PICAT: Uma Linguagem de Programação Multiparadigma

, Claudio Cesar de Sá, Miguel Alfredo Nunes, Jeferson L. R. Souza

miguel.nunes@edu.udesc.br

jeferson.souza@udesc.br

claudio.sa@udesc.br

Departamento de Ciência da Computação – DCC
Centro de Ciências e Tecnológias – CCT
Universidade do Estado de Santa Catarina – UDESC

23 de abril de 2019

Contribuições

- Alexandre Gonçalves;
- João Herique Faes Battisti;
- Paulo Victor de Aguiar;
- Rogério Eduardo da Silva;
- Hakan Kjellerstrand – (`http://www.hakank.org/picat/`)
- Neng-Fa Zhou – (`http://www.picat-lang.org/`)
- Outros anônimos que auxiliaram na produção deste documento;

Programação por Restrições (PR) – I

- A Programação por Restrições (PR) é conhecida por *Constraint Programming* ou simplesmente **CP**

Programação por Restrições (PR) – I

- A Programação por Restrições (PR) é conhecida por *Constraint Programming* ou simplesmente **CP**
- Uma poderosa teoria (e técnica) que contorna a complexidade de certos problemas exponenciais

Programação por Restrições (PR) – I

- A Programação por Restrições (PR) é conhecida por *Constraint Programming* ou simplesmente **CP**
- Uma poderosa teoria (e técnica) que contorna a complexidade de certos problemas exponenciais
- A PR encontrava-se inicialmente dentro da IA e PO, mas como várias outras, tornaram-se fortes e autônomas. Atualmente uma área de pesquisa bem forte em alguns países.

Programação por Restrições (PR) – II

- Aproximadamente o algoritmo da PR é dado:

Programação por Restrições (PR) – II

- Aproximadamente o algoritmo da PR é dado:
  1. Avaliar algebricamente os domínios das variáveis com suas restrições
  2. Intercala iterativamente a propagação de restrições com um algoritmo de busca
  3. A cada variável instanciada, o processo é repetido sobre as demais variáveis, reduzindo progressivamente o espaço de busca
  4. Volte ao passo inicial até que os domínios permaneçam estáticos e que as variáveis apresentem instâncias consistentes

Programação por Restrições (PR) – II

- Aproximadamente o algoritmo da PR é dado:
  1. Avaliar algebricamente os domínios das variáveis com suas restrições
  2. Intercala iterativamente a propagação de restrições com um algoritmo de busca
  3. A cada variável instanciada, o processo é repetido sobre as demais variáveis, reduzindo progressivamente o espaço de busca
  4. Volte ao passo inicial até que os domínios permaneçam estáticos e que as variáveis apresentem instâncias consistentes
- Este núcleo é uma busca por constantes otimizações

Programação por Restrições (PR) – II

- Aproximadamente o algoritmo da PR é dado:
  1. Avaliar algebricamente os domínios das variáveis com suas restrições
  2. Intercala iterativamente a propagação de restrições com um algoritmo de busca
  3. A cada variável instanciada, o processo é repetido sobre as demais variáveis, reduzindo progressivamente o espaço de busca
  4. Volte ao passo inicial até que os domínios permaneçam estáticos e que as variáveis apresentem instâncias consistentes
- Este núcleo é uma busca por constantes otimizações
- Uma das virtudes da PR: a legibilidade e clareza de suas soluções

Programação por Restrições (PR) – III

- Problemas combinatoriais com domínio nos inteiros são bons candidatos a serem resolvidos por PR

Programação por Restrições (PR) – III

- Problemas combinatoriais com domínio nos inteiros são bons candidatos a serem resolvidos por PR
- Quando temos problemas que precisamos conhecer **todas** as respostas, não apenas a melhor resposta

