nível mais profundo (estrutura em vez de conteúdo).

3.6.2 Cálculo do grau de adequação entre conjuntos de características

Conhecidas as correspondências entre as características de dois casos, pode então passar-se ao cálculo do grau dessa correspondência. Existem vários métodos para o cálculo do grau de semelhança entre dois valores:

- Comparação baseada na colocação numa hierarquia de abstracção.
- Cálculo da distância numa escala qualitativa.
- Cálculo da distância numa escala quantitativa.
- Comparação do grau em que os valores contribuem para a funcionalidade que lhes é atribuída.

Quando é usada uma hierarquia de abstracção para calcular o grau de semelhança, este é calculado em termos da abstracção comum mais específica relativamente aos dois valores considerados. Quanto mais a abstracção comum mais específica (ACME) estiver distanciada do topo da hierarquia, maior é a semelhança atribuída aos valores em estudo.

Quando se pretende calcular um grau relativo de adequação, um par de valores tem uma melhor adequação que outro se a ACME do primeiro par for mais específica do que a ACME do segundo par. Suponhamos, por exemplo, que temos *canário* e *gato* a desempenhar idênticas funções em dois casos e que queremos comparar estes valores com o valor *papagaio* no novo problema. A ACME para o par de valores *canário* e *papagaio* é *ave* (ver figura 21). Comparando *gato* com *papagaio* a ACME é *vertebrado*. Temos assim que a ACME *ave* para *canário* e *papagaio* é mais específica do que a ACME *vertebrado* para *gato* e *papagaio*. Logo *canário* e *papagaio* são reconhecidos como tendo um grau de semelhança superior a *gato* e *papagaio*.

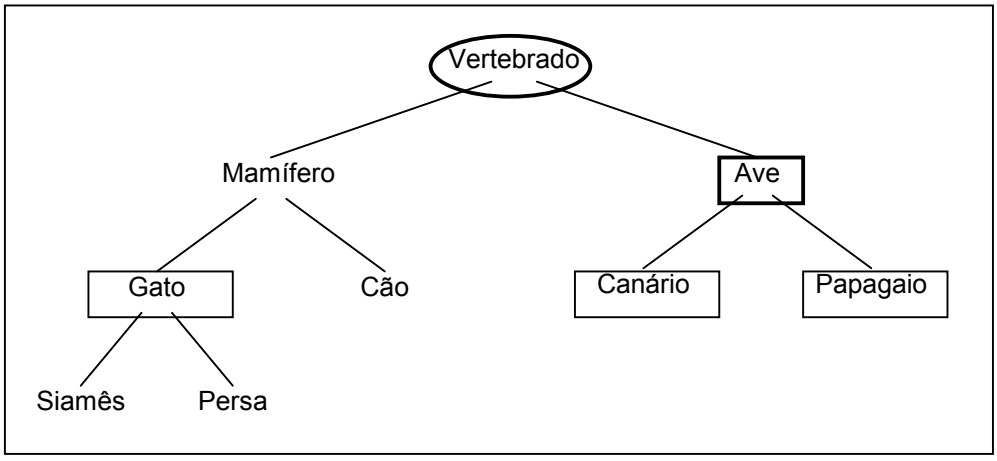

Figura 21 - Uma estrutura hierárquica de abstracção.

Outra forma de determinar o grau de semelhança entre dois descritores é medir a distância entre os valores numéricos dos dois descritores numa escala qualitativa. Se dois valores estão dentro da mesma região qualitativa, então são considerados iguais. Caso contrário, a distância entre as regiões qualitativas a que pertencem determina o grau de semelhança entre os dois valores.

A determinação qualitativa do grau de adequação pode não envolver a comparação de quaisquer valores qualitativos. Por exemplo, a motivação de um aluno pode ser elevada, moderada, reduzida ou nula. Cada um destes valores é um valor qualitativo numa escala qualitativa. A distância entre dois valores qualitativos numa escala qualitativa pode ser calculada contando o número de valores qualitativos entre os dois valores em comparação ou atribuindo a cada região qualitativa um valor numérico e comparando estes em termos quantitativos.

Quando os valores a comparar são valores numéricos podemos calcular a distância que os separa numa escala quantitativa. quanto maior a distância (normalizada) que os separa, menor o grau de semelhança entre eles.

No entanto, as técnicas até aqui apresentadas para a determinação do grau de adequação entre dois valores não são aplicáveis quando os itens a comparar não são qualitativa nem quantitativamente similares ou quando não há uma relação semântica evidente entre eles. Por exemplo, é necessário ter disponíveis técnicas que permitam determinar o grau de semelhança entre um caixote e dois tipos de cadeiras ou entre dois tipos de caixotes e uma cadeira.

O CASEY é um dos programas que lidam com este tipo de questões, assumindo que se, com base num modelo causal, é possível estabelecer uma relação causal entre dois itens qualitativamente diferentes então eles exibem um determinado grau de semelhança. O pressuposto que está por detrás desta asserção é de que se foi detectado, com base num modelo causal, que dois itens estão funcionalmente relacionados então é assumido que o grau de semelhança entre eles é função do comprimento da cadeia causal que estabelece essa relação funcional. Este princípio está na base da determinação do grau de adequação através do grau em que os dois valores contribuem para a funcionalidade que lhes é atribuída.

3.6.3 Atribuição de relevância às dimensões da representação

A qualidade dos resultados devolvidos pelas funções de cálculo do grau de adequação é limitada pelo conhecimento disponível sobre a relevância das dimensões consideradas no processo de adequação. A relevância atribuída a cada descritor determina o grau de atenção que vai ser dado à adequação desse descritor aquando da determinação da semelhança entre um caso em memória e uma nova situação.

A atribuição de relevância a um descritor pode ser feita dinâmica ou estaticamente, tomando ou não em conta aspectos do contexto em que a adequação tem lugar. A determinação do factor de relevância pode ser feita localmente, sobre um caso ou um pequeno grupo de casos, ou globalmente sobre todos os casos em memória. A atribuição local de relevância implica definir valores específicos de relevância para a mesma dimensão conforme o caso considerado.

Determinadas as correspondências entre os descritores de casos, calculado o grau de adequação entre descritores correspondentes e atribuída ou calculada a relevância de cada dimensão segundo a qual um caso é indexado, há então que ordenar os casos recolhidos segundo a sua potencial utilidade na resolução de um novo problema.

Nas subsecções seguintes são descritos métodos de ordenamento estático e dinâmico de casos.

3.6.4 Funções numéricas de ordenamento

Quando a um sistema de raciocínio baseado em casos cabe o cumprimento de um único tipo de objectivos, o cálculo do grau de adequação é, em geral, baseado numa função numérica de avaliação.

Normalmente em sistemas deste tipo, as correspondências entre dimensões da nova situação e de um caso em memória são encontradas por simples mapeamento entre os atributos dos descritores da situação e do caso.

Às dimensões dos índices dos casos em memória é associado um conjunto global de valores de relevância. No que respeita à adequação entre descritores correspondentes, o grau desta é calculado com base na relevância atribuída a cada dimensão, sendo calculado um valor de adequação agregada.

O processo de ordenamento faz então a seriação dos casos de acordo com os valores de adequação agregada para cada caso. Este método de adequação e ordenamento é normalmente designado por *adequação por maior proximidade* ou *vizinho mais próximo*.

O valor da adequação agregada é calculado aplicando uma função de avaliação composta por um somatório de termos, envolvendo o produto do valor de relevância de cada dimensão multiplicado pelo valor do grau de adequação entre descritores correspondentes. Com vista a normalizar o valor da adequação agregada, o resultado é dividido pela soma dos valores de relevância considerados.

Temos assim a seguinte função de avaliação:

$$S = \frac{\sum_{i=1}^{n} p_i * sem(d_i^N, d_i^R)}{\sum_{i=1}^{n} p_i}$$

em que p_i representa a relevância atribuída ou calculada para a dimensão i, *sem* é a função de cálculo do grau de adequação (semelhança) entre dois valores e d_i^N e d_i^R são, respectivamente, os valores dos descritores d_i na nova situação e no caso recolhido da memória.

A utilização de uma função numérica de avaliação deste tipo conduz a um processo muito simples de ordenamento de casos. O(s) caso(s) com o valor de adequação agregada mais elevado aparece(m) em primeiro lugar na seriação realizada.

3.6.5 Ordenamento dinâmico baseado em múltiplos valores de relevância

A forma mais simples de implementar um esquema de ordenamento sensível ao contexto é por recurso a vários conjuntos de critérios de adequação, associando a cada conjunto informação sobre as circunstâncias em que esse conjunto deve ser utilizado. Cada conjunto de critérios tem um âmbito de aplicação que pode abranger:

- Uma partição da base de casos.
- Os casos com o mesmo objectivo.
- Os casos com a mesma tarefa.

Quando a base de casos só é consultada com vista à concretização de um tipo de objectivos e a relevância das dimensões consideradas varia com o tipo da nova situação, as partições na base de casos podem ser organizadas em função do novo problema. Nestas circunstâncias é atribuído a cada partição um conjunto distinto de valores de relevância para as dimensões de representação dos casos.

