Tecnologia para Sistemas Inteligentes
Apontamentos para as aulas sobre

Representação e Raciocínio com Enquadramentos

Luís Miguel Botelho

Departamento de Ciências e Tecnologias da Informação
Instituto Superior de Ciências do Trabalho e da Empresa

Abril de 2012

Tecnologias para Sistemas Inteligentes

Apontamentos para as aulas

Índice

Representação e Raciocínio com Enquadramentos

O raciocínio baseado na lógica de primeira ordem (FOP, "*First Order Logic*" ou FOPC, "*First Order Predicate Calculus*") é um raciocínio monótono, no sentido em que o acréscimo de novos conhecimentos à base de conhecimentos não resulta na invalidação de conclusões previamente geradas. Isto significa também que a teoria que se deriva de uma base de conhecimentos mantém-se constante ou aumenta com a introdução de novos conhecimentos.

O raciocínio não monótono não tem estas propriedades. A introdução de novos conhecimentos pode invalidar conclusões previamente geradas (i.e., pode originar um decréscimo da teoria). Pode saber-se se um determinado tipo de raciocínio é monótono pelas regras de inferência desse raciocínio. Se a conclusão de uma regra de inferência depender de todo o conteúdo da base de conhecimentos, temos um raciocínio não monótono. Se as conclusões das regras de inferência dependerem apenas de um número reduzido de premissas (mas não de todas), o raciocínio é monótono.

O raciocínio com base no factor de confiança é um raciocínio não monótono porque o factor de confiança global de uma conclusão depende de toda a base de conhecimentos (é o maior dos factores de confiança locais dessa conclusão). Portanto se a conclusão A com factor de confiança global CFA for produzida a partir de uma dada base de conhecimentos, a introdução de novos conhecimentos poderá fazer aumentar a confiança de A (invalidando a confiança global anteriormente determinada).

Existem diversos outros tipos de raciocínio não monótono, entre os quais, os mais frequentemente referidos e usados são a chamada hipótese do mundo fechado (CWA, "Closed World Assumption") e o raciocínio por omissão ("default reasoning").

A hipótese do mundo fechado consiste em dizer que tudo quanto não se diz explicitamente é falso. Isto é, aquilo que não se puder derivar de uma base de conhecimentos não é verdadeiro. Isto significa dizer que a parte relevante do mundo (de um ponto de vista específico) está totalmente representada na teoria que se deriva da base de conhecimentos (o que não estiver, não é verdade). O Prolog usa esta hipótese.

O raciocínio por omissão é uma forma de raciocínio em que as conclusões podem ser produzidas por omissão de informação em contrário. Neste tipo de raciocínio, as regras têm enunciados do tipo "*Geralmente (não havendo nada em contrário), se P(x), então Q(x)*". Por exemplo, a regra "*todos os pássaros voam*" passa a dizer-se "*geralmente os pássaros voam*".

1 Representação baseada em enquadramentos

Na área da Inteligência Artificial, usam-se enquadramentos como método para representar conhecimento sobre conceitos. A Figura 1 ilustra uma hierarquia de enquadramentos que representa diversas classes (animais, mamíferos, aves, morcegos, vacas e pinguins) e indivíduos (Flor e Sienna). Uma das principais características dos enquadramentos é a possibilidade de aglomerar em cada unidade de representação, diversas informações, regras e procedimentos fortemente relacionados entre si. A outra vantagem (menos importante, do meu ponto de vista) é a utilização implícita de um tipo particular de raciocínio por não monótono: a herança (múltipla ou simples) com excepções. Finalmente, o método de representação por enquadramentos presta-se à utilização de uma linguagem gráfica, o que tem as vantagens e desvantagens geralmente associadas a representações gráficas.