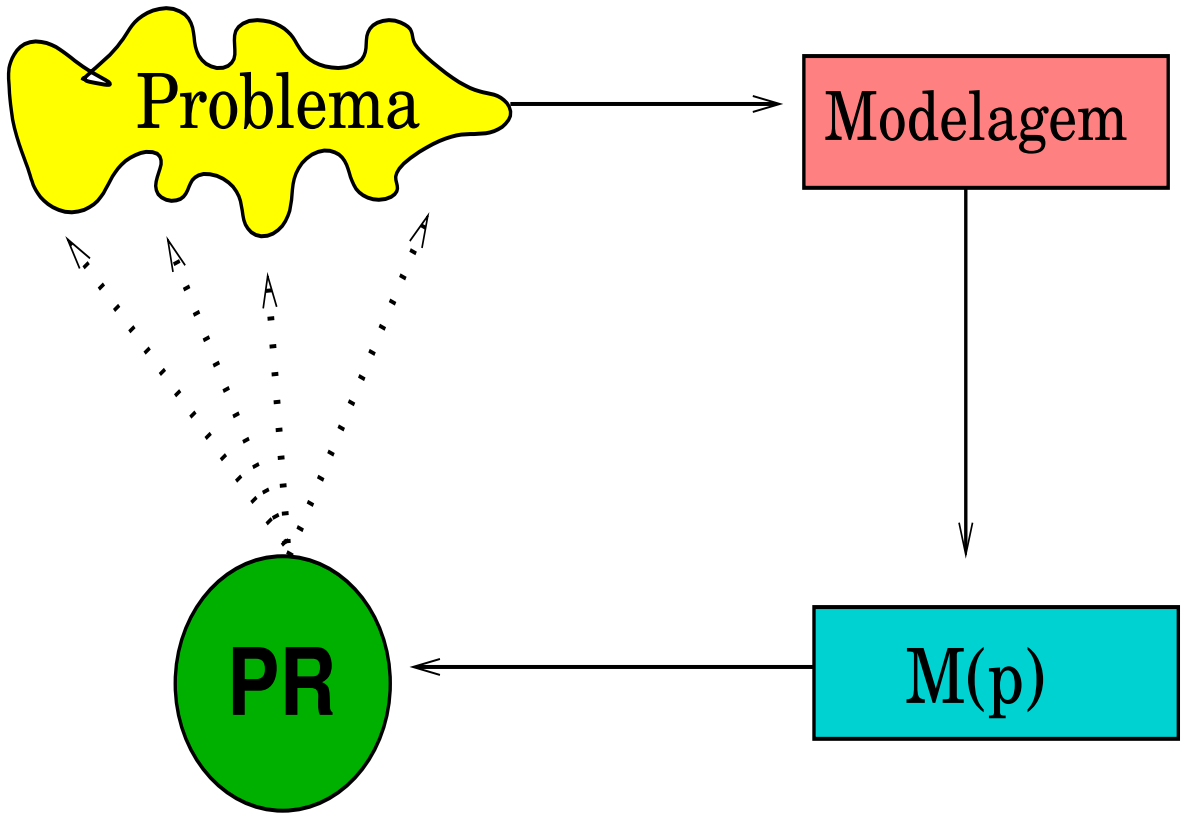
Programação por Restrições (PR) – III

- Problemas combinatoriais com domínio nos inteiros são bons candidatos a serem resolvidos por PR
- Quando temos problemas que precisamos conhecer **todas** as respostas, não apenas a melhor resposta
- Quando necessitamos de respostas *precisas* e não apenas as aproximadas. Há um custo computacional a ser pago aqui!

Programação por Restrições (PR) – III

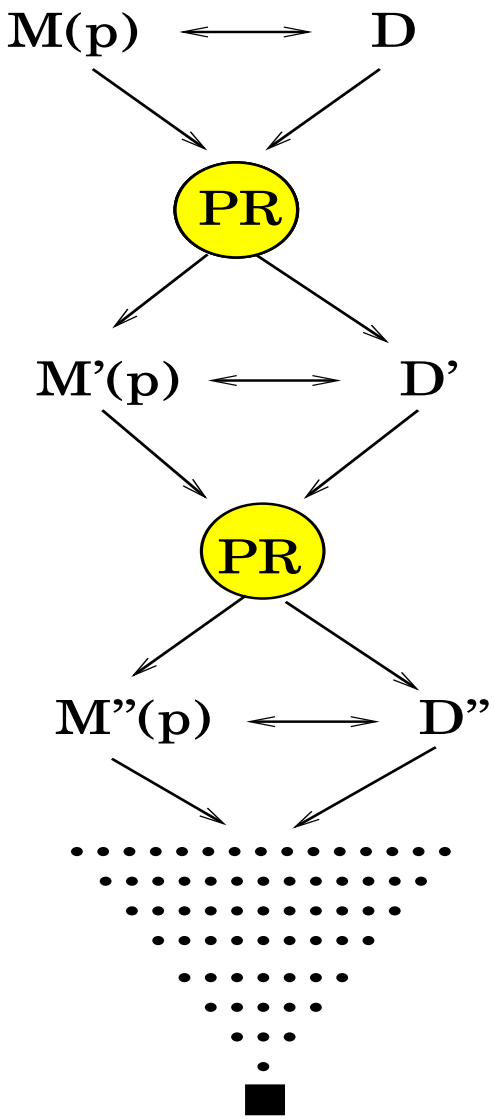
- Problemas combinatoriais com domínio nos inteiros são bons candidatos a serem resolvidos por PR
- Quando temos problemas que precisamos conhecer **todas** as respostas, não apenas a melhor resposta
- Quando necessitamos de respostas *precisas* e não apenas as aproximadas. Há um custo computacional a ser pago aqui!
-