3.6.6 Ordenamento dinâmico baseado em heurísticas de preferência

Um segundo modo de implementar o ordenamento dinâmico de casos baseia-se na utilização de um método de adequação dirigido ao ordenamento dos casos relativamente uns aos outros, em vez de calcular valores individuais para o grau de adequação. Nestas circunstâncias, uma forma de realizar o ordenamento é através da aplicação de um esquema de preferências. Num esquema de preferências vários modos de ordenamento são postos em confronto. As regras que compõem a estrutura de preferências têm o formato "sendo todo o resto igual, escolher os casos com as seguintes características…".

Num esquema com esta natureza, as heurísticas de preferência dão prioridade aos casos claramente orientados para os objectivos do sistema de raciocínio. Se não existem casos nesta situação, então são seleccionados aqueles em que há adequação relativamente a características

para as quais, não sendo evidente a relação com os objectivos perseguidos, há possibilidade desta existir. A implementação de um esquema de heurísticas de preferência põe duas questões:

- Que heurísticas de preferência considerar?
- Qual a ordem de aplicação das heurísticas?

Em geral, têm sido aplicadas três heurísticas pela ordem que a seguir se indica. Primeiro, os casos que sugerem melhores soluções. Ou seja, aqueles com mais hipóteses de terem a mesma solução que o novo problema ou que sejam facilmente adaptáveis à nova situação. Segundo, preferir aqueles casos que sugerem caminhos mais eficientes na direcção da solução. Terceiro, aqueles que são mais semelhantes à nova situação têm prioridade sobre outros com uma adequação agregada mais fraca.

3.7 Adaptação de Casos

Uma vez escolhido o caso ou casos que serão usados na construção de uma solução para a situação actual, e dado que frequentemente nenhuma situação anterior é exactamente igual à nova, o próximo passo é adaptar as soluções dos casos recolhidos ao caso actual. Se o problema actual é bastante semelhante a um que já foi anteriormente resolvido, então a solução anterior pode ser usada directamente e não é necessária nenhuma adaptação. Esta é, no entanto, uma ocorrência pouco usual, sendo necessárias estratégias de adaptação para tratar as situações mais frequentes em que a solução não pode ser usada sem alteração.

O processo de adaptação pode ter lugar durante a formulação da solução ou depois de haver *feedback* resultante da projecção dos resultados ou da aplicação dessa solução. Quando a adaptação surge em resultado da aplicação da solução toma então o nome de *correcção*.

As técnicas para adaptar casos variam com o tipo de tarefa efectuada pelo sistema CBR e com o domínio de aplicação do sistema. Os sistemas CBR para planeamento ou resolução de problemas têm, normalmente, critérios rigorosos que uma solução potencial tem de observar. Os sistemas cuja tarefa é o projecto ou explicação dão mais ênfase a soluções criativas, encorajando uma variedade de técnicas mais vasta que, embora possam fornecer muitas "más" soluções, podem fornecer algumas interessantes e criativas.

Outra tendência na investigação das estratégias de adaptação tem sido tentar descobrir regras mais gerais, independentes do domínio, para modificar os casos. Se estas regras puderem ser descobertas, podem ser adicionadas a qualquer sistema CBR, quer a sua tarefa seja planear receitas ou diagnosticar doenças de coração.

A adaptação de casos pode tomar várias formas como sejam: inclusão de novos elementos na anterior solução, eliminação de partes da solução, substituição de certos elementos por outros e transformação de parte da anterior solução.

Existem basicamente dois tipos de técnicas de adaptação no CBR:

- Adaptação estrutural, em que as regras de adaptação são aplicadas directamente às soluções guardadas nos casos. Este tipo de adaptação é usada no JUDGE e no CHEF.
- Adaptação derivacional que reusa os algoritmos, métodos ou regras que geraram a solução original para produzir uma nova solução para o problema actual. Neste método a sequência de planeamento que gerou a solução original tem de ser armazenada na base de casos conjuntamente com a solução, como no MEDIATOR. A adaptação derivacional só pode, no entanto, ser usada em casos bem compreendidos.

Um conjunto ideal de regras de adaptação deve ser suficientemente forte para gerar soluções completas a partir do início e um sistema CBR eficiente pode necessitar tanto de regras de adaptação estrutural para adaptar soluções mal entendidas como de mecanismos derivacionais para adaptar soluções de casos bem compreendidos.

Kolodner descreveu dez estratégias de adaptação, além da adaptação nula (ausência de adaptação), divididas em três grandes grupos: métodos de substituição, métodos de transformação e outros métodos:

Adaptação nula – é uma técnica simples que se aplica quando é recolhida uma solução para o problema actual sem ser adaptada. A adaptação nula é útil em problemas que envolvem raciocínios complexos mas soluções simples. Por exemplo, quando uma pessoa requer um empréstimo bancário, após responder a inúmeras questões a resposta final é bastante simples: conceder o crédito ou rejeitar o crédito.

- **Métodos de Substituição** – Consistem na troca de objectos ou valores da solução em memória por outros mais adequados à nova situação.

  Reinstanciação - significa remodelar a estrutura da solução anterior com novos argumentos, sendo usada para renovar as características de uma solução anterior com novas características. Consideremos, por exemplo, o CHEF, um sistema CBR que gera novas receitas através da recolha e adaptação de pratos anteriores. O CHEF usa esta técnica para gerar uma receita de frango e ervilhas a partir de uma receita de carne de vaca e brócolos. O frango substitui a carne de vaca na receita e as ervilhas substituem os brócolos.

  Ajuste de parâmetros - é um método baseado na aplicação de heurísticas para ajustar um parâmetro da solução do caso anterior com base nas diferenças entre as descrições do caso anterior e do actual. Esta é uma técnica de adaptação estrutural que compara parâmetros específicos do caso recolhido e do caso corrente para modificar a solução de um modo adequado. É usada, por exemplo, no JUDGE que recomenda uma pena menor quando um crime é menos violento.

  Pesquisa local – Consiste em pesquisar uma estrutura auxiliar de conhecimento com vista a encontrar um substituto de um valor da solução em memória. Esta pesquisa é efectuada na hierarquia semântica à procura de um substituto para algum objecto na solução anterior que seja inapropriado para a nova situação. Se numa ementa anterior se serviram laranjas à sobremesa, mas elas não estão actualmente disponíveis, a pesquisa local permite ao sistema procurar na hierarquia semântica por alimentos relativamente próximos das laranjas (por exemplo maçãs ou outra fruta) que possam ser servidos como substituição.

  Sondagem de memória – Consiste na pesquisa de estruturas auxiliares de conhecimento ou de casos em memória com vista à obtenção de uma descrição específica. É uma pesquisa mais ampla que a anterior, em estruturas de conhecimento auxiliares ou na memória de casos, à procura de uma dada descrição.

  Pesquisa dirigida – São também sondadas as estruturas auxiliares de conhecimento e casos em memória, mas aqui são usadas heurísticas de pesquisa especializadas para guiar o processo de procura. A pesquisa é efectuada directamente nas porções da base de conhecimento onde há maior probabilidade de encontrar uma substituição (determinada pelas heurísticas).

  Substituição baseada em casos - são usados outros episódios em memória para sugerir substituições no caso que serve de base à construção da nova solução. Por exemplo, um sistema para planear refeições após determinar que a *lasagna* que estava a planear para prato principal não é apropriado pode substituí-lo relembrando outras ementas italianas cujo prato principal tem *pasta*.

- **Métodos de Transformação** – Consistem na modificação ou transformação de um pedaço da solução anterior com vista a criar uma adequação à nova situação.

  Transformação baseada em senso comum - faz uso de heurísticas de senso comum para substituir, eliminar ou adicionar componentes a uma solução. Por exemplo, pode

usar-se uma heurística de senso comum para transformar a *lasagna* numa *lasagna* vegetariana através da eliminação da carne.

Correcção guiada por modelos - a transformação é orientada por um modelo causal. É o que no sistema CELIA, que é usado para diagnóstico, e no KRITIK usado no projecto de sistemas de manutenção física.

- **Outros Métodos**

Heurísticas de adaptação e correcção específica - são aplicadas na geração de adaptações específicas num determinado domínio. Estas heurísticas são indexadas pelas situações em que são aplicáveis. Normalmente aplicam-se em domínios não cobertos pelos outros métodos. Estas heurísticas são indexadas pelas situações em que são aplicáveis. O sistema de planeamento de ementas JULIA usa heurísticas de adaptação especiais para modificar a estrutura das suas soluções. Estas heurísticas podem também implementar estratégias para reparar soluções quando estas falharam. O CHEF usa heurísticas de reparação especiais com este objectivo. As heurísticas de adaptação especiais são frequentemente implementadas como críticos e controladas através de sistemas baseados em regras. A aplicação crítica permite implementar alguns dos tipos de adaptação listados atrás, como o ajuste de parâmetros, transformação de senso comum e reparação guiada por modelos.