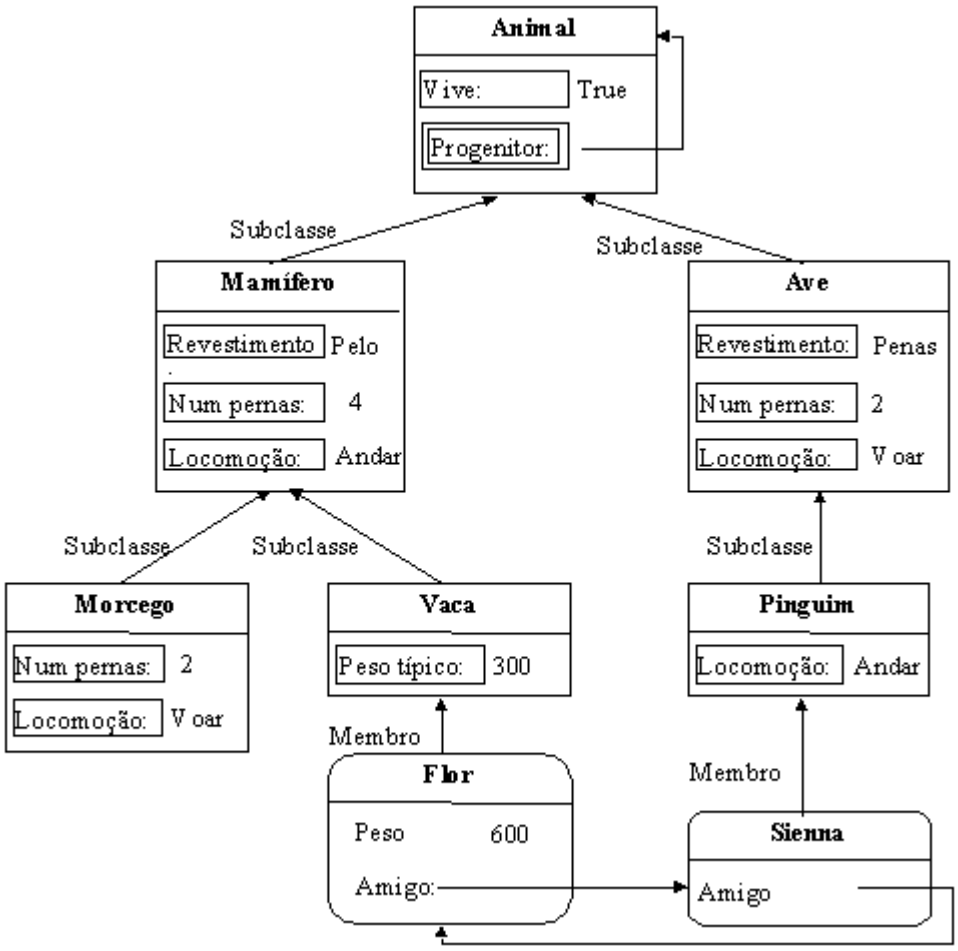

Figura 1 - Herança com excepções

A Figura 1 representa uma hierarquia de enquadramentos. Apesar de recorrer a uma linguagem gráfica, tem bastante rigor. Os rectângulos são enquadramentos que representam classes. Os rectângulos de cantos arredondados são enquadramentos que representam indivíduos. Cada enquadramento é caracterizado por um conjunto de atributos. Os atributos podem ser de três tipos conforme estão indicados por um rectângulo de traço duplo, ou por um rectângulo de traço simples, ou sem rectângulo.

O atributo Revestimento da classe Mamífero (indicado com um rectângulo) tem o valor Pelo. Isto significa que (em geral) todos os mamíferos têm o corpo coberto de pelo.

O atributo Progenitor da classe Animal (indicado com um rectângulo de traço duplo) tem o valor Animal (uma classe). Isto significa que (em geral) para cada indivíduo da classe Animal existe um indivíduo da classe Animal (coincidência) que é seu progenitor.

O atributo Amigo do enquadramento Flor (indicado sem rectângulo) tem o valor Sienna. Isto significa que o amigo do indivíduo Flor é o indivíduo Sienna. O atributo Peso do enquadramento Flor tem valor 600. Isto significa que o peso do indivíduo Flor é 600. Os atributos sem rectângulo são privados do enquadramento a que pertençam. No caso de o enquadramento representar uma classe, os indivíduos da classe não herdam os atributos privados da classe.

Além de representar atributos, os sistemas de enquadramentos representam também relações hierárquicas entre enquadramentos através dos arcos Membro e Subclasse. O arco membro (de um indivíduo para uma classe) significa que um dado enquadramento representa um indivíduo que é membro da classe representada por outro enquadramento. O arco Subclasse significa que um dado enquadramento é uma subclasse de uma classe representada por outro enquadramento.

Os arcos Membro e Subclasse permitem a utilização de um mecanismo de herança com excepções. Os morcegos têm duas pernas, o que é uma excepção em relação ao número de pernas dos mamíferos. O modo de locomoção dos pinguins (andar) também é uma excepção à regra geral das aves, as quais voam.

Além da designação, dos atributos e dos arcos entre enquadramentos, também se podem representar métodos no interior dos enquadramentos.

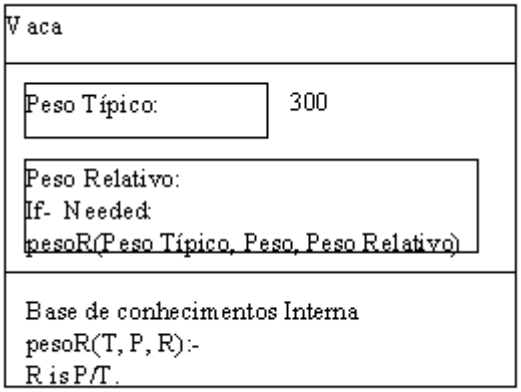

Figura 2 – Enquadramento com um método definido na sua BC interna

A Figura 2 representa uma nova versão do enquadramento Vaca em que o predicado *peso_relativo/3* é definido internamente ao enquadramento. Este predicado é usado quando se pretende determinar o valor do peso relativo de um indivíduo da classe das vacas.

2 Herança múltipla

Nesta secção mostramos que a herança é um tipo de raciocínio não monótono. A Figura 3 mostra um sistema de enquadramentos em que a introdução de nova informação (consistente com o conhecimento existente) invalida conclusões produzidas antes da sua introdução. Numa primeira situação, a classe dos homens não tem qualquer caracterização particular, toda a informação é herdada da classe dos mamíferos. Na situação seguinte, é introduzida nova informação na base de conhecimentos, dizendo que os elementos da classe dos homens têm duas pernas. Esta nova informação (totalmente consistente com a informação prévia) implica que conclusões já produzidas tenham que ser invalidadas.

