Projecto de Lógica para Programação

META-FORMS^{TM}

Licenciatura em Engenharia Informática e de Computadores
Instituto Superior Técnico

2011-2012

1 Introdução

Este projecto visa a implementação, em Prolog, de uma versão simplificada do jogo META-FORMS^{TM}, criado por Michael & Robert Lyons. O objectivo do jogo é colocar 9 figuras distintas num tabuleiro 3x3, respeitando um conjunto de pistas (cada conjunto de pistas será, doravante, denominado *desafio*). Cada desafio representa uma e uma só configuração do tabuleiro e pode assumir que em cada desafio são dadas exactamente 9 pistas, sendo cada uma relativa à posição de uma das figuras em jogo. A Figura 1 ilustra um desafio. A pista relativa ao círculo vermelho indica explicitamente a posição a ocupar no tabuleiro por esta peça. As pistas relativas ao círculo amarelo e quadrado azul são dadas por tabuleiros parciais. Poderá constatar, após uma breve reflexão, que a primeira indica que o círculo amarelo pode ocupar uma das quatro posições do quadrado 2x2 do topo esquerdo do tabuleiro; a segunda que o quadrado azul pode estar em qualquer posição da última coluna do tabuleiro. Todas as outras pistas, explicitam onde é que as figuras não podem ocorrer.

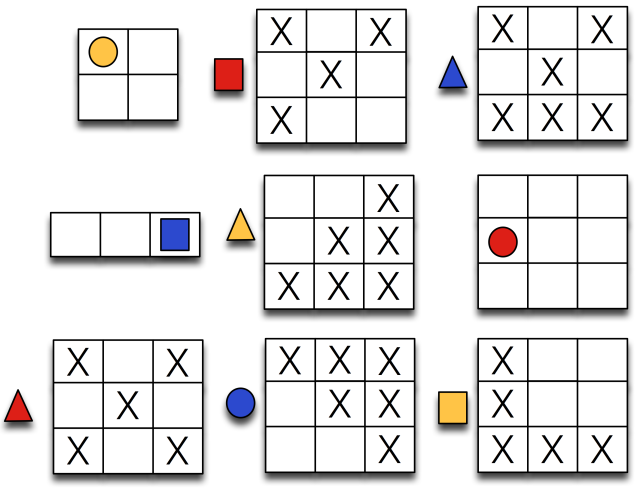

Figura 1: Exemplo de um desafio

A Figura 2 representa a solução deste desafio.

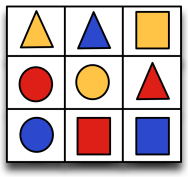

Figura 2: Solução do desafio anterior

Na secção 2 apresentam-se o jogo e as pistas/predicados a implementar, na secção 3 precisam-se alguns detalhes da implementação e, na secção 4, explica-se como deve ser feita a entrega e como vai ser avaliado o projecto.

2 O META-FORMS^{TM} em Prolog

2.1 O tabuleiro

Identifica-se cada casa do tabuleiro por um par, em que o primeiro elemento representa a linha e o segundo a coluna ocupados por essa casa, sendo as colunas identificadas pelas constantes `left`, `middle` e `right` e as linhas por `top`, `center` e `bottom`, tal como mostra a Figura 3.

(top, left)	(top, middle)	(top, right)
(center, left)	(center, middle)	(center, right)
(bottom, left)	(bottom, middle)	(bottom, right)

Figura 3: Denominação das casas do tabuleiro

2.2 As peças

Consideram-se três formas de peças – triângulo, círculo e quadrado – e três cores – azul, amarelo e vermelho. As nove peças são combinações de uma forma com uma cor, sendo as peças únicas para cada par (forma, cor), ou seja, existe um e apenas um triângulo amarelo, um azul, etc.

Cada peça será representada em Prolog através do functor `peca`, de aridade 2, sendo que as constantes `triangulo`, `circulo` e `quadrado` são usadas para representar, respectivamente, as formas triângulo, círculo e quadrado, e as constantes `azul`, `amarelo` e `vermelho` usadas para representar as cores com o

mesmo nome. Assim, por exemplo, o termo `peca(quadrado, azul)` representa o quadrado azul.