Metodologia da Construção de Modelos



Um modelo

Fluxo de Cálculo da PR



Onde o objetivo da PR é:

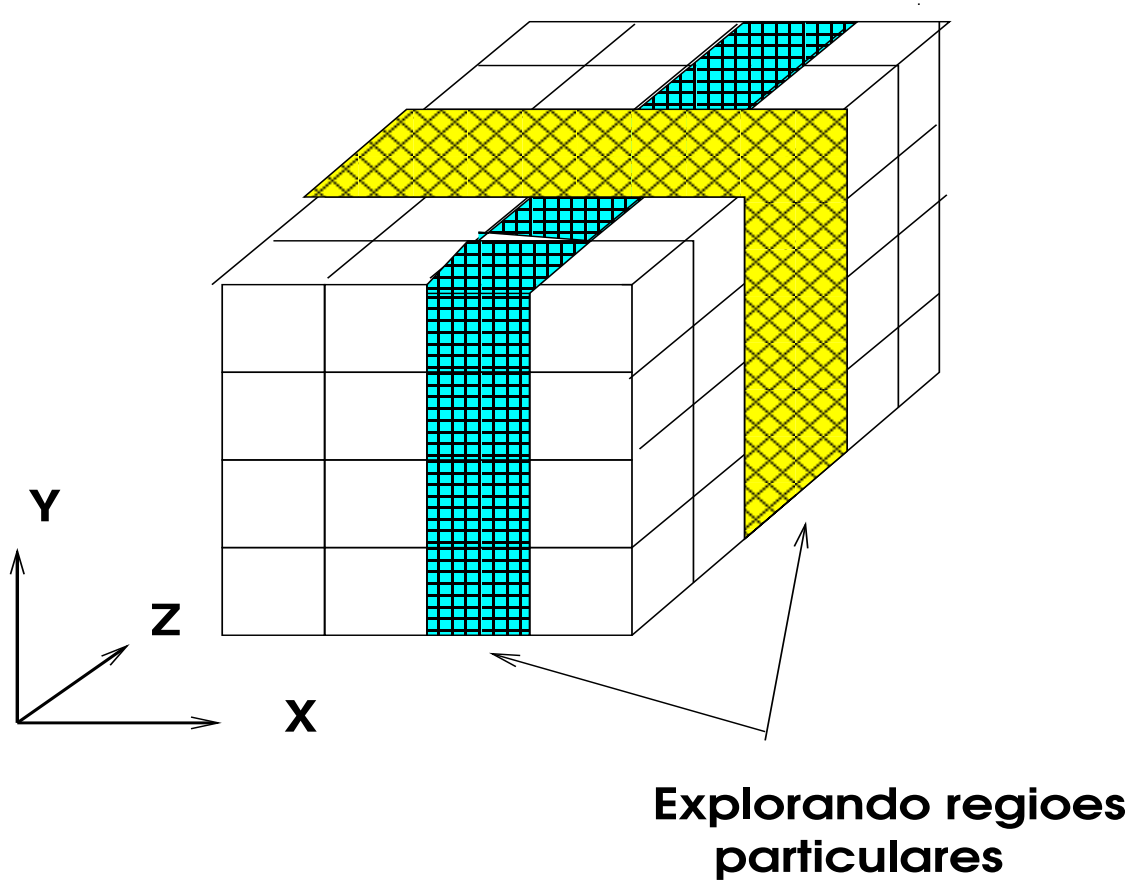


Figura 1: Realizar buscas com regiões reduzidas – promissoras (regiões factíveis de soluções)

