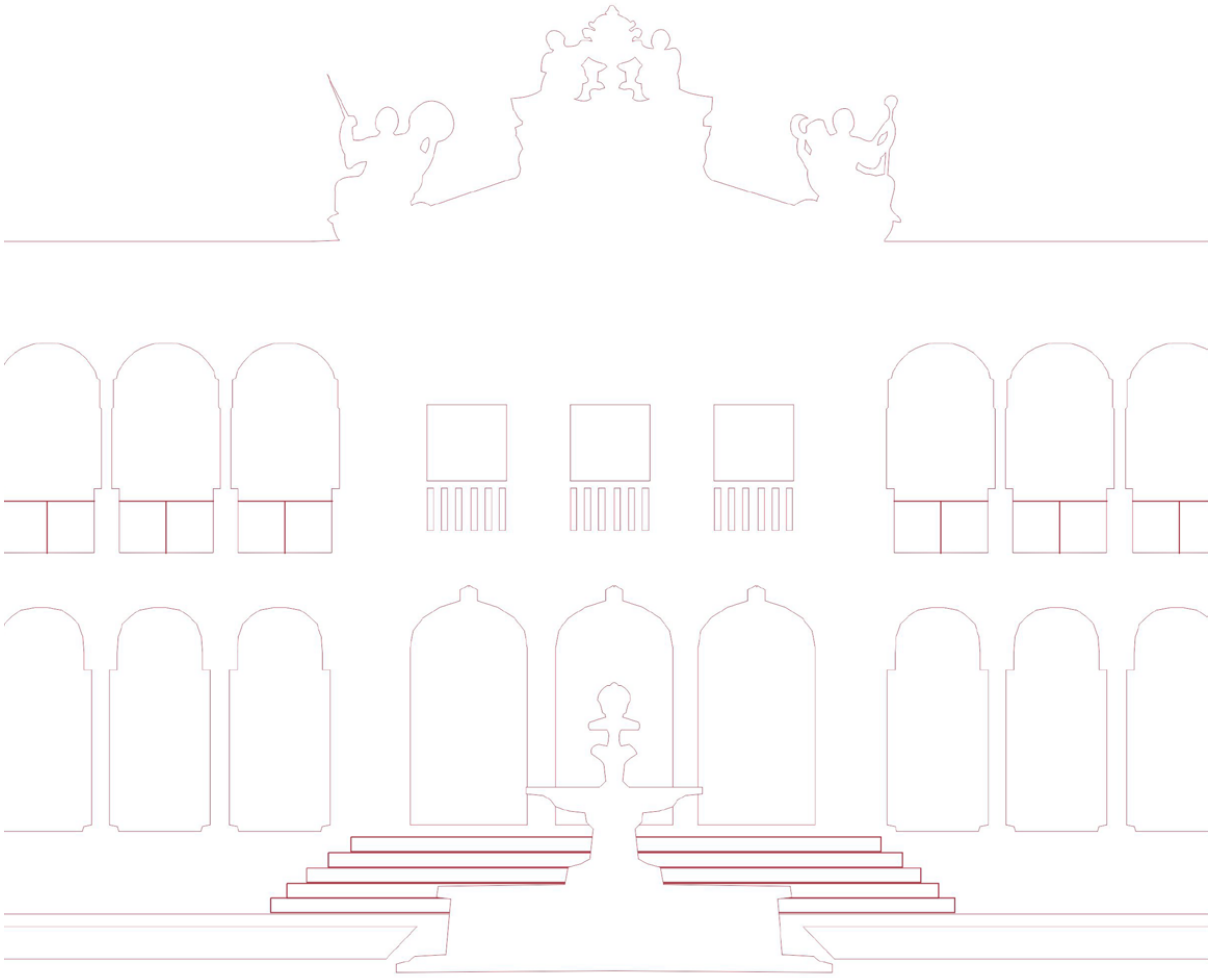
UNIVERSIDADE DE ÉVORA
ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

T2-IA



Gustavo Oliveira nº46395, Berke Balci nº64498, Semiha Çetintaş nº64751

Évora, 25 de junho de 2025

Considere o problema de resolver um Kakuro.

Um Kakuro tem um tabuleiro de nxm com casas em branco que devem ser preenchidas com números de 1 a 9. Todas as casas em branco numa linha ou numa coluna devem ter números diferentes. As casas preenchidas com \x, com x um valor inteiro, significam que a soma dos números nas casas nessa linha devem somar x. As casas preenchidas com y\, com y um valor inteiro, significam que a soma dos números nas casas nessa coluna devem somar y.