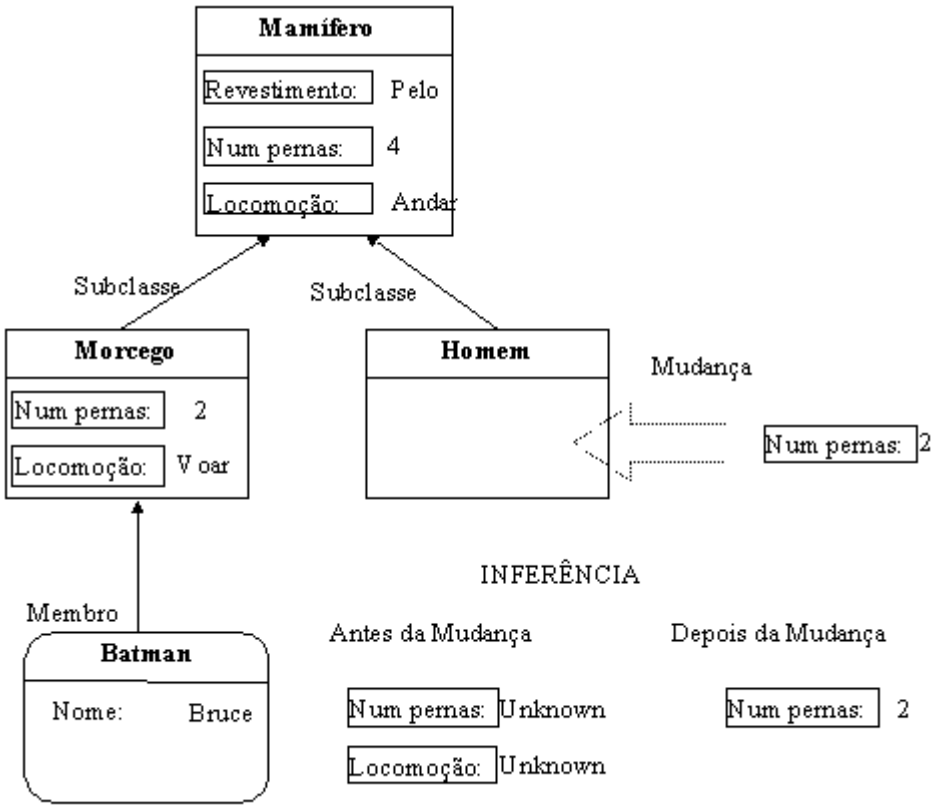

Figura 3 - Herança múltipla

Na primeira situação (classe Homens sem especificação particular), podemos concluir através de herança que o número de pernas e o modo normal de locomoção do Batman não são conhecidos. Por ser membro da classe dos morcegos, o Batman deveria ter duas pernas e locomover-se andando. Por outro lado, por ser membro da classe dos homens e consequentemente da classe dos mamíferos, o Batman deveria ter 4 pernas e deveria voar.

Na situação seguinte, a introdução da nova proposição "*(Em geral) todos os homens têm duas pernas*", implica a geração da conclusão "*o Batman tem 2 pernas*", inviabilizando a conclusão prévia "*o Batman tem um número indeterminado de pernas*".

Em geral, o acréscimo de nova informação a um sistema de enquadramentos pode implicar a remoção (inviabilização) de conclusões anteriores. Isto significa que a herança é um raciocínio não monótono.

Nota: não gosto deste exemplo porque uma conclusão "unknown" não é propriamente uma conclusão.

3 Exemplo: representação e raciocínio com enquadramentos

Nesta secção, analisa-se o raciocínio efectuado em sistemas de representação por enquadramentos. Na primeira sub-secção apresenta-se um exemplo de representação de conhecimento através de enquadramentos. Na segunda secção, descreve-se um formalismo para representação de enquadramentos em texto. As duas secções seguintes descrevem o raciocínio baseado em enquadramentos, encadeado para trás e encadeado para a frente.

3.1 Exemplo

O exemplo que se apresenta na Figura 3 permite ilustrar a utilização da representação baseada em enquadramentos ("frames") com um raciocínio guiado pelos dados (encadeado para a frente), e com um raciocínio guiado pelos objectivos (encadeado para trás).

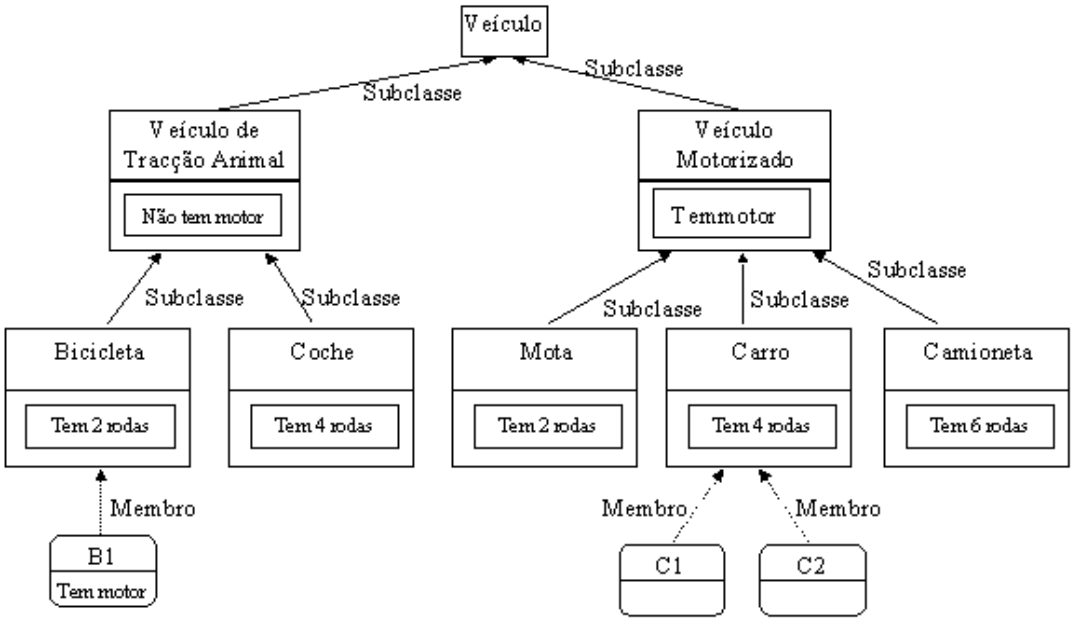

Figura 4 - Hierarquia de enquadramentos

Na Figura 4 representa-se conhecimento sobre a classe dos veículos, os quais se podem dividir em veículos de tracção animal e veículos motorizados. Por sua vez, os veículos de tracção animal podem ser bicicletas ou coches; e os veículos motorizados podem ser motas, carros ou camionetas.