Redução Iterativa em Sub-problemas

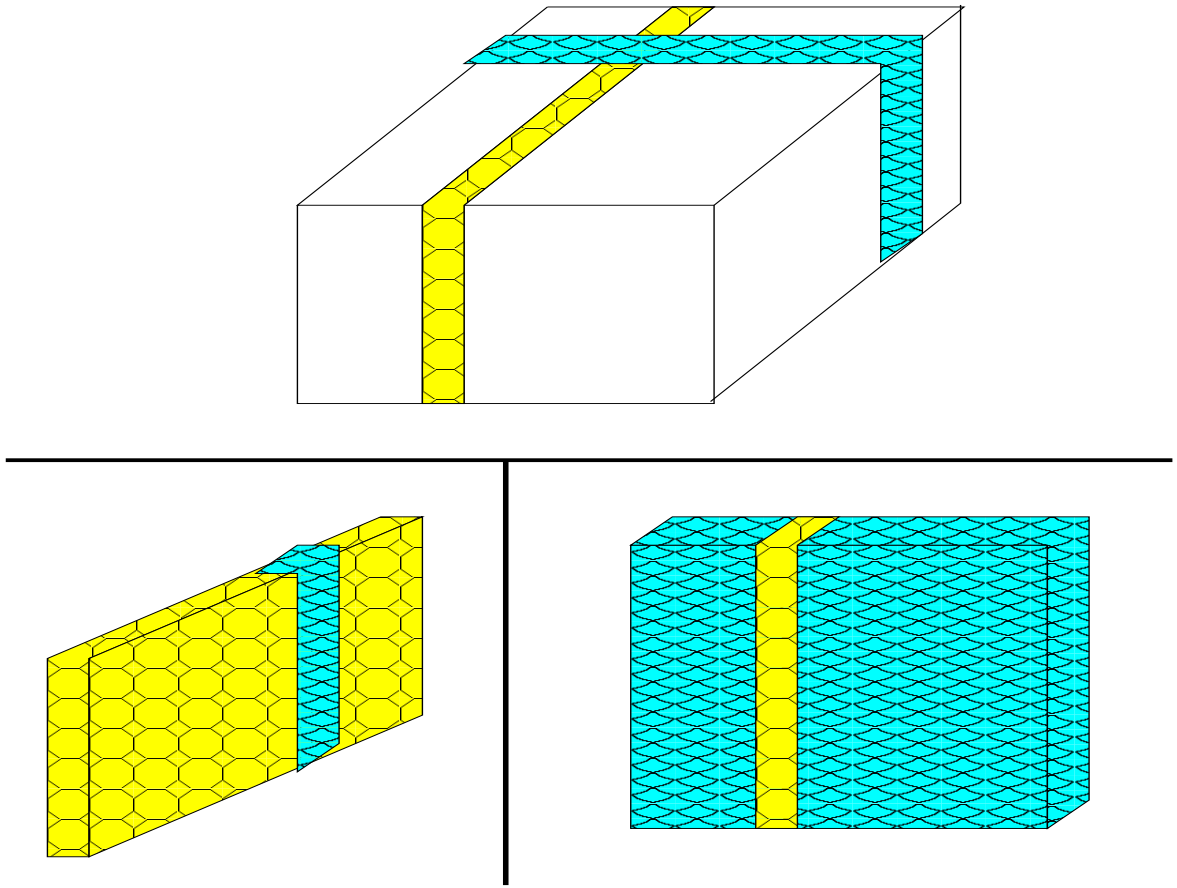


Figura 2: Redução de um CP em outros sub-problemas CPs equivalentes

A PR tem os seguintes elementos:

A PR tem os seguintes elementos:

- Um conjunto de **variáveis**: X_1, X_2, X_3, ..., X_n

A PR tem os seguintes elementos:

- Um conjunto de **variáveis**: X_1, X_2, X_3, ..., X_n
- Um conjunto de **domínios** dessas variáveis: D_{X_1}, D_{X_2}, D_{X_3}, ..., D_{X_n}

A PR tem os seguintes elementos:

- Um conjunto de **variáveis**: X_1, X_2, X_3, ..., X_n
- Um conjunto de **domínios** dessas variáveis: D_{X_1}, D_{X_2}, D_{X_3}, ..., D_{X_n}
- Finalmente, as **restrições**, que são relações n-árias entre estas variáveis

A PR tem os seguintes elementos:

- Um conjunto de **variáveis**: X_1, X_2, X_3, ..., X_n
- Um conjunto de **domínios** dessas variáveis: D_{X_1}, D_{X_2}, D_{X_3}, ..., D_{X_n}
- Finalmente, as **restrições**, que são relações n-árias entre estas variáveis
- Exemplo: $D_{X_1} = D_{X_2} = \{3, 4\}$ e $X_1 \neq X_2$

- Para o exemplo anterior um código em Picat é dado por:

- Para o exemplo anterior um código em Picat é dado por:
  - `[X1, X2] ::  3..4`
  - `X1 #!= X2`

- Para o exemplo anterior um código em Picat é dado por:
  - `[X1, X2] :: 3..4`
  - `X1 #!= X2`
- Em resumo, as relações da PR tem o símbolo '#'

- Para o exemplo anterior um código em Picat é dado por:
  - `[X1, X2] :: 3..4`
  - `X1 #!= X2`
- Em resumo, as relações da PR tem o símbolo '#'
- Para tornar toda esta sintaxe da PR disponível, Picat tem um módulo para suporte da PR: `import cp`

Exemplo – 01 – Soma de Números Primos

- Dado um número par qualquer, encontre dois de números primos, N_1 e N_2, diferentes entre si, que somados deêm este número par.

Exemplo – 01 – Soma de Números Primos

- Dado um número par qualquer, encontre dois de números primos, N_1 e N_2, diferentes entre si, que somados deêm este número par.
- Exemplo:
  Seja o PAR $= 18$
  Uma soluç ao:
  $N_1 = 7$ e $N_2 = 11$
  pois
  $N_1 + N_2 = 18$

- N_1 e N_2 assumem valores no domínio dos números primos. Logo, é importante ter os números primos prontos!

Modelagem do Problema

- N_1 e N_2 assumem valores no domínio dos números primos. Logo, é importante ter os números primos prontos!
- A soma destes números é o par fornecido como entrada, N_{PAR}: $N_1 + N_2 = N_{PAR}$

Modelagem do Problema

- N_1 e N_2 assumem valores no domínio dos números primos. Logo, é importante ter os números primos prontos!
- A soma destes números é o par fornecido como entrada, N_{PAR}: $N_1 + N_2 = N_{PAR}$
- N_1 e N_2 são diferentes entre si $N_1 \neq N_2$

Modelagem do Problema

- N_1 e N_2 assumem valores no domínio dos números primos.
  Logo, é importante ter os números primos prontos!
- A soma destes números é o par fornecido como entrada, N_{PAR}:
  $N_1 + N_2 = N_{PAR}$
- N_1 e N_2 são diferentes entre si
  $N_1 \neq N_2$
- Como são inteiros: $N_1 < N_{PAR}$ e $N_2 < N_{PAR}$
  Sim, é óbvio, mas isto faz uma redução significativa de domínio!

Código Completo

- Acompanhar as explicações do código de: `https://github.com/claudiosa/CCS/blob/master/picat/soma_N1_N2_primos_CP.pi`
- Confira a execução e testes

```
modelo =>
    PAR = 382,
    Variaveis = [N1,N2],
    % Gerando um domino soh de primos
    % L_dom = [I : I in 1..1000, eh_primo(I) == true],   %OU
    L_dom = [I : I in 1..1000, prime(I)],
    Variaveis :: L_dom,
```

Uma ótima estratégia: sair com um domínio de números candidatos!

```
    % RESTRICOES
    N1 #!= N2,
    N1 #< PAR,
    N2 #< PAR,
    N1 + N2 #= PAR,

% A BUSCA
solve([ff], Variaveis),
  % UMA SAIDA
printf("\n  N1: %d\t N2: %d", N1,N2),
printf("\n.....................................")
.
```

```
import cp.

% main => modelo .
% main ?=> modelo, fail.
% main =>  true.

main =>
    L = findall(_, $modelo),
    writef("\n Total de solucoes:  %d \n", length(L)) .
```

Saída – I

```
Picat> cl('soma_N1_N2_primos_CP').
Compiling:: soma_N1_N2_primos_CP.pi
** Warning  : redefine_preimported_symbol(math): prime / 1
soma_N1_N2_primos_CP.pi compiled in 7 milliseconds
loading...

yes

Picat> main.

  N1: 3  N2: 379
.....................................
  N1: 23  N2: 359
.....................................
  N1: 29  N2: 353
.....................................
```