Questão 1 - Represente este problema como um problema de satisfação de restrições.

A - As variáveis que variáveis considera e o que representam.

As variáveis desse problema são cada uma das casas em branco a serem preenchidas com um valor inteiro.

B - O domínio das variáve

O domínio das variáveis corresponde aos valores inteiros que podem assumir.

```
Dominio = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

C - Restrições.

As casas preenchidas com \x, com x um valor inteiro, Devem ter a soma dos inteiros nessa linha igual a x. As casas preenchidas com y\, com y um valor inteiro, Devem ter a soma dos inteiros nessa coluna igual a y.

D - Estado Inicial.

```
estado_inicial(e([
    v(c(1), [1,2,3,4,5,6,7,8,9], _),
    v(c(2), [1,2,3,4,5,6,7,8,9], _),
    v(c(3), [1,2,3,4,5,6,7,8,9], _),
    v(c(4), [1,2,3,4,5,6,7,8,9], _),
    v(c(5), [1,2,3,4,5,6,7,8,9], _),
    v(c(6), [1,2,3,4,5,6,7,8,9], _),
    v(c(7), [1,2,3,4,5,6,7,8,9], _),
    v(c(8), [1,2,3,4,5,6,7,8,9], _),
    v(c(9), [1,2,3,4,5,6,7,8,9], _),
    v(c(10), [1,2,3,4,5,6,7,8,9], _),
    v(c(11), [1,2,3,4,5,6,7,8,9], _)
], [])).
```

E- Código do predicado verifica restrições .

```
% ---------------------------
% Grupos de soma (exemplo)
% ---------------------------

grupo(h1, [c(3), c(7), c(10)], 23).
grupo(h2, [c(4), c(8), c(11)], 9).
grupo(h3, [c(1), c(5), c(9)], 24).
grupo(h4, [c(2), c(6)], 15).

grupo(v1, [c(1), c(2)], 13).
grupo(v2, [c(3), c(4), c(5), c(6)], 24).
grupo(v3, [c(7), c(8), c(9)], 23).
grupo(v4, [c(10), c(11)], 11).

% ---------------------------
% Restri   o geral
% ---------------------------

```

```
ve_restricoes(e(_, Afectadas)) :-
    forall(grupo(_, Casas, Soma),
           verifica_grupo_kakuro(Casas, Soma, Afectadas)).

% ----------------------------
% Valida  o de cada grupo
% ----------------------------

verifica_grupo_kakuro(Casas, Soma, Atribuidas) :-
    % extrai apenas as casas deste grupo j  atribu das
    findall(V, (member(v(c(ID),_,V), Atribuidas), member(c(ID), Casas), nonvar(V)),
        Valores),
    all_different(Valores),
    sumlist(Valores, Parcial),
    Parcial =< Soma,
    ( length(Valores, L), length(Casas, L) -> Parcial = Soma ; true ).

% ----------------------------
% Auxiliares
% ----------------------------

all_different([]).
all_different([X|Xs]) :-
    \+ member(X, Xs),
    all_different(Xs).

sumlist([], 0).
sumlist([H|T], S) :-
    sumlist(T, ST),
    S is H + ST.
```

Questão 2 - Resolva o problema com o algoritmo de backtracking. No relatório, indique o número de nós visitados até à primeira solução e para encontrar todas as soluções.

- - - - -

Questão 3 - Resolva o problema modificando o algoritmo anterior para que faça a verificação para a frente (forward checking). No relatório, indique o número de nós visitado até à primeira solução e para encontrar todas as soluções.

- - - - -

Questão 4 - Modifique o algoritmo anterior como entender de forma melhorar a complexidade (temporal e espacial). Sugestão: pode alterar a ordem para atribuir o valor às variáveis, por exemplo, as variáveis que têm mais restrições ou a variável que tem menos valores no domínio. No relatório, indique o número de nós visitado até à primeira solução e para encontrar todas as soluções.

- - - - -

Questão 5 - Resolva o exemplo abaixo 7x7 como um problema de satisfação de restrições.

A - As variáveis que variáveis considera e o que representam.

As variáveis desse problema são cada uma das casas em branco a serem preenchidas com um valor inteiro.

B - O domínio das variáve

O domínio das variáveis corresponde aos valores inteiros que podem assumir.