Repetição* ou *analogia derivacional - é o processo de usar o método de obter uma solução anterior ou pedaço de solução para derivar uma solução para a nova situação, i.e., repete o método usado para obter a solução no caso anterior em vez de usar a própria solução. Por exemplo, um sistema de raciocínio que recorra a analogia derivacional para resolver um problema de probabilidades segue os mesmos passos que usou no passado para resolver um problema idêntico. Outro exemplo, um sistema de planeamento de refeições que sabe que anteriormente escolheu laranjas para a sobremesa por um processo de escolher uma fruta da época (inverno) pode adaptar a refeição anterior para uma refeição de verão usando o mesmo processo para escolher uma fruta apropriada para o verão.

3.7.1 Métodos de Substituição

Os processos de substituição envolvem a escolha e integração de um substituto numa solução antiga, em substituição de um elemento que não se aplica ao novo problema.

A substituição por reinstanciação é usada quando um caso em memória e um novo caso têm a mesma estrutura, mas em que vários papéis são preenchidos de forma diferente no caso anterior e no novo caso. Na reinstanciação, as diferentes componentes com papéis idênticos são substituídas no caso em memória pelas componentes correspondentes no novo problema. Por exemplo, quando o sistema CHEF cria uma receita de frango com ervilhas com base noutra que conhece de bife com bróculos, substitui o bife por frango e os bróculos por ervilhas através de reinstanciação. O papel da carne e dos vegetais desempenhado na receita em memória, respectivamente pelo bife e pelos bróculos, passa na nova receita a ser preenchido pelo frango e pelas ervilhas.

O ajuste de parâmetros é uma técnica de interpolação de valores numa nova solução, com base numa anterior. Ou seja, dado um caso em memória e um novo caso que difere num certo grau do caso anterior, a solução anterior é modificada numa extensão determinada pelo grau de dissemelhança entre os dois casos. Vários sistemas de raciocínio baseado em casos recorrem ao ajuste de parâmetros como método de adaptação. Em PERSUADER é aplicado o ajuste de parâmetros na adaptação de valores numéricos de um contrato antigo de modo a ajustá-los à nova situação. Por exemplo, na elaboração de um acordo laboral, o PERSUADER constrói o novo contrato ajustando o valor percentual dos aumentos salariais com base nas diferenças entre a evolução do custo de vida na localidade onde foi celebrado o contrato que tem em memória e o local do novo conflito laboral.

Na reinstanciação um conjunto completo de papéis muda de atribuição num caso. No entanto, por vezes, basta realizar substituições numa pequena parte de um episódio para o adequar ao novo problema. Designamos este método de substituição por pesquisa local. Por exemplo, numa refeição planeada pelo sistema JULIANA basta substituir *lasanha* por *lasanha de vegetais* para adaptar a refeição para comensais vegetarianos. Esta substituição é feita por procura numa estrutura hierárquica em que os dois pratos estão próximos (lasanha de vegetais aparece como uma instância directa de lasanha).

A pesquisa local é um meio conveniente de encontrar a substituição adequada se esta é facilmente detectada pesquisando a estrutura hierárquica de conhecimento a partir da componente que deve ser substituída.

Embora frequentemente seja possível criar estruturas que tornem a pesquisa local fácil, se as estruturas hierárquicas de memória são utilizadas com objectivos variados torna-se impraticável descrever todas as relações existentes para cada conceito relativamente aos restantes e assim a pesquisa local perde utilidade. Nestas circunstâncias a sondagem de memória pode ser útil.

Para ilustrar este facto consideremos ainda um exemplo do sistema JULIANA, em que este pretende planear uma refeição cujo prato principal é um *guisado*. JULIANA recorda então de uma refeição que tem como prato principal *guisado de frango*. Depois de ter sugerido uma refeição incluindo este prato, é-lhe indicado que serão convidados apreciadores de pratos de peixe. Entretanto a estrutura hierárquica de pratos principais contém *guisado de vegetais* e *guisado de frango* como subclasses da classe *guisado*. Para alguns dos convidados, guisado de frango é um prato que será certamente mal recebido. Aplicando pesquisa local é considerado guisado de vegetais que é um dos conceitos próximos em termos de hierarquia de pratos, um prato igualmente distante das preferências de alguns dos convidados. Quando a pesquisa local não resulta, é por vezes possível construir uma descrição parcial do item que conduziria a uma boa substituição e assim fazer uma procura directa desse item, ou seja, fazer sondagem da memória. Continuando com o exemplo do sistema JULIANA, este vai sondar a memória sobre um prato com peixe como ingrediente e que seja cozinhado como um guisado. Em resultado desta pesquisa directa da memória este vai devolver como sugestão uma caldeirada (que pode ser entendida como um guisado de peixe).

Na sondagem de memória, esta é questionada através de uma descrição parcial do que se pretende. Na pesquisa dirigida vai-se um passo mais à frente: são dadas à memória instruções sobre como encontrar um dado elemento.

Um exemplo do SWALE ilustra esta ideia. O SWALE procura explicação para a morte de Len Bias, um famoso e aparentemente saudável basquetebolista. O SWALE recorda o caso de Jim Fixx, que estava aparentemente de boa saúde e que morreu de ataque cardíaco após uma corrida de manutenção. Depois da sua morte os médicos descobriram-lhe uma malformação cardíaca. Assim, a explicação para a sua morte é que o esforço desenvolvido durante a corrida provocou uma sobrecarga cardíaca que foi a causa do ataque de coração.

Baseando-se no caso de Jim Fixx, o SWALE coloca a hipótese de que Len Bias tivesse uma malformação não detectada e que o esforço resultante da prática de corrida de manutenção tenha conduzido ao ataque cardíaco. No entanto, Len Bias não praticava corrida de manutenção. O SWALE detecta a contradição e procura substituir *corrida de manutenção* na explicação para Jim Foxx por algo mais apropriado para Len Bias e que tenha o mesmo papel da corrida de manutenção no resultante ataque cardíaco. Por outras palavras, o SWALE pergunta-se o que terá Len Bias feito que levasse ao ataque cardíaco?

Para responder a esta questão, o SWALE tenta encontrar uma acção que Len Bias realizasse frequentemente e que produzisse esforço cardíaco suplementar. Acontece que as actividades de Len Bias podem ser enumeradas de diferentes formas. Para colocar este processo de procura sob controlo, o SWALE recorre a uma heurística especializada de pesquisa. Neste caso a heurística seguida é procurar os *papéis temáticos* desempenhados por Len Bias na sua vida e para cada papel temático considerar acções específicas associadas a esse papel.

O papel de Len Bias como basquetebolista é o primeiro a ter tido em conta. Acontece que os jogadores de basquetebol praticam frequentemente corrida ao ar livre. Dado a corrida ao ar livre ser uma forma de exercício semelhante à corrida de manutenção, o SWALE identifica esta acção como potencial responsável por esforço cardíaco suplementar conducente a ataque cardíaco. Assim o SWALE propõe que o ataque de Len Bias foi provocado por este praticar corrida ao ar livre que por sua vez provocou um esforço cardíaco suplementar (num coração com uma malformação não identificada), causa do ataque cardíaco.

A pesquisa local é uma técnica apropriada para a selecção de um substituto, caso este exista na memória hierárquica que se encontre nas proximidades do item a substituir. Por sua vez, a pesquisa dirigida é adequada à procura de substituto em locais de memória bem definidos. No entanto, por vezes o melhor local para encontrar um substituto é num outro caso semelhante ao que está a ser adaptado – substituição baseada em casos.

Um exemplo retirado do sistema CLAVIER ilustra este método. Dado um conjunto de peças em material compósito para serem cozidas numa autoclave, o CLAVIER deve derivar um conjunto de configurações que levem a que todas as peças sejam cozidas uniformemente e no mesmo espaço de tempo. O CLAVIER seleccionou um caso que sugere uma disposição para as peças que têm maior urgência em serem processadas. Na configuração sugerida todas as peças, menos uma correspondem às peças que devem seguir com maior rapidez para tratamento. Assim, o CLAVIER deve procurar um substituto no conjunto de peças a serem tratadas para a posição na configuração que não pode ser ocupada pela peça inicialmente considerada. Com vista a decidir sobre um substituto, o CLAVIER procura na biblioteca bocados de casos similares aquele em que houve uma discrepância na peça considerada no caso de base e na lista de peças a serem incluídas na nova configuração. Seguidamente o CLAVIER examina as peças feitas pelos episódios recolhidos e escolhe aquele que contém um pedaço mais próximo da estrutura espacial local à peça na origem do problema. Temos assim que o CLAVIER começa por seleccionar um caso com uma composição de peças globalmente semelhante à do novo problema e depois, se há uma peça que não se ajusta à sugestão feita por esse caso, procura um com uma configuração local semelhante à posição onde é necessário fazer uma substituição.