2.3 As pistas

2.3.1 Pistas básicas

A pista mais básica corresponde à indicação explícita do local onde a peça deve ocorrer no tabuleiro. Por exemplo, a pista dada na Figura 4, indica que o triângulo vermelho deve ocorre na posição (center, left).

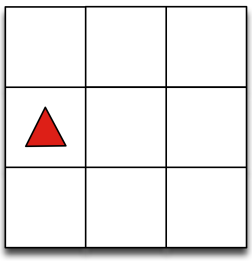

Figura 4: Ocorrência do triângulo vermelho em (center, left).

O predicado `coloca`/4 é usado para representar esta pista, em Prolog. O seu primeiro argumento indica a peça em jogo, o segundo e o terceiro a posição ocupada no tabuleiro (linha e coluna, respectivamente); o último argumento representa o tabuleiro em questão. Por exemplo, `coloca(peca(triangulo, vermelho), center, left, Tabuleiro)` representa a pista da Figura 4.

2.3.2 Pistas nível intermédio

Chamamos pistas intermédias às que representam indicações indirectas para a colocação de uma peça. São dadas através de representações parciais do tabuleiro, cujas casas assumimos designar como indicado na Figura 5.

Assim, por exemplo, a pista da Figura 6 corresponde a dizer que o triângulo vermelho tem de ser colocado numa posição central, seja na coluna da esquerda (`left`), do meio (`middle`) ou na da direita (`right`).

Para representar cada uma destas pistas consideram-se os predicados que se seguem (em que a variável `Peca` representa a peça em jogo, as variáveis `Linha` e `Coluna`, respectivamente, a linha e a coluna onde a peça deve ser colocada e `Tabuleiro` o tabuleiro em uso):

1. `rectanguloVertical(Peca, Linha, Coluna, Tabuleiro)`
2. `rectanguloHorizontal(Peca, Linha, Coluna, Tabuleiro)`
3. `linhaTriplaVertical(Peca, Linha, Tabuleiro)`
4. `linhaTriplaHorizontal(Peca, Coluna, Tabuleiro)`
5. `quadrado(Peca, Linha, Coluna, Tabuleiro)`
6. `linhaDuplaVertical(Peca, Linha, Tabuleiro)`
7. `linhaDuplaHorizontal(Peca, Coluna, Tabuleiro)`

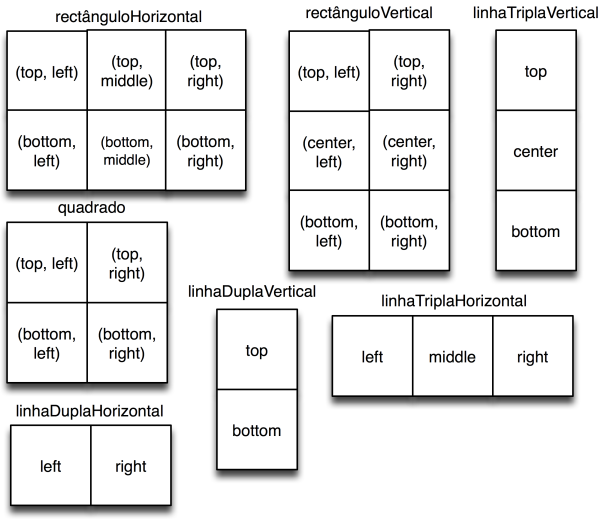

Figura 5: Figuras associadas às pistas intermédias.

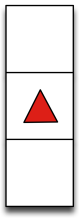

Figura 6: Pista relativa à localização do triângulo vermelho.