Saída – II

```
 ....................................
  N1: 353  N2: 29
....................................
  N1: 359  N2: 23
....................................
  N1: 379  N2: 3
....................................
 Total de solucoes:  18

yes

Picat>
```

Exemplo – 02 – Escala de Consultórios

- Seja um Posto Atendimento Médico, um PA, com 4 consultórios e 7 especialidades médicas

Exemplo – 02 – Escala de Consultórios

- Seja um Posto Atendimento Médico, um PA, com 4 consultórios e 7 especialidades médicas
- O problema é distribuir estes médicos nestes 4 consultórios tal que alguns requisitos sejam atendidos (restrições satisfeitas)

Exemplo – 02 – Escala de Consultórios

- Seja um Posto Atendimento Médico, um PA, com 4 consultórios e 7 especialidades médicas
- O problema é distribuir estes médicos nestes 4 consultórios tal que alguns requisitos sejam atendidos (restrições satisfeitas)
- A abordagem aqui é ingênua e sem muitos critérios

Modelagem do Problema

- Vamos usar uma matriz bi-dimensional para representar o problema. Linhas $\leftrightarrow$ consultórios (1 a 4), e as colunas $\leftrightarrow$ dias da semana (1 a 5)

Modelagem do Problema

- Vamos usar uma matriz bi-dimensional para representar o problema. Linhas $\leftrightarrow$ consultórios (1 a 4), e as colunas $\leftrightarrow$ dias da semana (1 a 5)
- Esta matriz será preenchida com valores/códigos de 1 a 7, de acordo com a especialidade médica.

Modelagem do Problema

- Vamos usar uma matriz bi-dimensional para representar o problema. Linhas $\leftrightarrow$ consultórios (1 a 4), e as colunas $\leftrightarrow$ dias da semana (1 a 5)
- Esta matriz será preenchida com valores/códigos de 1 a 7, de acordo com a especialidade médica.
- Assim o domínio da matriz `Quadro` (4×5) será preenchida com um destes códigos.

Modelagem do Problema

- Vamos usar uma matriz bi-dimensional para representar o problema. Linhas $\leftrightarrow$ consultórios (1 a 4), e as colunas $\leftrightarrow$ dias da semana (1 a 5)
- Esta matriz será preenchida com valores/códigos de 1 a 7, de acordo com a especialidade médica.
- Assim o domínio da matriz `Quadro` (4×5) será preenchida com um destes códigos.
- Vamos utilizar restrições globais: `member` e `all_different`

Modelagem do Problema

- Vamos usar uma matriz bi-dimensional para representar o problema. Linhas $\leftrightarrow$ consultórios (1 a 4), e as colunas $\leftrightarrow$ dias da semana (1 a 5)
- Esta matriz será preenchida com valores/códigos de 1 a 7, de acordo com a especialidade médica.
- Assim o domínio da matriz `Quadro` (4×5) será preenchida com um destes códigos.
- Vamos utilizar restrições globais: `member` e `all_different`
- As restrições globais se aplicam sobre um conjunto de variáveis.

- A fase de busca e propagação do comando `solve(Critérios, Variáveis)`, há dezenas de combinações possíveis: consultar o guia do usuário

Modelagem – Comentários

- A fase de busca e propagação do comando `solve(Critérios, Variáveis)`, há dezenas de combinações possíveis: consultar o guia do usuário
- Tem-se os predicados extras ... são muitos, todos os da CP

- A fase de busca e propagação do comando `solve(Critérios, Variáveis)`, há dezenas de combinações possíveis: consultar o guia do usuário
- Tem-se os predicados extras ... são muitos, todos os da CP
- Finalmente, exemplos sofisticados– de PR com PICAT: `http://www.hakank.org/picat/` – ***My Picat page*** – por Hakan Kjellerstrand

Código Completo

- Acompanhar as explicações do código de: `https://github.com/claudiosa/CCS/blob/master/picat/horario_medico_CP.pi`
- Confira a execução e testes

Código em Partes

```
modelo =>
    Dias = 5, % segunda= 1, ...., sexta-feira = 5
    Consultorio = 4,
    L_dom = [ oftalmo, otorrino, pediatra,  gineco,
%                 1         2          3          4
              cardio, dermato, clin_geral ],
%                 5        6         7
   Quadro = new_array(Consultorio, Dias ), %% Lin x Col
   Quadro :: 1 .. L_dom.len , %% operador len . "eh colado"
...
```