Uma questão que se coloca na substituição baseada em casos é: porque não usar em primeiro lugar o caso que contém o substituto? A razão para não o fazer é que, embora esse caso possa fornecer uma boa sugestão quanto a um substituto, não é, em princípio, o caso que comporta a melhor semelhança global relativamente à nova situação e, consequentemente, não é provável que seja o caso que fornece uma melhor solução global.

3.7.2 Métodos de Transformação

Os métodos de substituição são adequados se já existe um substituto para um valor inadequado. Se esse substituto não existe, há que encontrar uma forma de transformar o caso de modo a adaptá-lo às restrições impostas pela nova situação. Quando a transformação é realizada em resultado do *feedback* da avaliação de uma determinada solução então toma a forma de correcção.

Para exemplificar a transformação baseada em senso comum, imaginemos uma receita de lasanha em que temos como restrição a não utilização simultânea de leite e carne. Assim, um destes elementos tem de ser eliminado e/ou substituído por outro. Dado que neste prato a carne é entendida como um ingrediente secundário (a lasanha pode ser recheada por outros ingrediente que não carne), uma possibilidade é eliminar a carne recorrendo à heurística de transformação *eliminar característica secundária*. Outra possibilidade é substituir a carne por algo que cumpra a mesma função (usando a heurística de transformação *substituir característica por outra com a mesma função*). Isto requer, primeiro a determinação da função da carne na lasanha, depois encontrar um substituto usando um método de substituição. Por exemplo, se olharmos para a carne como uma fonte de proteínas, esta pode ser substituída por peixe ou soja. Se olharmos para

a carne como um fornecedor de textura, podemos substituí-la, por exemplo, por espinafres ou beringelas.

Enquanto as transformações baseadas em senso comum têm origem em conhecimento de senso comum sobre regularidades no universo, a transformação e a correcção baseada em modelos é um método apoiado em conhecimento sobre ligações causais num sistema ou situação.

Dois programas que aplicam extensivamente a correcção baseada em modelos são o CASEY e o KRITIK. O CASEY usa correcção baseada em modelos em tarefas de diagnóstico para adaptar a explicação causal da patologia de um determinado paciente a uma nova situação. O KRITIK recorre à correcção com base em modelos em duas tarefas: *design* de novos sistemas mecânicos a partir de experiências anteriores e criação de explicações causais para o comportamento de um novo sistema a partir das explicações causais existentes para um sistema que conhece.

3.7.3 Outros Métodos

As metodologias descritas até aqui envolvem métodos de adaptação de aplicação geral. São métodos independentes do domínio, embora alguns deles apoiados em conhecimento específico no domínio de aplicação. São assim *métodos fracos* de adaptação.

Na área da inteligência artificial a experiência demonstrou que embora os métodos fracos sejam de aplicação geral, o recurso a técnicas mais específicas produz frequentemente melhores resultados e com maior eficiência. Esta noção aplica-se também nos processos de adaptação: as heurísticas específicas de adaptação e correcção, quando estão disponíveis, fornecem meios poderosos de condução do processo. As heurísticas de adaptação referem-se, tipicamente, a três tipos de adaptação:

- Adaptação específica no domínio.
- Modificação estrutural.
- Correcção.

A adaptação específica no domínio refere-se a regras de substituição e de transformação específicas de um domínio. Por exemplo, um cozinheiro sabe que a manteiga pode ser substituída por margarina. Esta é uma adaptação específica no domínio da culinária.

A modificação estrutural é outra função comummente desempenhada por heurísticas específicas. Por exemplo, o JULIA usa heurísticas específicas de adaptação na transformação de estruturas.

As heurísticas específicas de adaptação são ainda usadas na condução de processos de correcção. Por exemplo, quando provamos um cozinhado e detectamos que está ensosso sabemos então o que fazer para corrigir o tempero.

Se bem que muitas das estratégias e métodos descritos anteriormente possam ser aplicados nos processos de correcção, várias heurísticas são específicas do processo de correcção, não sendo utilizáveis anteriormente na adaptação. Em particular, as estratégias usadas no CHEF para corrigir planos falhados caem nesta categoria.

Qualquer dos métodos de adaptação até agora descritos altera uma solução anterior no sentido de a adequar à nova situação. No entanto, por vezes é mais adequado derivar a nova solução ou solução parcial recorrendo aos mesmos meios usados na construção da solução anterior. Um exemplo está na realização por parte de um aluno de um trabalho de casa de matemática ou física. O professor desenvolveu alguns exemplos no quadro durante a aula. Em casa, o aluno tenderá a aplicar os passos seguidos nos exemplos que foram apresentados na aula. Este processo é denominado reexecução derivacional e a criação de uma solução é orientada por um processo baseado em casos. Consideremos ainda o exemplo do programa JULIA no planeamento de uma refeição vegetariana para uma pessoa alérgica ao leite. Dado estarmos na Primavera e esta ser a estação em que é possível obter espargos frescos, o JULIA propõe um

prato de *soufflé* de espargos. Acontece que o *soufflé* leva queijo. O JULIA tem de resolver o problema de adaptar o *soufflé* de modo a cumprir as restrições da situação e recorda-se de uma refeição em que adaptou um prato à base de tomate com queijo como ingrediente secundário eliminando o queijo. O JULIA vai então repetir este processo de inferência.

A reexecução derivacional teve uma primeira implementação através do programa ARIES, um antecessor de PRODIGY/ANALOGY. O ARIES reexecuta todo o conjunto de passos de raciocínio que compõem um caso anterior para derivar a resposta para um novo problema. Este processo extensivo de reexecução derivacional é chamado *analogia derivacional*. A analogia derivacional é útil quando derivações intermédias devem ser executadas para construir a solução para o problema inicial e quando os resultados dessas derivações intermédias são dependentes de constantes no problema que são diferentes das existentes no caso que fornece os passos de derivação. Isto passa-se, por exemplo, na resolução de problemas de matemática ou em situações de planeamento.

3.7.4 Avaliação e Correcção

À geração de uma nova solução segue-se a sua avaliação. A avaliação é um processo interpretativo geralmente baseado em modelos ou em casos.

Na avaliação baseada em modelos é utilizado um modelo causal para simular o efeito da solução criada. O KRITIK é um programa que comporta avaliação baseada em modelos. Um exemplo deste processo ocorre quando o KRITIK pretende projectar um sistema de refrigeração de ácido sulfúrico. No passado tinha planeado um sistema idêntico para ácido nítrico, dispondo em memória das especificações para esse sistema. O KRITIK decide aplicar esse plano na construção do novo sistema e seguidamente, com vista a projectar os efeitos desse plano, cria o respectivo modelo causal. Usa um *plano de alteração do produto químico* para trocar ácido nítrico por ácido sulfúrico no modelo. O plano de alteração da substância acrescenta também ao modelo a especificação de que o ácido sulfúrico apresenta acidez elevada.

Tendo realizado estas alterações no modelo causal, o KRITIK executa o modelo para simular o comportamento de novo sistema. Daí conclui que o sistema projectado é inadequado, já que as alterações nas propriedades da substância a refrigerar colidem com as especificações dos tubos do sistema, preparados para substâncias de baixa acidez.

A avaliação de uma solução pode também ser baseada em casos. Nesta abordagem uma nova solução é confrontada com um conjunto de casos em memória. As interpretações sugeridas por esses casos são analisadas e comparadas com vista à avaliação da nova solução.

Por exemplo, o JULIA para avaliar o plano de uma refeição recorda um conjunto de casos com diferentes avaliações. Se uma avaliação que inviabiliza uma solução é aplicável ao novo caso, então o JULIA procura corrigir a nova solução. No processo de avaliação são recordados casos com soluções semelhantes à que é proposta, dando preferência àqueles em que a descrição do problema está mais próxima do novo problema.

Outra forma de avaliação baseada em casos compreende a recolha de casos envolvendo um problema semelhante ao actual, comparando a nova solução com a que foi criada para esses casos.

Tal como na derivação de soluções, o nível de conhecimento que está acessível para o processo de avaliação determina a sua automatização ou o recurso a um processo interactivo. A avaliação baseada em casos recorre a vários tipos de conhecimento:

- Lista de factores constituintes da solução que devem ser tidos em consideração na avaliação.
- Prioridade relativa desses factores.
- Extensão da influência desses factores na solução como um todo, bem como nas diversas componentes dessa solução.

- Aspectos que influenciam os valores desses factores e como os valores de uns factores influenciam outros factores.

Quando cada uma destas formas de conhecimento está acessível é então possível avançar para uma avaliação baseada em casos. No entanto, muito frequentemente é difícil definir com precisão estes tipos de conhecimento. Se bem que seja comum o sistema de raciocínio poder determinar em que medida cada factor tem um papel na solução, é muitas vezes mais fácil ou mais adequado o sistema adquirir esse conhecimento interactivamente. Particularmente quando a determinação da qualidade da solução assenta em valores estéticos ou julgamentos individuais, é importante a intervenção de um ser humano na avaliação das soluções.