Como exemplo, a pista dada na Figura 6 seria representado, em Prolog, por `linhaTriplaVertical(peca(triangulo, vermelho), center, Tabuleiro)`,

2.3.3 Pistas nível avançado

As pistas ditas de nível avançado representam restrições indicando onde NÃO deve ser colocada uma peça. Estas pistas podem ser dadas por um tabuleiro 3x3, bem como pelos tabuleiros parciais correspondentes às pistas positivas (Figura 5). Assim, por exemplo, a Figura 7 representa duas pistas: a primeira indica que o triângulo vermelho não pode estar nas posições (top, left), (center, left) e (center, middle); a segunda indica que o triângulo vermelho não pode estar em nenhuma posição do centro.

Em termos de predicados, consideram-se os seguintes (assuma que a variável `Peca` representa a peça em jogo, `ListaNeg` a lista com as posições onde a peça não deve ser colocada (uma espécie de lista *negativa*) e `Tabuleiro` o tabuleiro em uso):

1. `matrizNeg(Peca, ListaNeg, Tabuleiro)`
2. `rectanguloVerticalNeg(Peca, ListaNeg, Tabuleiro)`

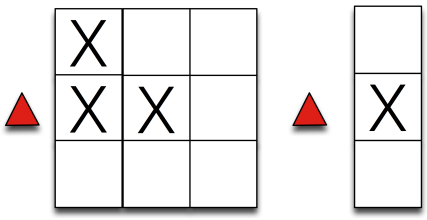

Figura 7: Pista de nível avançado sobre a localização do triângulo vermelho.

3. `rectanguloHorizontalNeg(Peca, ListaNeg, Tabuleiro)`
4. `linhaTriplaVerticalNeg(Peca, ListaNeg, Tabuleiro)`
5. `linhaTriplaHorizontalNeg(Peca, ListaNeg, Tabuleiro)`
6. `quadradoNeg(Peca, ListaNeg, Tabuleiro)`
7. `linhaDuplaVerticalNeg(Peca, ListaNeg, Tabuleiro)`
8. `linhaDuplaHorizontalNeg(Peca, ListaNeg, Tabuleiro)`

Deve ser claro que `ListaNeg` contém pares apenas nos casos da `matrizNeg`, `rectanguloVerticalNeg`, `rectanguloHorizontalNeg` e `quadradoNeg`, dado que nestas situações é necessário indicar as linhas e as colunas para definir as posições onde não ocorrem as peças. Por exemplo, as pistas da Figura 7 seriam indicadas recorrendo aos predicados `matrizNeg(peca(triangulo, vermelho), [(top, left), (center, left), (center, middle)], Tabuleiro)` e `linhaTriplaVerticalNeg(peca(triangulo, vermelho), [center], Tabuleiro)`, respectivamente.

3 Detalhes de implementação

3.1 Ferramenta

Recomenda-se o swi-prolog, dado que é o que vai ser usado na avaliação do projecto (em `http:/www.swi-prolog.org/` encontrará ferramentas para as várias plataformas).

3.2 Testes e formato dos dados

Os alunos têm à sua disposição uma bateria de testes representando vários desafios, servindo os ficheiros *testes.pl*, *resultados.txt* e *correProjecto.pl* para levar a cabo esses testes.

O ficheiro *testes.pl* contém 5 desafios e o ficheiro *resultados.txt* a sua solução. O ficheiro *correProjecto.pl* contém o código abaixo. Supõe-se que TODO o código dos alunos se encontra no ficheiro chamado EXACTAMENTE *metaforms.pl*, onde deverá estar definido o predicado `desafio`/2 cujo primeiro argumento é o identificador do desafio a realizar e o segundo o tabuleiro com a solução. O desafio é especificado recorrendo aos predicados apresentados anteriormente. Quanto ao tabuleiro com a solução, este é representado por uma lista com 9 elementos, em que as três primeiras casas correspondem à linha de topo, as três seguintes à linha do centro e as três últimas à linha inferior do tabuleiro (a correspondência deve ser feita da esquerda para a direita).

```
start :- carregaDados, desafios.
carregaDados :- ['metaforms.pl'], ['testes.pl'].
desafios :- desafio(1, Tabuleiro), write(Tabuleiro)...
```