Além desta descrição hierárquica, os enquadramentos apresentados na Figura 4 representam ainda as seguintes proposições:

Geralmente (i.e., se não for dito nada em contrário), os veículos de tracção animal não têm motor;
Geralmente (i.e., se não for dito nada em contrário), os veículos motorizados têm motor;
Geralmente (i.e., se não for dito nada em contrário), as bicicletas têm 2 rodas;
Geralmente (i.e., se não for dito nada em contrário), os coches têm 4 rodas;
Geralmente (i.e., se não for dito nada em contrário), as motas têm 2 rodas;
Geralmente (i.e., se não for dito nada em contrário), os carros têm 4 rodas;
Geralmente (i.e., se não for dito nada em contrário), as camionetas têm 6 rodas.
B_1 é uma bicicleta;
C_1 é um carro;
C_2 é um carro;
B_1 tem motor (bicicleta específica com um motor auxiliar).

Note-se que as proposições R_i podem ser comparadas a regras (com excepções). Por exemplo, R_1 poderia ser parcialmente representada pela regra "*Se X é um veículo de tracção animal, então (em princípio) X não tem motor*".

3.2 Representação textual de enquadramentos

Nesta subsecção apresenta-se um formalismo textual para representar porções de hierarquis de enquadramentos. O formalismo descrito é totalmente introduzido neste texto e não tem qualquer outro propósito senão o de facilitar as descrições que se seguem sem recorrer a ferramentas gráficas.

As entidades com existência autónoma numa estrutura de enquadramentos são os enquadramentos (classes e objectos) e os arcos. Os atributos e propriedades dos enquadramentos serão representados apenas no contexto do seu enqudramento. A tabela da Figura 5 contém uma descrição dos símbolos usados na representação de estruturas de um sistema de representação por enquadramentos.

Símbolo	Significado	Exemplo	Descrição
=	Representação do valor de um atributo	A=V	V é o valor do atributo A
<texto>	Uma propriedade binária representa-se por texto livre	Tem pele	Propriedade da entidade representada pelo enquadramento em que aparece.
[]	Conjunto de atributos e/ou propriedades	[]	Conjunto vazio de atributos e/ou propriedades
		[…]	Conjunto com qualquer número de atributos e/ou propriedades
		[A1=V1, A2=V2, …]	Conjunto de atributos e/ou propriedades que contem pelo menos os atributos A1 e A2.
?	Qualquer coisa (enquadramento, atributo, valor, propriedade)	[A1=?, ?=V2, ?]	Conjunto com um atributo A1 com qualquer valor, com um atributo qualquer com o valor V2, e com mais um atributo ou propriedade
class	Functor para representação de um enquadramento do tipo classe	class(Bicicleta, [])	Enquadramento sem atributos que representa a classe das bicicletas
obj	Functor para representação de um enquadramento do tipo objecto individual	obj(B1, [cor=vermelho])	Enquadramento que representa o objecto individual B1, o qual é vermelho.
—<texto>→	Arco entre dois enquadramentos	obj(B1)—Membro→class(Bicicleta)	B1 é membro da classe das bicicletas. Os atributos de B1 e de Bicicleta não especificados.
…	Sub-estrutura numa hierarquia de enquadramentos.	obj(B1)—Membro→…—Subclasse→class(Bicicleta)	Caminho de B1 para Bicicleta. O primeiro arco desse caminho é um arco *Membro*; o último é um arco *Subclasse*.

Figura 5 – Símbolos usados na representação textual de enquadramentos

3.3 Raciocínio guiado por objectivos (encadeado para trás)

O conhecimento representado na Figura 4 pode ser usado com raciocínio orientado pelos objectivos (i.e., raciocínio encadeado para trás). A Figura 6 ilustra uma possível interacção com um sistema baseado em conhecimento cuja base de conhecimentos contém os enquadramentos da Figura 4. Como sempre, no raciocínio encadeado para trás, o utilizador (ou outro sistema) faz uma pergunta ao sistema. Este procura a estrutura de representação de conhecimento que produz a resposta para a pergunta feita (objectivo). Uma vez seleccionada essa estrutura, o sistema verifica se as condições para a sua utilização estão satisfeitas. Para isso, é possível que o sistema tenha que criar um ou mais novos (sub)objectivos.

		Estrutura de representação
Utilizador:	B_1 é uma bicicleta?	obj(B1)—Membro→class(Bicicleta)
Sistema:	Sim	obj(B1, [Tem motor])—Membro→class(Bicicleta, [Tem 2
Utilizador:	B_1 tem motor?	obj(B1, [tem motor, …])
Sistema:	Sim	obj(B1, [tem motor])
Utilizador:	Quantas rodas tem	obj(B1, [Tem ? rodas, …])
Sistema:	B_1 tem 2 rodas	obj(B1, [Tem motor])—Membro→class(Bicicleta, [Tem 2
Utilizador:	C1 tem motor?	obj(C1, [Tem motor])
Sistema:	Sim	obj(C1)—Membro→…—Subclasse→class(Veículo

Figura 6 Sistema de enquadramentos com raciocínio encadeado para trás

Na primeira interacção, o sistema procura uma estrutura de representação cuja utilização permite concluir "*B_1 é uma bicicleta*" (i.e., "*B_1 é membro da classe Bicicleta*"). A estrutura obj(B1, Tem motor])—Membro→class(Bicicleta, [Tem 2 rodas]) representa esse facto (ver Figura 4), pelo que a resposta é afirmativa.