Quando os problemas detectados na solução têm uma correspondência unívoca relativamente aos métodos de correcção (adaptação) disponíveis, o processo pode ser integralmente automatizado. Igualmente quando a tomada de decisões sobre que métodos utilizar é computacionalmente simples é desnecessário recorrer a um processo interactivo. É, no entanto, comum que a escolha do método de correcção seja uma decisão complexa. Quando a decisão sobre que método de correcção aplicar é complexa, mas a execução do método tem uma complexidade aceitável, pode ser deixado ao utilizador a decisão sobre que estratégia utilizar, encarregando-se o sistema do resto do trabalho. De forma semelhante, quando a tomada de decisão é computacionalmente simples, mas a execução do método não o é, então os papéis invertem-se. Quando a decisão e aplicação do método são computacionalmente difíceis, todo o processo deve ser interactivo, cabendo eventualmente ao sistema o papel de dar sugestões ao utilizador.

Como foi já referido, a correcção é geralmente baseada, tal como a avaliação, em modelos ou em casos. Também já foi descrito como a correcção baseada em modelos é realizada, tendo sido apresentado um exemplo com o programa KRITIK.

No que respeita à correcção baseada em casos, esta tem lugar, por exemplo, no PERSUADER na correcção de um impasse negocial com base no modo como corrigiu outros impasses no passado. Na correcção de uma solução é comum o sistema de raciocínio procurar encontrar em memória descrições de falhas idênticas ou, em alternativa, falhas semelhantes ou explicadas de forma idêntica. Seguidamente selecciona anteriores correcções e verifica se aparentam ser aplicáveis à situação corrente. Em caso afirmativo procura adaptar essas correcções ao novo caso.

3.8 Aprendizagem

A capacidade de aprendizagem é uma propriedade decorrente do funcionamento normal de um sistema de raciocínio baseado em casos.

Um sistema de raciocínio baseado em casos aprende, em primeiro lugar, em resultado das novas experiências que memoriza e da indexação desses episódios. O grau de aprendizagem está dependente da variedade de experiências com que o sistema contactou no passado, da capacidade de gerar bons índices para essas experiências e de conseguir explicar os sucessos e falhas porque passou. Quanto maiores forem as aptidões do sistema relativamente a estas tarefas mais efectivo este será enquanto sistema de aprendizagem.

Começando pela questão da variedade de experiências, temos que o sistema não aprende só pelo facto de repetir um grande número de vezes a mesma tarefa. Quanto mais diversificado for o leque de experiências porque passou no passado, maior o número de situações que o sistema passa a puder resolver. Por outro lado, um sistema com um conjunto diversificado de experiências, mas em que estas tenham sido pouco aprofundadas não será capaz de reconhecer aspectos subtis de uma nova situação.

Os casos que são guardados são tão úteis quanto o forem os índices que lhes estão associados. Em última análise, são os índices que permitem recordar um caso sob determinadas circunstâncias. No entanto, quando falamos de aprendizagem, não é só a aprendizagem de

índices que deve ser tida em conta, mas também a reindexação de casos é um tema importante. Quando um caso é seleccionado e se revela inadequado (porque não pode ser aplicado ou a sua aplicação redundou numa falha) isto leva à necessidade de reindexá-lo de modo a que não volte a ser seleccionado em situações idênticas. Inversamente, quando é indicado ao sistema de raciocínio que um determinado caso em memória deveria ter sido utilizado, esse episódio deve ser reindexado de modo a que em idênticas situações passe a ser seleccionado. Igualmente, quando a utilização de uma caso revela algo que não era previamente conhecido, este deve ser reindexado de forma a reflectir esse novo conhecimento. O PROTOS e o CELIA são exemplos de sistemas que fazem reindexação de casos.

A capacidade de explicar porque uma solução é bem sucedida é também importante para o sistema de raciocínio. As explicações sobre o sucesso de uma solução resultam na escolha de índices para um caso que reflectem o conjunto de situações em que esse episódio é útil. Esses índices permitem assim ao sistema reconhecer oportunidades de reutilização de soluções anteriores. Vários planificadores (por exemplo, TRUCKER, RUNNER e EXPEDITOR) centram-se na escolha dos índices necessários ao reconhecimento dessas oportunidades.

Quando é apresentada ao sistema uma situação fora do âmbito das experiências em memória e este é capaz de resolver o problema por adaptação, a memorização do novo caso permitir-lhe-á resolver, no futuro, problemas idênticos de forma mais eficiente. Se o sistema falhou na construção de uma nova solução, poderá ainda aprender autonomamente ou não a partir dessa falha. A possibilidade de aprender a partir de uma falha está dependente do *feedback* que é fornecido, dos mecanismos de que o sistema dispõe de explicação desse *feedback* e das capacidades que tem de correcção de uma solução que tenha falhado.

Sistemas com capacidade de explicar as suas falhas e de determinar como as poderia ter evitado, guardam em memória a nova situação, indexando-a de modo a reconhecer no futuro a eminência de uma falha idêntica, bem como indicações sobre como a evitar. Se, complementarmente, o sistema tiver conhecimento sobre como eliminar uma falha, esse conhecimento pode também ser associado à nova situação, com vista à indexação de métodos de correcção. Esta indexação leva a que estes métodos sejam aplicados na futura construção de soluções para problemas semelhantes.

Adicionalmente, quando os casos de falha, armazenados na base de casos, são apropriadamente indexados, estes são úteis ao sistema na determinação das componentes do problema que devem ser tidas em conta no sentido de evitar a repetição da falha. Por exemplo, consideremos um sistema de raciocínio que é confrontado com uma situação que difere em aspectos subtis de outra em memória. O sistema sugere como solução para o novo problema aquela que tem para um problema semelhante e esta é classificada como incorrecta. Se o sistema conseguir explicar a falha, então esta ocorrência vai, futuramente, melhorar a precisão do raciocínio, dado que os casos de falha que vai guardar vão levá-lo a ter em consideração pequenas diferenças que eram anteriormente desprezadas. O programa PROTOS, ao recorrer às ligações etiquetadas pelas diferenças permite este tipo de raciocínio. O MEDIATOR e o CHEF ao guardarem os casos de falha juntamente com as correcções a fazer no sentido de evitar essas falhas têm também em consideração as componentes de uma situação que devem ser tomadas em conta com vista evitar futuros insucessos.

Para além de ocorrer a aprendizagem como resultado dos processos de raciocínio, também podemos pensar na inclusão de capacidades de aprendizagem com vista a tornar as componentes do sistema mais eficientes. Por exemplo, podem-se considerar mecanismos de determinação de que casos devem ser memorizados de modo a aumentar a eficiência do processo de recolha de casos. Assim, o sistema poderá incluir mecanismos de aprendizagem de quais as soluções que envolvem processos computacionais complexos. Por outro lado, casos há que ao serem acrescentados à biblioteca não aumentam as capacidades de resolução de problemas do sistema. Estes casos afectam a eficiência dos processos de recolha e requerem espaço adicional em memória. Assim, um caso só deve ser memorizado se permite ao sistema realizar alguma tarefa

que antes não era possível ou se torna o sistema mais eficiente relativamente a tarefas que já conseguia desempenhar anteriormente. O sistema de raciocínio pode ainda procurar ser antecipativo sobre a memorização de casos. Se o sistema sabe que há uma tarefa importante e que não tem o conhecimento necessário para a realizar ou que a executa de forma ineficiente, pode então colocar-se em situação de adquirir antecipadamente os casos de que necessita para melhorar o seu esempenho.

O IVY é um sistema de diagnóstico de doenças hepáticas com base em exames de raios X. Este sistema inclui na base de casos episódios sobre tarefas em relação às quais sabe ter tido dificuldades no passado. Por exemplo, o sistema por vezes é incapaz de estabelecer a diferença entre vários diagnósticos. Quando isto acontece, o IVY cria para si um objectivo que é aprender essas diferenças logo que lhe surja uma oportunidade. A partir daí, quando encontra um caso que o pode ajudar no cumprimento desse objectivo guarda-o em memória.

O AQUA, um sistema que interpreta histórias de ficção científica, gera perguntas durante o processo de leitura sobre as motivações dos protagonistas da história. Se não consegue encontrar respostas para essas questões durante a leitura do texto então o objectivo de compreensão da história mantém-se activo enquanto lê novas histórias. Logo que reconhece nessas novas histórias elementos úteis à compreensão da história anterior, o AQUA guarda esses elementos.

De salientar que o processo de aprendizagem nestes dois sistemas é antecipativo e controlado pelas necessidades do sistema de raciocínio. O IVY e o AQUA geram objectivos e guardam casos que os possam ajudar a atingir esses objectivos.

Os casos em memória podem também ser aplicados na aquisição de generalizações. Por exemplo, programas que recorrem a árvores de discriminação com redundância e a árvores de descritores partilhados (por exemplo, CYRUS, MEDIATOR, PERSUADER e JULIA) adquirem generalizações à medida que acrescentam casos à memória. Esse conhecimento generalizado é útil no processo e indexação. Sistemas como IVY e CELIA acumulam não apenas casos, mas também conhecimento geral de que necessitam e que está incluído nesses casos. Quando percepcionam que precisam de determinado conhecimento para melhorar as suas estruturas de controlo ou outros meios de raciocínio, procuram derivar esse conhecimento a partir dos casos que têm em memória.