Para testar o projecto sugere-se a utilização da seguinte linha de comando[1] que permitirá guardar os resultados do programa no ficheiro `myresultados.txt`:

```
swipl -s correProjecto.pl -g start -t halt  > myresultados.txt
```

A avaliação será feita em moldes semelhantes aos testes e é muito importante que sejam respeitadas todas estas indicações. A título de exemplo, a lista Tabuleiro deverá ter o seguinte conteúdo: [peca(circulo, amarelo), peca(quadrado, amarelo), peca(triangulo, amarelo), peca(triangulo, vermelho), peca(circulo, azul), peca(quadrado, azul), peca(circulo, vermelho), peca(quadrado, vermelho), peca(triangulo, azul)], após o seguinte desafio de *testes.pl*:

```
desafio(2, Tabuleiro) :-
coloca(peca(circulo, amarelo), top, left, Tabuleiro),
coloca(peca(quadrado, azul), center, right, Tabuleiro),
coloca(peca(quadrado, vermelho), bottom, middle, Tabuleiro),
coloca(peca(triangulo, vermelho), center, left, Tabuleiro),
coloca(peca(quadrado, amarelo), top, middle, Tabuleiro),
coloca(peca(triangulo, azul), bottom, right, Tabuleiro),
linhaTriplaHorizontal(peca(triangulo, amarelo), right, Tabuleiro),
linhaTriplaVertical(peca(circulo, vermelho), bottom, Tabuleiro),
coloca(peca(circulo, azul), center, middle, Tabuleiro).
```

4 Entrega e avaliação

O projecto deve ser entregue via Fénix e em papel, nos serviços apropriados, até às **15h00 do dia 21 de Maio de 2012**:

- Entrega via Fénix: zip (e não RAR!) – NUM-GRUPO.zip (ex: 01.zip) – contendo: a) O ficheiro *metaforms.pl*; b) o relatório, em formato .pdf – NUM-GRUPO.pdf (ex: 21.pdf). O ficheiro *metaforms.pl* deverá conter o código do projecto; o relatório – máximo de 5 páginas e 10 000 caracteres (incluindo espaços) – deverá incluir: a) uma breve descrição da implementação das diferentes funcionalidade e a motivação para as decisões tomadas; b) uma breve descrição dos principais problemas encontrados ao longo da realização do projecto e uma explicação das respectivas soluções.

- Entrega em papel do relatório e do código: folhas agrafadas ou numa capa, na reprografia do DEI (alunos da Alameda) ou na portaria do edifício do IST-Taguspark (alunos do Tagus).

Atenção que se a entrega levar até 24 horas de atraso serão retirados 2 valores à nota obtida no projecto; se a entrega tiver até 48 horas de atraso, o projecto será penalizado em 4 valores. Não serão aceites projectos a partir das 48 horas de atraso.

[1] Atenção que a chamada ao Prolog pode mudar de acordo com o sistema operativo.

Será levada a cabo uma **avaliação automática** com exercícios de vários níveis – básico (3 valores), intermédio (5 valores), avançado (4 valores) (todos diferentes dos usados nos testes); será igualmente tida em conta a **qualidade do código** – 4 valores (uso adequado do Prolog, comentários, paragrafação, ausência de *warnings*, nomes de predicados e de variáveis cuidadosamente escolhidos, etc.); o **relatório** vale 4 valores e os parâmetros de avaliação são a sua organização, clareza e qualidade da escrita.

Os alunos devem ainda ter em conta que:

- Projectos que não respeitem os formatos terão 0 na avaliação automática.
- Caso se detectem cópias os alunos terão 0 no projecto e não poderão fazer LP este semestre.
- Poderá haver uma discussão oral do projecto e/ou uma demonstração do seu funcionamento (a ser decidido caso a caso).

Bom trabalho e uma dica: *em Roma sê Romano*, ou seja, se vai trabalhar em Prolog, aproveite o que o Prolog tem para oferecer.