Na segunda interacção, passa-se algo muito semelhante: a estrutura obj(B1, [tem motor]) representa a afirmação "*B_1 tem motor*".

Na terceira interacção, o sistema tem que usar uma estrutura de representação que permite concluir "*B_1 tem X rodas*". Todas as representações class(Bicicleta, [Tem 2 rodas]), class(Coche, [Tem 4 rodas]), class(Mota, [Tem 2 rodas]), class(Carro, [Tem 4 rodas]) e class(Camioneta, [Tem 6 rodas]) são as estruturas candidatas que poderão cocluir a respeito do número de rodas de um dado enquadramento. Para que qualquer destas estruturas possa ser usada para gerar esta conclusão, tem que se determinar se as pré condições da conclusão estão satisfeitas. Para que a primeira estrutura produza a conclusão "*B_1 tem 2 rodas*", é necessário que a proposição "*B_1 é membro da classe Bicicleta*" possa ser estabelecida. Como já vimos, a estrutura obj(B1, Tem motor])—Membro→class(Bicicleta, [Tem 2 rodas]) representa essa relação, portanto a resposta pode ser dada.

Na quarta interacção, o sistema tem que seleccionar as estruturas de representação que lhe permitam afirmar "*C_1 tem motor*". O enquadramento que representa a classe dos veículos motorizados (class(Veículo Motorizado, [Tem motor])) permite produzir essa conclusão desde que as pré-condições sejam satisfeitas, i.e., desde que *C_1* seja membro da classe *Veículo Motorizado*. Como a proposição "*C_1 é membro da classe Veículo Motorizado*" não está representada explicitamente nos enquadramentos apresentados na Figura 4, o sistema tem que verificar se existe alguma subclasse de *Veículo Motorizado* da qual C_1 seja um membro. A estrutura obj(C1)—Membro→class(Carro, [Tem 4 rodas])—Subclasse→class(Veículo Motorizado, [Tem motor]) representa essa relação.

3.4 Raciocínio guiado pelos dados (encadeado para a frente)

Quando o raciocínio é guiado pelos dados, sempre que um facto emparelha com as condições, o sistema produz as conclusões.

Usando o conhecimento representado na Figura 4, podemos imaginar a seguinte interacção (Figura 7)

Utilizador:	C_3 é membro da classe Carro!
Sistema:	C_3 tem 4 rodas C_3 é membro da classe Veículo Motorizado C_3 tem motor C_3 é membro da classe Veículo

Figura 7 - Interacção guiada pelos dados

Na interacção apresentada na Figura 7, o utilizador introduz um novo facto no sistema: "*C_3 é membro da classe Carro*", o qual se representa através da estrutura obj(C3)—Membro→class(Carro). Dado que o raciocínio do sistema é encadeado para a frente, o sistema vai verificar que estruturas de representação ficam com a condição satisfeita devido à introdução deste novo facto. A estrutura class(Carro, [Tem 4 rodas]) permite concluir que C_3 têm 4 rodas.

Dado que

obj(C3)—Membro→class(Carro) e

class(Carro, [Tem 4 rodas])—Membro→class(Veículo Motorizado, [Tem motor])

o sistema concluí obj(C3)—Membro→class(Veículo Motorizado, [Tem motor])

(este facto não aparece representado explicitamente). Usando este último facto, o sistema também concluí "*C_3 tem motor*" (obj(C3, [Tem motor])).

Finalmente, usando

obj(C3)—Membro→class(Veículo Motorizado, [Tem motor]) e

class(Veículo Motorizado, [Tem motor])—Subclass→class(Veículo)

o sistema conclui "*C_3 é membro da classe Veículo*" (obj(C3)—Membro→class(Veículo)).

4 Implementação de um sistema de enquadramentos com herança simples

Um sistema de enquadramentos é um sistema de representação que facilita a estruturação do conhecimento.

Uma das características deste método é a possibilidade de agrupar informações e conhecimentos distintos mas relacionados na mesma unidade de representação. Estas unidades de representação, as quais descrevem conceitos complexos em termos das suas propriedades mais simples, chamam-se enquadramentos ou quadros ("frames"). A característica mais distintiva dos enquadramentos é a representação de conceitos compostos por oposição à representação de conceitos atómicos da lógica e dos métodos baseados na lógica como os sistemas de regras.

A outra característica é a possibilidade de organizar o conhecimento em estruturas hierárquicas em que as propriedades dos enquadramentos de nível hierárquico superior são herdadas pelos enquadramentos de nível hierárquico inferior.

Num enquadramento é possível representar quer propriedades estáticas (e.g., atributos cujo valor não se altera) quer propriedades dinâmicas, isto é, atributos cujo valor é determinado pela execução de um procedimento ou de uma função. As propriedades dos enquadramentos chama-se ranhuras ("slots"). Os procedimentos associados às propriedades dos enquadramentos chamam-se demónios ("demons"). A secção 4.1 centra-se na implementação de um sistema de representação por enquadramentos com propriedades estáticas, e de um mecanismo de herança de propriedades. A secção 4.2 melhora a implementação da secção 4.1 pela definição de um sistema de representação de propriedades dinâmicas.