Daquilo que foi dito decorre que num sistema de raciocínio baseado em casos aprendizagem e resolução de problemas aparecem fortemente interligados, sendo a aprendizagem muitas vezes um efeito lateral dos processos de raciocínio envolvido.

4 Aplicações do Raciocínio Baseado em Casos

Existem dois estilos principais de CBR: CBR para resolução de problemas e CBR interpretativo. No caso da resolução de problemas as soluções para o novo problema são determinadas usando as soluções de casos anteriores como directrizes. Este tipo de CBR é caracterizado por uma forte utilização de processos de adaptação para gerar as soluções e processos interpretativos para avaliar as soluções produzidas.

No CBR interpretativo o objectivo é avaliar novas situações no contexto de situações anteriores. Um advogado, por exemplo, usa CBR interpretativo quando utiliza um conjunto de casos anteriores para justificar um argumento num novo caso. O CBR interpretativo usa casos para fornecer justificações para as soluções, permitindo avaliar as soluções quando não existem métodos bem definidos e interpretar/explicar das situações quando as suas próprias definições são difusas.

De seguida apresentam-se uma listagem de várias tarefas a que já foram aplicados sistemas CBR, classificados pelo tipo de CBR efectuado por cada sistema. Na tabela a seguir contém um resumo dos sistemas referidos ao longo do texto.

Programa	**Domínio**	**Tarefa**
ABE	Eventos anómalos	Adaptação de explicações
ANON	provérbios ou sentenças	Indexação de casos prototípicos
CASEY	Problemas do coração	Explicação de anomalias
CELIA	Avarias em automóveis	Diagnóstico
CHEF	Receitas	Reparação de planos
CLAVIER	Disposição de peças numa autoclave	Layout design
COACH	Estratégia (Futebol)	Reparação de planos, contra-planeamento
CSI BATTLE PLANNER	Militar	Critica e reparação de planos
CYCLOPS	Paisagens	Projecto
CYRUS	Eventos Políticos	Organização de memória
DMAP	Análise de linguagem natural	Classificação, reconhecimento
EXPEDITOR	Missões e viagens diárias	Planeamento com objectivos múltiplos
HYPO	Direito de patentes	Avaliação por comparação
JUDGE	Sentenças criminais	Avaliação por comparação
JULIA	Catering	Projecto guiado por objectivos
KRITIK	Peças mecânicas	Projecto
MEDIATOR	Disputas de senso comum	Projecto guiado por objectivos
MEDIC	Medicina	Diagnóstico com objectivos múltiplos
PARADYME	Cozinha	Recolha paralela
PERSUADER	Contratos de trabalho	Projecto guiado por objectivos
PLEXUS	Viagens de metropolitano	Reparação de planos temporais
PROTOS	Doenças auditivas	Diagnóstico, aprendizagem
SWALE	Morte e destruição	Explicação de anomalias
TRUCKER	Escalonamento	Planeamento Oportuno

Tabela 1 – Resumo dos Sistemas CBR

4.1 O CBR na Resolução de Problemas

O Raciocínio Baseado em Casos é de extrema utilidade numa grande variedade de aplicações da resolução de problemas, incluindo *Planeamento*, *Diagnóstico* e *Projecto*. Em cada um destes domínios, os casos são úteis tanto sugerindo soluções como avisando de possíveis problemas que podem ocorrer. Por exemplo, os sistemas de projecto baseado em casos, de planeamento baseado em casos e de justificação todos recuperam e adaptam soluções de problemas anteriores semelhantes. O CBR para resolução de problemas envolve a avaliação da situação, a recolha de casos e a avaliação das semelhanças. Adicionalmente, as semelhanças e diferenças entre o caso actual e os casos anteriores são usadas para determinar como as soluções dos casos anteriores podem ser adaptadas para se ajustarem à nova situação. Por exemplo, um sistema básico de planeamento baseado em casos gera um novo plano recuperando um caso anterior com um objectivo semelhante, determinando as diferenças entre o anterior caso e o actual e adaptando o plano de modo a ter em conta os novos objectivos. A resolução de problemas baseada em casos pode ser vista como explorando as afinidades entre dois tipos de semelhanças. Estes tipos de semelhanças aplicam-se a dois espaços diferentes: o espaço das descrições dos problemas e o espaço das soluções. Esta interpretação está explícita no processo de construção de uma solução da figura seguinte:

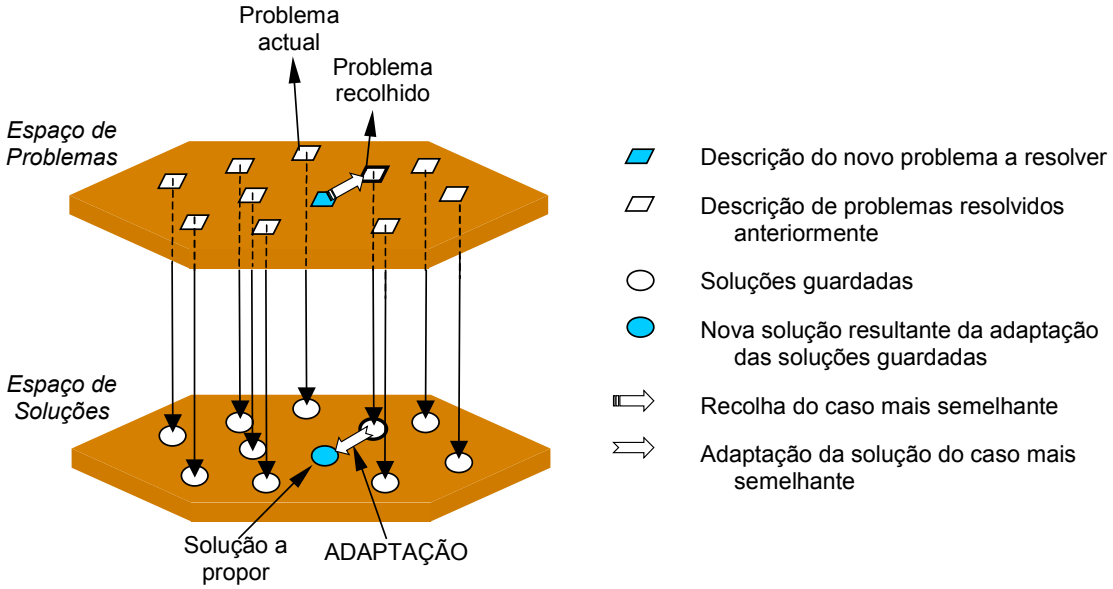

Figura 22 – Construção da solução para um novo problema.

Quando se depara com um novo problema, o sistema CBR avalia a situação para gerar uma descrição do problema e depois procura um problema anterior com uma descrição relevante (semelhante). A solução do problema mais relevante é usada como ponto de partida para gerar a solução do novo problema. Dispondo de formas correctas de descrever um problema, os problemas semelhantes têm soluções que são proveitosamente semelhantes, i. e., fáceis de adaptar à nova situação.

A figura sugere ainda um outro benefício do CBR: que múltiplos tipos de conhecimento podem ser usados para codificar informação equivalente. O conhecimento está contido não apenas na representação dos casos, método de indexação e base de caso, mas também na métrica de semelhança e nos métodos de adaptação.

4.1.1 CBR para Projecto

Nos projectos os problemas são definidos como um conjunto de restrições e o projectista deve determinar um produto que resolva o problema. Normalmente as restrições não especificam completamente o problema (i. e., existem várias soluções). No entanto, por vezes, as restrições sobredimensionam o problema (i. e., não existe nenhuma solução que satisfaça simultaneamente

todas as restrições). Além disso, no domínio dos projectos, a solução de uma parte de um problema de projecto está fortemente correlacionada com a solução das restantes partes. O CBR considera todas estas questões:

- Casos sugerem soluções para problemas subdimensionados. As soluções podem não ser completamente correctas, mas, dado que diversas soluções diferentes podem ser apropriadas, as heurísticas de adaptação podem (na maioria das situações) gerar facilmente uma solução satisfatória.
- Quando os problemas são sobredimensionados, os casos sugerem um conjunto alternativo de restrições que já foi considerado no passado. Embora possa ainda ser necessário algumas adaptações, o CBR permite evitar a aplicação de uma relaxação das restrições.
- Quando as subpartes do problema estão fortemente relacionadas, os casos podem fornecer o elo de ligação que permite uma solução integrada. Em vez de resolver as subpartes por decomposição, recomposição e resolver as divergências (de forma análoga à resolução de problemas decomponíveis), os casos sugerem uma solução completa e as partes que não se ajustem à solução proposta são adaptadas no momento.

