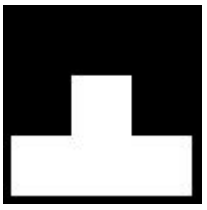
Terceiro Trabalho Prolog

INE5416 - Paradigmas de Programação
Aluno: João Vicente Souto
Matrícula: 16105151
Florianópolis, 24 de Novembro de 2017

1. Hit or miss

Foi implementado um predicado com comportamento similar ao descrito na descrição do trabalho sendo a detecção de um padrão T, estruturado como uma matriz, em uma imagem binária A, onde T é comporto por duas matrizes E e F, onde $E \subseteq F$.



Padrão E

Padrão F

A ideia principal é passar por todos os pontos da imagem A de forma a construir a imagem R no retorno da função recursiva como se fosse o append do prolog.

Principais predicados usados:

1. **hitmiss(A,E,F,R):** predicado inicial onde converte as matrizes em pontos.
2. **hitmiss_aux(...):** predicado auxiliar onde passo por todos os pontos de A (no caso S), e caso o ponto pertença ao padrão T, ele é inserido no imagem R com valor 1, caso contrário é inserido com valor 0.
    a. **hitmiss_aux(S, E, F, Rest):** duplica S para poder acessar o box do ponto.
    b. **hitmiss_aux([], _, _, _, []):** fim da recursão.
    c. **hitmiss_aux([(X, Y, V)|Tail], S, E, F, [(X, Y, 1)|Rest]):** pertence ao padrão.
    d. **hitmiss_aux([(X, Y, _)|Tail], S, E, F, [(X, Y, 0)|Rest]):** não pertence ao padrão.
3. **pattern_hit(...):** predicado responsável por verificar se pertence ao padrão.
    a. **pattern_hit([], _, _):** sem pontos.
    b. **pattern_hit([(X,Y,V)|Tail], E, F):** padrão E.
    c. **pattern_hit([(X,Y,V)|Tail], E, F):** padrão F.

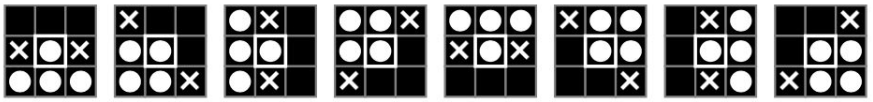
4. Outros predicados contendo os padrões e para construir o box do ponto (matriz contendo todos os pontos adjacentes com indices entre 0 e 3), não são o foco desse predicado então não vou detalhar aqui.

Exemplos do hitmiss na sequência: 1, 2, 3, 4, 5, 6, 7, 8 e 9.

Imagens originais junto dos arquivos.

1. Afinamento

Implementado seguindo o mesmo algoritmo contido na descrição do algoritmo onde uso os seguintes padrões:



A ideia principal foi passar pela imagem 8 vezes achando para cada uma das iterações os pontos que casam com os padrões e zerando eles. Nesta aplicação realizei a estratégia de acumulador, onde ao término da recursão já possuo a imagem R construída e só resta cortar a execução.

Principais predicados usados:

1. **afinamento(A, R):** início do afinamento onde converto a matriz em lista de pontos.
2. **afinamento_aux(...):** predicado que simula um while, enquanto tiver pontos, diferente de 0, reexecuta o afinamento.
    a. **afinamento_aux(R, R, 0):** finaliza operação de afinamento.
    b. **afinamento_aux(A, R, _):** refaz a operação de afinamento.
3. **encontra_inicio(A, Return, Q1):** Inicio do afinamento onde duplico A para poder pegar o box dos pontos.

4. **encontra_substitui(...):** para cada padrão executo o percorrimento da imagem A zerando os pontos que casam com o padrão e construindo a imagem R na descida da recursão.
    a. **encontra_substitui([], _, Return, Return, Q, Q, 315):** ultimo padrão da recursão.
    b. **encontra_substitui([], _, R0, Return, Q0, Q1, Etapa0):** final de uma das etapas indo para etapa + 45.
    c. **encontra_substitui([(X,Y,V)|Tail], A, R0, Return, Q0, Quant, Etapa):** casando com o padrão.
    d. **encontra_substitui([(X,Y,V)|Tail], A, R0, Return, Q0, Quant, Etapa):** não casa com o padrão.
5. **pattern_afinamento(B, Etapa):** Dependendo da etapa eu pego coordenadas diferentes do coord_pattern(...).
6. **pattern_aux(...):** verificador de padrão.
    a. **pattern_aux([], _):** fim da recursão.
    b. **pattern_aux([(X,Y,V)|Tail], P):** passando pelos pontos do box do ponto.
7. **pattern_or(...):** "Ou" criado para ignorar coordenadas com "x" nos padrões.
    a. pattern_or(x, _).
    b. pattern_or(0, 0).
    c. pattern_or(1, 1).

Exemplos do afinamento na sequência: 1, 2, 3, 4, 5, 6, 7, 8 e 9.

Imagens originais junto dos arquivos.

1. Aplicação: Contorno

A aplicação escolhida foi identificar o contorno das imagens, neste caso, dos exemplos disponibilizados pelo professor. O algorimo é similar ao do afinamento, porém é percorrido apenas uma vez a imagem A e para cada ponto é verificado se ele pertence pelo menos a um padrão.

Sobre os padrões procurou-se identificar os padrões pertencentes ao afinamento e mais alguns, como por exemplo, as quinas de uma imagem ou as pontas. Tais padrões são um tanto quanto distintos um dos outros por isso tive que implementar alguns predicados a mais do coord_pattern para abarcar esses padrões.

Principais predicados usados:

1. **aplicacao(A, R):** converte a imagem A para lista de pontos.
2. **contorno(A, R):** duplica a imagem A para poder ser possível encontrar o box de cada ponto e chama o predicado encontra_contorno(...).
3. **encontra_contorno(...):** aplico a estratégia de acumulador construindo a imagem R na descida da recursão.
    a. **encontra_contorno([], _, R0, R1):** fim da recursão.
    b. **encontra_contorno([(X, Y, V)|Tail], S, Rest, Return):** casa com algum padrão.
    c. **encontra_contorno([(X, Y, _)|Tail], S, Rest, Return):** não casa com nenhum padrão.
4. pattern_contorno(B): verifica se o ponto pertence a alguns dos padrões específicados.

Exemplos do afinamento na sequência: 1, 2, 3, 4, 5, 6, 7, 8 e 9.

Imagens originais junto dos arquivos.