Dado que o aspecto mais importante do método de representação por enquadramentos é a possibilidade de captar naturalmente conceitos compostos, as ferramentas computacionais para a criação de sistemas com enquadramentos devem privilegiar um modo de edição de conhecimento em que a sensações de unidade de representação composta e de agregação de informações estejam sempre presentes. Esta exigência encontra a resposta mais adequada nos editores especializados criados sobre sistemas de interacção gráfica. O conhecimento editado através dos interfaces gráficos deve ser convertido pela ferramenta computacional numa representação interna adequada à implementação do raciocínio com herança de propriedades.

Nas secções 4.1 e 4.2 analisam-se métodos de representação interna dos enquadramentos e mecanismos computacionais de manipulação dessas representações. A edição e a inspecção do conhecimento através de interfaces gráficas orientadas para as noções de unidade de representação composta e estruturada e de árvore hierárquica com herança de propriedades, e a conversão entre as representações ao nível da interface e as representações internas estão fora do âmbito deste documento.

4.1 Herança num sistema de enquadramentos

Em geral, um conjunto de enquadramentos relacionados entre si através de relações hierárquicas, representa-se graficamente através de uma estrutura em árvore como a da Figura 8.

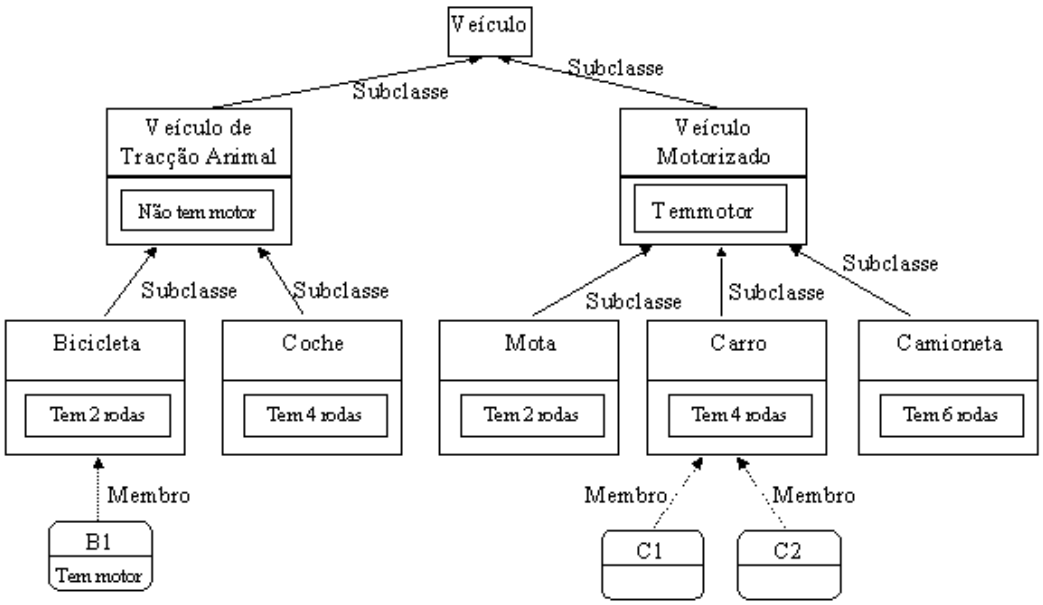

Figura 8 – Estrutura hierárquica de enquadramentos

Internamente, uma estrutura hierárquica de enquadramentos pode ser representada por exemplo através de um conjunto de factos. As relações *class/1* e *object/1* são usadas para enumerar os enquadramentos que representam classes e objectos individuais respectivamente. A relação *evalue/3* ("explicit value") é usada para representar o valor de uma propriedade de um enquadramento explicitamente representados na hierarquia (i.e., explicitamente introduzidos pelo engenheiro do conhecimento ou criados dinamicamente por um procedimento computacional). As relações *ememb/2* ("explicit member") e *esub/2* ("explicit subclass") são usadas para representar os arcos *Member* e *Subclass* explicitamente representados na hierarquia.

Além das relações explícitas é necessário definir as relações implícitas na hierarquia. Estas definições implícitas constituem o mecanismo de inferência do sistema de enquadramentos, isto é a herança de propriedades. São definidas as relações implícitas *value/3*, *memb/2* e *subclass/2*. A Figura 9 mostra a representação interna da hierarquia de enquadramentos explicitamente representada na Figura 8.

```
attribute(traccao).
attribute(nrodas).

class(veiculo).

class(vanimal).
esub(vanimal, veiculo).
evalue(vanimal, traccao, animal).

class(vmotor).
esub(vmotor, veiculo).
evalue(vmotor, traccao, motor).

class(bicicleta).
esub(bicicleta, vanimal).
evalue(bicicleta, nrodas, 2).

class(coche).
esub(coche, vanimal).
evalue(coche, nrodas, 4).

class(mota).
esub(mota, vmotor).
evalue(mota, nrodas, 2).

class(carro).
esub(carro, vmotor).
evalue(carro, nrodas, 4).

class(camioneta).
esub(camioneta, vmotor).
evalue(camioneta, nrodas, 6).

object(b1).
ememb(b1, bicicleta).
evalue(b1, traccao, animal).

object(c1).
ememb(c1, carro).
object(c2).

object(c2).
ememb(c2, carro).
```

Figura 9 – Representação interna de uma hierarquia de enquadramentos

A Figura 10 mostra a definição dos mecanismos de herança num sistema de enquadramentos. As ideias de base deste mecanismo de herança são as seguintes:

O objecto Obj tem a propriedade P se existir uma representação explícita dessa propriedade no enquadramento Obj, ou se Obj for membro de uma classe com essa propriedade.