Diversos sistemas CBR foram implementados para executar projectos baseados em casos. O JULIA idealiza refeições, o CYCLOPS usa Raciocínio Baseado em Casos para projecto de paisagens e o KRITIK combina raciocínio baseado em casos com raciocínio baseado em modelos para projectar pequenas peças mecânicas. O MEDIATOR, o primeiro sistema de resolução de problemas baseado em casos, resolve problemas de disputas simples, por exemplo, duas crianças que pretendem o mesmo rebuçado ou dois alunos que pretender usar a impressora ao mesmo tempo. O PERSUADER resolve disputas laborais.

Pelo menos um sistema para resolução de problemas de projecto foi colocado em funcionamento no mundo real. O CLAVIER tem sido usado pela *Lockheed* para colocar peças de materiais compósitos numa autoclave para serem cozidos. A tarefa é aparentemente uma arte, i. e., não é conhecido nenhum modelo de como o fazer e porquê. Peças de diferentes tamanhos necessitam de estar em diferentes partes do forno, mas o tamanho e densidade de algumas das peças podem impedir que outras sejam aquecidas correctamente. A pessoa encarregue de carregar o forno mantém uma recordação das suas experiências, tanto das que correram bem como das que não. Com base nestas experiências o CLAVIER consegue colocar as peças nas posições apropriadas do forno e evita colocar as peças em locais errados. O CLAVIER funciona tão bem quanto o perito em cujas experiências se baseia e pode, por isso, ser útil para a *Lockheed* quando o perito não está disponível. O CLAVIER usa quase sempre diversos casos para efectuar o seu projecto. Um fornece um posicionamento global que é depois adaptado convenientemente. Os outros são usados para preencher os buracos na planta que as regras de adaptação não conseguiram cobrir.

Na maioria dos problemas de projecto é necessário mais do que um caso para resolver o problema. Os problemas de projecto tendem a ser grandes e, enquanto um caso pode ser usado para resolver parte dele, não é em geral suficiente para o resolver na totalidade. Em geral um caso fornece a estrutura base para a solução e outros casos são usados para preencher os detalhes esquecidos. Deste modo é evitada a decomposição e posterior recomposição, assim como a satisfação e relaxação de grandes grupos de restrições.

4.1.2 CBR para Planeamento

O planeamento envolve um conjunto de complexidades. Um bom plano deve ter uma sequência conveniente de modo que os últimos passos do plano não desfaçam os resultados dos passos iniciais, os pré-requisitos dos últimos passos não sejam violados pelos resultados dos passos iniciais e que os pré-requisitos dos últimos passos são satisfeitos antes do passo ser iniciado. À medida que o número de passos do plano aumenta, a complexidade computacional

dos efeitos do projecto e de comparar os pré-requisitos aumenta exponencialmente. Além disso, um plano deve interagir com a complexidade do mundo real, incluindo o facto dele ser de muitas formas imprevisível, e que o tempo não é ilimitado. As sucessões de metas ou objectivos a atingir podem ter de ser alcançados quase simultaneamente. O tempo de planeamento pode ser retirado do tempo disponível para a execução do projecto. Dado que as condições se podem alterar entre o planeamento e a sua execução, o plano pode falhar na altura da execução, requerendo replanificação, recuperação ou reparação. Um planeamento com pouco tempo pode não contemplar certas oportunidades que serão melhor notados durante a fase de execução e tirar partido disso.

O CBR pode contemplar muitas destas questões:

- Os casos fornecem planos já trabalhados, em que a sequência e escalonamento dos pré-requisitos foram já organizados. Em vez de raciocinar desde o início, o sistema necessita apenas de fazer modificações nos planos anteriores.
- Se os casos forem indexados pela conjunção dos objectivos alcançados, podem ser usados para sugerir formas de alcançar determinados objectivos simultaneamente.
- Os avisos fornecidos pelos casos pode ajudar um projectista a antecipar-se e evitar problemas, diminuindo a probabilidade de falha na fase de execução.
- As estratégias de adaptação usadas para adaptar anteriores planos às novas situações podem ser usadas para recuperação e reparação na fase de execução.
- As sugestões feitas pelos casos permitem atalhar o processo de planeamento, disponibilizando mais tempo para a fase de execução.
- As sugestões feitas pelos casos permite ao sistema prestar atenção a algumas oportunidades mais facilmente durante a fase de planeamento.
- O sistema pode reparar em determinadas oportunidades durante a fase de execução e usar as suas estratégias de adaptação para actualizar o plano convenientemente.

O PLEXUS, um programa que sabe como "andar de metropolitano", é capaz de fazer reparações durante a fase de execução, adaptando e substituindo por passos semanticamente semelhante os passos que falharam.

O CHEF, um dos primeiros sistemas para planeamento, tem como objectivo antecipar problemas antes da fase de execução aprendendo com a partir de experiências anteriores problemáticas. Quando ocorrem problemas durante a fase de execução, o CHEF tenta explicá-los e depois determinar como resolvê-los. A reparação hipotética é guardada em memória e o caso é indexado por características que, com maior probabilidade, predizem que o problema vai reocorrer.

O TRUCKER é um programa de recados que guarda o trajecto dos seus objectivos pendentes e está preparado para tirar proveito das oportunidades que surgem, permitindo-lhe atingir os objectivos mais cedo que o esperado. O MEDIC é um programa de diagnóstico que é capaz de reusar anteriores planos para fazer diagnósticos, mas é suficientemente flexível para seguir em frente perante eventos inesperados. O EXPEDITOR planeia os eventos na vida de um pai solteiro que tem de lidar tanto com os filhos como com o trabalho. Este sistema esconde a suas experiências em atingir múltiplos objectivos intercalando-os. Embora seja lento nas suas planificações iniciais, ganha competência ao longo do tempo à medida que pode reutilizar o seus planos anteriores. O CSI BATTLE PLANNER mostra como os casos podem ser usados para criticar e reparar planos antes destes serem executados.

4.1.3 CBR para Diagnóstico

No diagnóstico, um sistema de resolução de problemas recebe um conjunto de sintomas sendo-lhe pedido que os explique ou esclareça. O sistema de diagnóstico pode usar casos para

sugerir explicações para os sintomas e para alertar para explicações que se verificou serem inapropriadas no passado.

Claro que não se pode esperar que um diagnóstico anterior se aplique intacto ao novo caso. Tal como no caso do planeamento e no projecto, é frequentemente necessário adaptar um diagnóstico anterior para se ajustar à nova situação. O CASEY é capaz de diagnosticar problemas de coração através da adaptação dos diagnósticos de pacientes anteriores aos novos pacientes. O CASEY é um programa relativamente simples implementado sobre um programa de diagnóstico baseado em modelos já existente. Quando um novo caso é semelhante a um já visto anteriormente o CASEY é várias ordens de grandeza mais eficiente a gerar o diagnóstico que o programa baseado em modelos. As adaptações do CASEY são baseadas num modelo causal, pelo que os seus diagnósticos são tão precisos quanto os feitos a partir do início com base no mesmo modelo causal.

Os casos são também úteis para mostrar o caminho de saída de anteriores dificuldades. O PROTOS está implementado para assegurar que isto é feito de modo eficiente. O PROTOS diagnostica doenças no ouvido. Neste domínio, muitos dos diagnósticos manifestam-se de modos muito semelhantes e são difíceis de diferenciar. Enquanto um aprendiz não têm conhecimento destas diferenças subtis, os peritos sim. O PROTOS começa como um aprendiz e, quando comete erros, um "mestre" explica-lhe os seus erros. Como resultado, o PROTOS aprende estas diferenças subtis, permitido-lhe no futuro mudar do diagnóstico mais óbvio para o diagnóstico correcto.

Geralmente, um diagnóstico desde o início é uma tarefa bastante demorada. Na maioria dos domínios do diagnóstico existe, no entanto, uma regularidade suficiente para que uma abordagem baseada em casos seja eficiente. É claro que o programa não pode assumir que a sugestão baseada em casos está correcta. Esta sugestão baseada em casos tem de ser validada. Frequentemente, no entanto, a validação é bastante mais fácil que a geração do diagnóstico.

4.2 CBR Interpretativo

No CBR interpretativo o objectivo é formar uma opinião sobre a classificação de uma nova situação, por comparação com casos que foram já classificados. O CBR interpretativo é um processo de avaliar situações ou soluções no contexto da experiência anterior. É tomada como entrada uma situação ou solução e a saída é a classificação da situação, um argumento que suporte a classificação ou solução e/ou justificações que suportem o argumento ou solução. É útil para classificação de situações, avaliação de uma solução, argumentação, justificação de uma solução (ou plano) e a projecção dos possíveis efeitos de uma decisão ou plano.

Como exemplo podemos referir que o CBR interpretativo tem uma importância fundamental na interpretação de conceitos legais e na aplicação de leis no sistema legal Norte Americano. Um advogado argumentando que o seu cliente deve ter direito a uma dedução fá-lo usando precedentes, i. e., mostrando que a dedução foi concedida em casos anteriores semelhantes e mostrando que esses casos são mais importantes do que aqueles em que a dedução foi recusada. O CBR interpretativo é também útil em tarefas como o diagnóstico em que um problema pode ser diagnosticado por comparação e contraste dos sintomas do caso actual com os dos casos anteriores de modo a determinar o melhor diagnóstico.