Código em Partes

```
    %% O medico 2 NUNCA trabalha no consultorio 1
    foreach ( J in 1 .. Dias )
        Quadro[1,J] #!= 2
    end,

    %% O medico 5 NUNCA trabalha no consultorio 4
    foreach ( J in 1 .. Dias )
        Quadro[4,J] #!= 5
    end,
...
```

Código em Partes

```
%% O Clin Geral deve vir o maior numero de dias ...
%% Esta restricao eh matematicamente é HARD
 foreach ( I in 1 .. Consultorio )
   member(7,[Quadro[I,J] : J in 1..Dias])
 end,

%% Ninguém trabalha no mesmo consultorio em dias seguidos
foreach ( J in 1 .. Dias )
    all_different( [Quadro[I,J] : I in 1..Consultorio] )
end,

%% Ninguém trabalha no mesmo dia em mais de um consultorio
 foreach ( I in 1 .. Consultorio )
    all_different( [Quadro[I,J] : J in 1..Dias] )
 end,
...
```

Código em Partes

```
% A BUSCA
solve([ff], Quadro),
  % UMA SAIDA

   printf("\n Uma escolha:"),
   print_matrix( Quadro ),
   print_matrix_NAMES( Quadro , L_dom ),
 printf(".............................\n") .
```

Código em Partes

```
print_matrix_NAMES( M, Lista ) =>
 L = M.length,
 C = M[1].length,
  nl,
   foreach(I in 1  .. L)
     foreach(J in 1  ..  C)
      printf(":%w \t" , print_n_lista( M[I,J], Lista) )
     % printf("(%d,%d): %w " , I, J, M[I,J] ) -- FINE
     end,
     nl
   end.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
print_n_lista( _, [] ) =  [].
print_n_lista( 1, [A|_] ) = A.
print_n_lista( N, [_|B] ) = print_n_lista( (N-1), B ) .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Saída - I

```
Picat> cl('horario_medico_CP.pi').
Compiling:: horario_medico_CP.pi
horario_medico_CP.pi compiled in 10 milliseconds
loading...

yes

Picat> main

 Uma escolha:
7 1 3 4 5
4 7 2 3 1
1 3 7 5 2
3 2 1 7 4
```

Saída - II

```
:clin_geral :oftalmo :pediatra :gineco :cardio
:gineco :clin_geral :otorrino :pediatra :oftalmo
:oftalmo :pediatra :clin_geral :cardio :otorrino
:pediatra :otorrino :oftalmo :clin_geral :gineco
.....................................
yes
```

Saída - III

```
$ time(picat horario_medico_CP.pi )

 Uma escolha:
7 1 3 4 5
4 7 2 3 1
1 3 7 5 2
3 2 1 7 4

:clin_geral  :oftalmo  :pediatra  :gineco  :cardio
:gineco  :clin_geral  :otorrino  :pediatra  :oftalmo
:oftalmo  :pediatra  :clin_geral  :cardio  :otorrino
:pediatra  :otorrino  :oftalmo  :clin_geral  :gineco
.....................................
real 0m0,023s
user 0m0,007s
sys 0m0,013s
[ccs@gerzat picat]$
```

Reflexões

- Há outros métodos para se resolver estes problemas.
  Exemplo: Programação Linear, Buscas Heurísticas, etc

Reflexões

- Há outros métodos para se resolver estes problemas.
  Exemplo: Programação Linear, Buscas Heurísticas, etc
- As restrições globais se aplicam sobre um conjunto de variáveis e há muitas outras importantes disponíveis no Picat

Reflexões

- Há outros métodos para se resolver estes problemas.
  Exemplo: Programação Linear, Buscas Heurísticas, etc
- As restrições globais se aplicam sobre um conjunto de variáveis e há muitas outras importantes disponíveis no Picat
- A área é extensa, contudo, Picat adere há todos requisitos da PR

Reflexões

- Há outros métodos para se resolver estes problemas.
  Exemplo: Programação Linear, Buscas Heurísticas, etc
- As restrições globais se aplicam sobre um conjunto de variáveis e há muitas outras importantes disponíveis no Picat
- A área é extensa, contudo, Picat adere há todos requisitos da PR
- Resumo da PR: segue por uma notação/manipulação algébrica restrita, simplificar e bissecionar as restrições, instanciar variáveis, verificar inconsistências, avançar sobre as demais variáveis, até que todas estejam instanciadas.