```
Dominio = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

C - Restrições.

As casas preenchidas com \x, com x um valor inteiro, Devem ter a soma dos inteiros nessa linha igual a x. As casas preenchidas com y\, com y um valor inteiro, Devem ter a soma dos inteiros nessa coluna igual a y.

D - Estado Inicial.

```
estado_inicial(e([
    v(c(1), [1,2,3,4,5,6,7,8,9], _),
    v(c(2), [1,2,3,4,5,6,7,8,9], _),
    v(c(4), [1,2,3,4,5,6,7,8,9], _),
    v(c(5), [1,2,3,4,5,6,7,8,9], _),
    v(c(6), [1,2,3,4,5,6,7,8,9], _),
    v(c(8), [1,2,3,4,5,6,7,8,9], _),
    v(c(9), [1,2,3,4,5,6,7,8,9], _),
    v(c(10), [1,2,3,4,5,6,7,8,9], _),
    v(c(11), [1,2,3,4,5,6,7,8,9], _),
    v(c(12), [1,2,3,4,5,6,7,8,9], _),
    v(c(13), [1,2,3,4,5,6,7,8,9], _),
    v(c(16), [1,2,3,4,5,6,7,8,9], _),
    v(c(17), [1,2,3,4,5,6,7,8,9], _),
    v(c(19), [1,2,3,4,5,6,7,8,9], _)
],
[
    v(c(3), [1,2,3,4,5,6,7,8,9], 4),
    v(c(7), [1,2,3,4,5,6,7,8,9], 4),
    v(c(14), [1,2,3,4,5,6,7,8,9], 2),
    v(c(15), [1,2,3,4,5,6,7,8,9], 5),
    v(c(18), [1,2,3,4,5,6,7,8,9], 8)
])).
```

E- Código do predicado verifica restrições .

```
% ----------------------------
% Grupos de soma (exemplo)
% ----------------------------

grupo(h1, [c(1), c(4)], 13).
grupo(h2, [c(13), c(17)], 11).
grupo(h3, [c(2), c(5), c(9), c(14), c(18)], 26).
grupo(h4, [c(3), c(6), c(10), c(15), c(19)], 28).
grupo(h5, [c(7), c(11), c(16)], 18).
grupo(h6, [c(8), c(12)], 3).

grupo(v1, [c(1), c(2), c(3)], 20).
grupo(v2, [c(4), c(5), c(6), c(7), c(8)], 23).
grupo(v3, [c(9), c(10), c(11), c(12)], 14).
grupo(v4, [c(13), c(14), c(15), c(16)], 23).
grupo(v5, [c(17), c(18), c(19)], 19 ).

% ----------------------------
% Restri   o geral
% ----------------------------

ve_restricoes(e(_, Afectadas)) :-
    forall(grupo(_, Casas, Soma),
           verifica_grupo_kakuro(Casas, Soma, Afectadas)).

% ----------------------------
% Valida   o de cada grupo
% ----------------------------

verifica_grupo_kakuro(Casas, Soma, Atribuidas) :-
    % extrai apenas as casas deste grupo j   atribu das
    findall(V, (member(v(c(ID),_,V), Atribuidas), member(c(ID), Casas), nonvar(V)),
        Valores),
    all_different(Valores),
    sumlist(Valores, Parcial),
```

```
35      Parcial =< Soma,
36      ( length(Valores, L), length(Casas, L) -> Parcial = Soma ; true ).
37
38  % ----------------------------
39  % Auxiliares
40  % ----------------------------
41
42  all_different([]).
43  all_different([X|Xs]) :-
44      \+ member(X, Xs),
45      all_different(Xs).
46
47  sumlist([], 0).
48  sumlist([H|T], S) :-
49      sumlist(T, ST),
50      S is H + ST.
```

F - O algoritmo que usou para resolver o problema

```
1   % ----------------------------
2   % Algoritmo Utilizado - Backpropagation
3   % ----------------------------
4
5   back(e([], A), A, N, N).
6   back(E, Sol, NIn, NOut) :-
7       sucessor(E, E1),
8       ve_restricoes(E1),
9       N1 is NIn + 1,
10      back(E1, Sol, N1, NOut).
11
12  sucessor(e([v(N,D,V)|R],E),e(R,[v(N,D,V)|E])):- member(V,D).
```

G - O número de nós visitados até à primeira solução e o número de nós para encontrar todas as soluções.

- - - - -