Na sua forma mais simples, o CBR interpretativo envolve quatro tarefas. Primeiro o sistema deve fazer uma avaliação da situação de modo a determinar quais as características que são na realidade relevantes. Segundo, com base nos resultados da avaliação da situação, o sistema tenta encontrar um ou mais casos anteriores relevantes. Terceiro, o sistema compara os casos relevantes recolhidos com a nova situação para determinar que interpretação se pode aplicar. Finalmente, a situação actual e a respectiva interpretação são guardadas como um novo caso.

O CBR interpretativo é mais útil na avaliação quando não existem métodos computacionais para avaliar uma solução. Normalmente nestas situações existem tantas indefinições que, mesmo que existissem métodos computacionais disponíveis, o conhecimento necessário para os aplicar

não estaria presente. Um especialista que usa casos para o ajudar a avaliar e justificar decisões ou interpretações está a fazer uso da sua falta de conhecimento assumindo que o ambiente é consistente.

4.2.1 Raciocínio Justificativo e Adverso (Concorrente)

Raciocínio adverso ou concorrente significa fazer argumentações persuasivas de modo a convencer os outros de que nós ou as nossas posições estão correctas. Um argumento persuasivo estabelece uma posição e justifica-a, seja através de factos seja através de inferências. Mas, na maior parte das vezes, a única forma de justificar uma posição é citando casos ou experiências anteriores relevantes. A área do direito disponibiliza, por isso, um bom domínio para o estudo do raciocínio adverso e da justificação baseada em casos e muita da investigação nesta área usa o direito.

O HYPO é o primeiro e mais sofisticado sistema de raciocínio legal baseado em casos. O método seguido pelo HYPO para gerar um argumento e justificar uma solução ou posição tem vários passos. Primeiro é analisada a nova situação em busca de factores relevantes. Com base nestes factores são recolhidos casos semelhantes. Estes casos são "posicionados" relativamente à nova situação. Alguns serão a favor dela e outros contra ela. O melhor dos casos de cada um destes conjuntos é seleccionado. O caso que melhor suporta a situação é usado para criar um argumento para a solução proposta. O caso que melhor contraria a situação é usado para propor contra exemplos. Os casos do conjunto que suporta a situação são então usados para contrariar os contra exemplos. O resultados desta sequência de operações é um conjunto de argumentos "em três níveis" que suportam a solução, cada um dos quais é justificado com casos. Um efeito marginal importante de gerar estes argumentos é que os potenciais problemas são revelados.

Em geral os casos são úteis para construir argumentos e justificar posições quando não os princípios concretos não existem ou são em número reduzido, se os princípios são inconsistentes ou se os seus significados não estão bem especificados.

4.2.2 Classificação e Interpretação

Interpretação no contexto do CBR significa decidir quando um conceito se ajusta a alguma classificação com limites abertos ou difusos. A classificação pode ser feita à mão no momento, com base na tarefa ou pode ser bem conhecida, mas não estar convenientemente definida em termos de condições necessárias e suficientes. Muitas das classificações assumidas a seguir são do tipo aberto. Por exemplo, assumimos que um veículo se refere a algo com rodas usado para transporte, mas se um sinal disser – *Proibida a entrada de veículos no parque* – provavelmente não se está a referir a uma cadeira de rodas ou a um carrinho de bebé, embora ambos se ajustem a esta definição simples.

Uma das formas de funcionamento da classificação CBR é perguntar se o novo conceito é suficientemente parecido com outro conhecido para ter a mesma classificação. O PROTOS, um sistema que permite diagnosticar doenças auditivas, funciona desta forma. Em vez de fazer a classificação de novos casos usando condições necessárias e suficientes, o PROTOS faz a classificação tentando encontrar o caso da base de casos mais semelhante ao novo caso. O novo caso é então classificado de acordo com a classificação desse caso. Para o fazer, o PROTOS mantém a informação de quão prototípico é cada um dos seus casos e o que diferencia os diversos casos dentro de uma mesma classificação. Primeiro é escolhida a classificação mais plausível e depois o caso com maior semelhança nessa classe.

Quando não existe nenhum caso suficientemente semelhante, é por vezes necessário considerar situações hipotéticas. Muito do trabalho neste tipo de interpretação vem também da área do direito. O HYPO usa hipotéticos para uma diversidade de tarefas necessárias para uma boa interpretação: para redefinir situações anteriores em termos de novas situações, para criar novos casos standard quando não existe o necessário, para explorar e testar os limites da razoabilidade de um conceito, para reconsiderar um caso excluindo alguns aspectos, para procurar suposições ocultas e para organizar ou agrupar os casos. O HYPO cria hipotéticos

fazendo cópias da situação actual que são mais fortes ou mais fracas que a situação real para um lado ou para outro. Esta tarefa é orientada por um conjunto de heurísticas que propõem como devem ser criados os hipotéticos com base nas necessidades de raciocínio.

4.2.3 Projecção de Resultados

A projecção, o processo de predizer os resultados de uma decisão ou plano, é uma parte importante da avaliação de qualquer esquema de planeamento ou tomada de decisão. Quando se conhece tudo acerca de uma situação, a projecção é simplesmente o processo de correr inferências conhecidas a partir da solução para verificar onde conduzem. Mais frequentemente, no mundo real nada é conhecido na sua totalidade e os seus efeitos ou resultados não podem ser previstos com exactidão a partir de um simples conjunto de inferências.

Os casos fornecem uma forma de prever os resultados com base naquilo que se verificou no passado. Casos com planos semelhantes que tiveram falhas podem indicar problemas potenciais no plano. Casos com planos semelhantes que tiveram sucesso dão credibilidade ao plano actual.

A projecção automática baseada em casos não tem sido alvo de investigação, mas o apoio a pessoas que fazem projecções tem merecido atenção. O CSI's BATTLE PLANNER é um sistema de recolha de casos cuja interface foi feita para permitir a uma pessoa usar casos para prever resultados. Um pretendente a comandante pode propor um plano como solução ao sistema. O BATTLE PLANNER recolhe o caso mais parecido que use um plano semelhante e divide-os em situações de sucesso e de fracasso. A pessoa pode então examinar os casos e usá-los para corrigir o seu próprio plano, tentando depois uma avaliação semelhante de plano corrigido. Uma forma de utilização alternativa consiste na pessoa usar o sistema para fazer uma análise de sensibilidade. Manipulando os detalhes da situação e verificando a alteração do número de vitórias e derrotas, a pessoa pode determinar quais os factores da situação que são cruciais, devendo ser alterados, e quais os que devem permanecer inalterados.

4.2.4 O CBR Interpretativo e a Resolução de Problemas.

Muito do trabalho na área da interpretação tem estado centrada no domínio do direito e tendo em vista justificar um argumento a favor ou contra uma determinada interpretação da lei. No entanto, o CBR interpretativo não se destina exclusivamente a problemas de interpretação. É também bastante útil nas fases de avaliação e crítica da resolução de problemas sempre que não existe um bom modelo causal do domínio de aplicação. Apesar de ainda existir pouco trabalho desenvolvido nesta área, as tarefas envolvidas no CBR interpretativo têm potencial para desempenhar papéis importantes na resolução de problemas. Primeiro, se a estrutura da solução for conhecida, podem ser usado CBR interpretativo para escolher os casos que podem fornecer essa solução. Segundo, a elaboração de argumentos e justificações resultam na percepção de quais as características em que nos devemos centrar. Além disso, os métodos interpretativos podem ser usados para prever a utilidade, qualidade ou resultados de uma solução.

5 Bibliografia

[Watson 1997] Ian Watson, "Applying Case-Based Reasoning: techniques for Enterprise Systems," San Francisco, California, Morgan Kaufmann Publishers, Inc., 1997.

[Kolodner 1993] Janet Kolodner, "Case-Based Reasoning," San Francisco, California, Morgan Kaufmann Publishers, Inc., 1993.

[Leake 1996] David B. Leake, "Case-Based Reasoning: Experiences, Lessons & Future Directions," Menlo Park, California, AAAI Press/The MIT Press, 1996.

[Aamodt 1994] A. Aamodt, E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," AI Communications, IOS Press, Vol. 7, 1, 1994, pp. 39-59.

[Kolodner 1991] Janet L. Kolodner and Menachem Y. Jona, "Case-Based Reasoning: An Overview," Technical Report #15. The Institute for the Learning Sciences. NorthWestern University. June, 1991.

[Mántaras] Ramon López de Mántaras and Enric Plaza, "Case-Based Reasoning: An Overview,".

[Watson 1994] Ian Watson and Farhi Marir (1994), "Case-Based Reasoning: A Review," The Knowledge Engineering Review, Vol. 9, No. 4, 1994, pp. 327-354.

[Riesbeck 1989] C.K. Riesbeck and R.C. Schank, "Inside Case-Based Reasoning," Hillsdale, Lawrence Erlbaum Associates, 1989.