O objecto *Obj* é membro da classe *Class* se existir a representação explícita de um arco *Member* de *Obj* para *Class*, ou se existir um arco *Member* de *Obj* para uma classe intermédia *Sub* e *Sub* for uma subclasse de *Class*.

Finalmente, *Sub* é uma subclasse de *Class* se existir um arco *subclass* de *Sub* para *Class*, ou se existir um arco *subclass* de *Sub* para uma classe intermédia C e C for uma subclasse de *Class*.

```
frame(X):-
   class(X).
frame(X):-
   object(X).

value(Frame, Prop, Val):-
   evalue(Frame, Prop, Val), !.
value(Frame, Prop, Val):-
   memb(Frame, Class),
   evalue(Class, Prop, Val), !.

memb(Obj, Class):-
   ememb(Obj, Class), !.
memb(Obj, Class):-
   ememb(Obj, C),
   subclass(C, Class), !.

subclass(Sub, Class):-
   esub(Sub, Class), !.
subclass(Sub, Class):-
   esub(Sub, C),
   subclass(C, Class), !.
```

Figura 10 – Definição dos mecanismos de herança

O enquadramento B_1 representa uma bicicleta com um motor auxiliar. Por um lado, de acordo com a sua representação explícita, a tracção de B_1 é motorizada. Por outro lado, de acordo com a herança de propriedades, B_1 a tracção de B_1 é animal porque B_1 pertence à classe das bicicletas, a qual é uma subclasse dos veículos de tracção animal. Para contornar este problema, convenciona-se que as propriedades herdadas de níveis hierárquicos inferiores sobrepõem-se às propriedades herdadas de níveis hierárquicos superiores. Usando esta convenção, dado que a tracção motorizada é herdada da própria definição do enquadramento, enquanto que a tracção animal é herdada de um nível superior da hierarquia, o enquadramento B_1 tem tracção motorizada. Este é o significado (implícito) da representação por enquadramentos. A implementação desta convenção é feita através do *"cut"* que aparece nas definições das relações implícitas. Devido ao *"cut"*, a primeira solução encontrada ao percorrer a hierarquia de baixo para cima, é a solução adoptada.

É devido à convenção da primazia das propriedades herdadas de níveis hierárquicos inferiores que se pode dizer que a herança é um tipo de raciocínio por omissão ("default reasoning").

A Figura 11 mostra uma interacção com o sistema de enquadramentos representados na Figura 8.

```
?- value(b1, traccao, X).
X=motor
yes

?- value(b1, A, V).
A=traccao, V=motor
yes

?- attribute(A), value(b1, A, V).
A=traccao, V=motor;
A=nrodas, V=2;
no

?- object(Obj), memb(Obj, carro).
Obj=c1;
Obj=c2;
no

?- object(Obj), value(Obj, nrodas, X), X > 2.
Obj=c1;
Obj=c2;
no
```

Figura 11 – Interacção com o sistema de enquadramentos da Figura 8

Na primeira interrogação, pergunta-se qual é o tipo de tracção de B1.

Na segunda interacção, pretende-se perguntar quais os valores de todos os atributos de B1. No entanto, esta interrogação não resulta porque, devido ao *"cut"* na definição das relações implícitas, o sistema apenas dá a primeira resposta encontrada. Para contornar esta contrariedade, recorre-se à relação *attributes/1*[1] de todos os atributos existentes no sistema. A terceira interrogação usa este mecanismo. *attributes/1* é usado como mecanismo gerador de soluções; *value/3* é usado para seleccionar apenas as soluções correctas.

Na quarta interrogação, pergunta-se quais os membros da classe dos carros. A relação *object/1* é usada para propor respostas para a pergunta; o predicado *memb/2* é usado para testar e seleccionar soluções correctas.

Na quinta interrogação, pergunta-se quais os objectos cujo número de rodas é superior a dois. *objects/1* é usado para propor soluções; a conjunção *value(Obj, nrodas, X), X>2* é usada para testar soluções propostas e seleccionar as correctas.

Apesar da flexibilidade exibida na Figura 11, o mecanismo implementado na Figura 10 tem diversos problemas. O primeiro problema é uma limitação do sistema: incapacidade de lidar com herança múltipla. O segundo problema é mais um erro do que uma limitação conceptual. Se perguntarmos se B1 tem tracção animal, o sistema responde afirmativamente porque o mecanismo de *"cut"* nas definições da Figura 10 apenas evita que o sistema **produzida** mais respostas além da primeira, mas não evita que o sistema **confirme** uma propriedade herdada de um nível superior da hierarquia se existir uma propriedade contraditória herdada de um nível inferior.

4.2 Herança de métodos num sistema de enquadramentos

Na secção anterior definiram-se os mecanismos de herança de propriedades numa árvore hierárquica de enquadramentos. Nesta secção alteram-se algumas definições para permitir que o valor de um atributo seja computado dinamicamente pela execução de um procedimento em vez de estabelecido estaticamente quando o enquadramento é criado.

Duas coisas são necessárias. A primeira consiste em definir uma forma de especificar que o valor de um atributo deve ser calculado através de um dado procedimento. A segunda consiste em redefinir a relação *value/3* de modo a que o procedimento especificado seja executado com os argumentos apropriados sempre que o valor de um atributo for determinado através desse procedimento.

Supondo que temos o enquadramento *pessoa* e diversos enquadramentos membros da classe das pessoas, cada um dos quais com um atributo para a data de nascimento. Se pretendermos saber a idade de cada pessoa, podemos definir um atributo dinâmico para a idade no enquadramento pessoa, o qual é computado através da execução de um procedimento que calcula idade a partir da data de nascimento (Figura 12).

[1] A relação *attributes* poderia ter dois argumentos em vez de um só. Com dois argumentos poderia estabelecer-se a relação dos atributos de cada enquadramento.

```
age(Date(Year1, Month1, Day1), Age):-
   Date(Date(Year2, Month2, Day2)),
   Month2 > Month1,
   !,
   Age is Year2 – Year1.
age(Date(Year1, Month1, Day1), Age):-
   Date(Date(Year2, Month2, Day2)),
   Month2 = Month1,
   Day2 >= Day1,
   !,
   Age is Year2 – Year1.
age(Date(Year1, Month1, Day1), Age):-
   Date(Date(Year2, Month2, Day2)),
   Age is Year2 – Year1 - 1.
```

Figura 12 – Procedimento em Prolog para calcular a idade

O sistema de enquadramentos descrito está representado na Figura 13. A classe *pessoa* tem o atributo *idade* cujo valor fica instanciado na variável Idade após a execução do procedimento *age/2*, ao qual se passa a data de nascimento da pessoa. O operador dois-pontos (':') foi usado para especificar o procedimento a executar.

```
attribute(nome).
attribute(idade).
attribute(dnascimento).

class(pessoa).
evalue(
   pessoa,
   idade,
   Idade:age(value-of(self, dnascimento), Idade)).

object(p1).
ememb(p1, pessoa).
evalue(p1, nome, 'Luis Botelho').
evalue(p1, dnascimento, Date(59, 12, 18)).

object(p2).
ememb(p2, pessoa).
evalue(p2, nome, 'Miguel Botelho').
evalue(p2, dnascimento, Date(94, 1, 6)).

object(p3).
ememb(p3, pessoa).
evalue(p3, nome, 'Catarina Botelho').
evalue(p3, dnascimento, Date(95, 3, 25)).
```

Figura 13 – Sistema de enquadramentos com propriedades dinâmicas

O sistema de enquadramentos não se limita a executar os procedimentos especificados nos atributos dos enquadramentos; é necessário efectuar primeiro uma operação de substituição de argumentos. Por exemplo, o operador *value-of* especifica o valor de um atributo de um enquadramento. Quando o procedimento *age/2* é chamado para calcular a idade da pessoa p1, em vez do termo *value-of(self, dnascimento)*, deve ser passado o valor do atributo *dnascimento* do enquadramento *p1*. *"self"* é uma constante especial que representa a identificação do enquadramento que herda o valor do atributo calculado pelo procedimento especificado na classe.

A Figura 14 apresenta a redefinição da relação implícita *value/3* a qual contempla a especificação de ranhuras com demónios.

```
value(Frame, A, V):-
   val(_, Frame, A, V).
value(Obj, A, V):-
   memb(Obj, Class),
   val(Obj, Class, A, V).

val(Self, Frame, A, V):-
   evalue(Frame, A, V:PSpec),
   !,
   replace_arguments(Self, Frame, PSpec, Procedure),
   call(Procedure).
val(_, Frame, A, V):-
   evalue(Frame, A, V), !.

replace_arguments(Self, Frame, PSpec, Procedure):-
   PSpec =.. [P|Args],
   replace_args_list(Self, Frame, Args, Replacement),
   Procedure =.. [P|Replacement].

replace_args_list(Self, Frame, [OldArg|OldArgs], [NewArg|NewArgs]):-
   replace_single_argument(Self, Frame, OldArg, NewArg),
   replace_args_list(Self, Frame, OldArgs, NewArgs).
replace_args_list(_, _, [], []).

replace_single_argument(Self, Frame, value-of(self, A), V):-
   !,
   value(Self, A, V).

replace_single_argument(Self, Frame, value-of(X, A), V):-
   !,
   value(X, A, V).
replace_single_argument(_, _, Arg, Arg).
```

Figura 14 – Valor de um atributo dinâmico

A definição de *value/3* com demónios é essencialmente a mesma que a anterior definição sem contemplar demónios. A diferença é a substituição de *evalue/3* pelo novo predicado *val/4*. *val(Self, Frame, Attribute, Value)* serve para determinar o valor de atributos possivelmente dinâmicos de enquadramentos sem usar herança.

Se um atributo dinâmico estiver definido numa classe e for necessário determinar o valor desse atributo numa instância dessa classe (i.e., num enquadramento membro da classe), pode ser necessário passar a identificação da classe para o demónio a ser executado. Essa identificação é passada no argumento Self de *val/4*.

O argumento Frame de *val/4* contém a identidade do próprio enquadramento onde o atributo está definido. Possivelmente, este argumento não é necessário.

Os argumentos Attribute e Value são respectivamente o nome e o valor de um atributo.

val/4 usa a especificação do atributo contida nos factos *evalue/3*. Se *evalue/3* tiver a especificação de um demónio, *val/4* executa o demónio, substituindo os argumentos pelos valores especificados. Se evalue/3 não contiver a especificação de um demónio, o valor devolvido por *val/4* será o valor contido no facto *evalue/3*.

replace_arguments/4 substitui a especificação dos argumentos que devem ser passados ao demónio pelos seus valores actuais. A constante *"self"* é substituída pela identificação do enquadramento que herda o valor computado pelo procedimento especificado na definição da classe. O operador *value-of(Frame, Attribute)* é substituído pelo valor do atributo Attribute no enquadramento Frame. Poderiam definir-se outras substituições para *replace_arguments/4*.

replace_arguments/4 usa *replace_arguments_list/4* para substituir toda a lista de especificações de argumentos pelos seus valores. *replace_arguments_list/4* usa *replace_single_argument/4* para substituir um argumento individual.